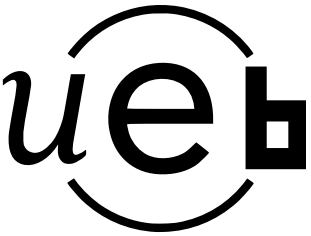
N° d'ordre : 4369 ANNÉE 2011



THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Romain Tavenard

préparée à l'unité de recherche 6074 : IRISA
Institut de Recherche en Informatique et Systèmes Aléatoires
Mathématiques-STIC

Indexation de séquences de descripteurs

Thèse soutenue à Rennes le 4 Juillet 2011

devant le jury composé de :

Marie-Odile Cordier
Professeure, Université de Rennes 1
Présidente

Philippe Joly
Professeur, Université Paul Sabatier
Rapporteur

Hervé Glotin
Professeur, Université Sud Toulon Var
Rapporteur

Stéphane Marchand-Maillet
Maître de conférences, Université de Genève
Examinateur

Patrick Gros
Directeur de recherche, INRIA
Directeur de thèse

Laurent Amsaleg
Chargé de recherche, CNRS
Co-directeur de thèse

À Aude et Camille.

Table des matières

Remerciements

Je tiens à remercier chaleureusement Philippe Joly et Hervé Glotin, d'une part, pour leur travail de rapporteur de ce manuscrit, et Marie-Odile Cordier et Stéphane Marchand-Maillet d'autre part pour avoir accepté les charges respectives de présidente du jury et d'examinateur.

Je remercie Laurent Amsaleg pour son soutien indéfectible tout au long des quelques années de travail que clôturent ce document. Les nombreux échanges que nous avons pu avoir autour de ces travaux ont constitué pour moi les meilleurs moments de cette thèse et j'espère avoir la chance de continuer à travailler avec des personnes aussi passionnées et intéressantes que lui. J'ai également une pensée particulière pour Guillaume Gravier et Patrick Gros dont les conseils avisés se sont avérés précieux pour mener à bien ce travail.

Je ne saurais remercier suffisamment Hervé Jégou, pour son énergie incroyable et ses conseils toujours judicieux, mais surtout pour son amitié solide.

Merci également à Vincent Claveau avec qui j'ai toujours pris un grand plaisir à discuter de sujets aussi divers qu'intéressants. Je remercie les autres membres de l'équipe Texmex et notamment Sébastien Campion pour son expertise précieuse, Loïc Lesage pour son soutien aussi discret qu'efficace et Camille Guinaudeau et Gwénolé Lecorvé pour les discussions que j'ai eu le plaisir d'avoir avec eux. Je remercie enfin Mathieu Lagrange dont la rencontre s'est avérée aussi inattendue que fructueuse.

Je ne remercierai jamais assez Aude sans qui ces travaux n'auraient probablement pas eu la même saveur et Camille, qui, s'il ne devrait pas lire ces lignes de sitôt, m'a beaucoup apporté depuis son arrivée. Je tiens également à remercier mes parents pour leur soutien sans faille à mes projets depuis de nombreuses années.

Je ne peux terminer ces remerciements sans avoir une pensée particulière pour Simon (pour la guitare et les hommages répétés à Fabrice), Kévin et Sam Oun (pour AdR, les makis et *les* filets mignons) et Alexandre (pour la batterie, surtout !). Je remercie aussi chaleureusement Thomas de me supporter depuis si longtemps et d'avoir pris la peine de venir assister à ma soutenance de thèse. Je pense enfin à Benoît qui m'a longtemps permis de troquer avantageusement les sandwich-cafet du mardi midi contre du bon temps tennistique où les chambrages étaient de rigueur.

Introduction

La production de documents implique la mise en œuvre de moyens pour les conserver afin de permettre un accès facilité aux informations qu'ils contiennent. La recherche d'information est donc une discipline qui a dû se développer dès les premiers temps de l'écriture. Toutefois, l'informatisation de la société, récente à cette échelle, a encouragé le développement de nouvelles méthodes de recherche d'information, sous l'effet combiné des nouvelles capacités de calcul offertes et de l'explosion de la quantité d'information à traiter.

Un *medium* symptomatique de cette évolution est l'image. En effet, jusqu'aux années 1990, les seuls systèmes de recherche d'information pour des bases de données d'image reposent sur une annotation manuelle du contenu des images. Seulement, cette annotation a le triple inconvénient d'être fastidieuse à mettre en œuvre, de ne pas permettre un passage à l'échelle et d'être subjective et donc de dépendre des annotateurs. Pour cette raison, des systèmes ont été développés pour, d'une part, décrire le contenu des images d'une manière qui soit à la fois pertinente pour l'application visée et objective et, d'autre part, permettre des recherches efficaces dans des bases d'images décrites.

Pour faire face à la problématique d'efficacité, il est nécessaire d'organiser le contenu des bases considérées afin de faciliter l'étape de recherche. Cette organisation est appelée *indexation*. Elle est indispensable lorsque l'on souhaite effectuer des recherches rapides dans de grandes bases, la recherche exhaustive n'étant pas envisageable pour des raisons de coût de calcul. Il est communément admis que, même si cette étape d'indexation est relativement coûteuse, elle permet de fortement réduire le temps de requête qui est le point critique du système.

En ce qui concerne la description, les images ayant un contenu riche en information, les descripteurs extraits pour les représenter se présentent la plupart du temps sous la forme de vecteurs numériques de grande dimension. On définit alors une métrique dans ces espaces qui, associée à la description utilisée, a pour but d'exprimer la similarité des contenus en termes de distance entre les vecteurs numériques extraits, la recherche d'information se traduisant alors par une recherche de plus proches voisins dans un espace vectoriel. Or, il a été énoncé, sous le terme de *malédiction de la dimension*, que la recherche de vecteurs de grande dimension introduit des problématiques particulières qu'il est important de considérer lorsque l'on imagine un système de recherche. Notre but n'est pas ici de proposer une liste exhaustive des effets néfastes de la malédiction de la dimension sur l'indexation, mais plutôt de souligner l'importance de sa prise en compte lorsque l'on souhaite imaginer un système de recherche de plus proches voisins en grande dimension.

Au-delà de l'exemple des images, se développe aujourd'hui la diffusion de doc-

uments multimédias temporels. Qu'il s'agisse d'enregistrements radiodiffusés, de vidéos auto-produites comme celles que l'on peut retrouver sur des plateformes comme YouTube ou Dailymotion, ou encore de musique, les volumes de contenus générés sont immenses et les besoins en outils de recherche également. Jusqu'alors, la plupart des approches proposées ont consisté en la recherche d'images ou de trames audio négligeant l'aspect temporel. Les deux systèmes commerciaux de référence d'identification de chansons [Haitsma 01, Wang 06], décrits plus en détail au chapitre suivant, fonctionnent, par exemple, sur ce principe, tout comme le système *Video Google* de recherche d'objets dans les vidéos [Sivic 03]. Cela s'explique par des raisons tout d'abord historiques. En effet, au moment où ont été développées ces techniques, des méthodes de recherche de plus proches voisins dans les espaces vectoriels, efficaces à grande échelle, commençaient à voir le jour et c'est tout naturellement que les premières propositions de la recherche de documents temporels les ont dérivé. D'autre travaux, souhaitant permettre une modélisation plus fine de l'enchaînement temporel des descripteurs, ont proposé l'utilisation de métriques adaptées, notamment basées sur l'alignement dynamique. Il faut toutefois noter que ces métriques ont une complexité de calcul qui rend difficile leur utilisation à grande échelle.

Nous proposons dans ce document des méthodes permettant de profiter de telles métriques pour des applications de recherche de séquences dans de grandes bases ainsi qu'une analyse critique de l'opportunité de leur utilisation en fonction de la nature des données considérées.

Le premier chapitre décrit les principaux résultats de l'état-de-l'art de la recherche de séquences. Les méthodes existantes sont classées en trois grandes catégories en fonction de la nature des données qu'elles considèrent. Nous nous intéressons dans un premier temps aux méthodes d'alignement dynamique qui, si elles proviennent du domaine de la reconnaissance de la parole, sont aujourd'hui largement utilisées pour la recherche de séquences mono-dimensionnelles telles que des cours de bourse, par exemple. Nous abordons ensuite les problématiques liées à la considération de séquences symboliques, pour lesquelles il existe une bibliographie importante notamment issue du domaine de la génomique. Enfin, nous nous focalisons sur les séquences multimédias et présentons quelques unes des principales méthodes de recherche développées dans ce cadre.

Les deux chapitres suivants proposent, dans des domaines où il a été montré que l'alignement dynamique est un outil approprié à la recherche de séquences, deux approches permettant d'interroger efficacement des bases de séquences. Les différentes propositions introduites dans ces chapitres visent à l'interrogation efficace de bases de données de séquences de vecteurs numériques de grande dimension.

Plus précisément, le deuxième chapitre s'intéresse tout particulièrement aux méthodes d'indexation à grande échelle utilisant un algorithme d'alignement dynamique appelé *Dynamic Time Warping* comme métrique de référence. L'accent est ainsi mis sur une forte prise en compte du caractère temporel des données tout au long de la chaîne d'indexation. Dans ce cadre, deux propositions centrales sont effectuées. La première vise à introduire de l'approximatif dans la recherche dans le but d'accélérer significativement les requêtes. La seconde a pour but d'utiliser des algorithmes adaptés aux données dans l'étape d'indexation pour permettre une organisation plus équilibrée des séquences au sein d'un arbre d'indexation.

Le troisième chapitre de ce document se concentre sur le cas particulier de la recherche de sous-séquences, à travers une application de recherche de reprises musicales. Nous nous intéressons ici à une approche fonctionnant en deux étapes, la première n'intégrant qu'une information temporelle très locale alors que la deuxième utilise une analyse plus fine pour les séquences les plus susceptibles de correspondre à la requête.

Le quatrième chapitre de ce document propose une discussion sur l'utilité de la mise en œuvre de métriques de comparaison fines (et donc coûteuses) pour les séquences dans divers cadres applicatifs. Pour ce faire, des expériences sont conduites qui comparent trois choix possibles de considération de l'axe du temps, allant d'une absence totale de prise en compte à une modélisation fine. Partant de ces observations, nous proposons une grille d'analyse de l'opportunité d'utiliser l'une ou l'autre de ces approches en fonction de la nature des données considérées.

Enfin, nous concluons sur les contributions de cette thèse et ses perspectives dans le dernier chapitre.

Contribution annexe

Au-delà des travaux présentés ici, nous avons également eu l'occasion de proposer une famille de métriques de comparaison fines des séquences qui ne sont pas abordées dans ce manuscrit car n'étant pas adaptées à l'indexation à grande échelle. Ces métriques reposent sur l'utilisation de modèles de régression en mesurant la similarité entre deux séquences dans l'espace des modèles. Ces travaux sont plus longuement détaillés dans [Tavenard 09].

Bibliographie

[Haitsma 01] J. Haitsma, T. Kalker & J. Oostveen. *Robust Audio Hashing for Content Identification.* In Proceedings of the International Workshop on Content-Based Multimedia Indexing, 2001.

[Sivic 03] J. Sivic & A. Zisserman. *Video Google : a Text Retrieval Approach to Object Matching in Videos.* In Proceedings of the International Conference on Computer Vision, volume 2, pages 1470–1477, 2003.

[Tavenard 09] R. Tavenard, L. Amsaleg & G. Gravier. *Model-Based Similarity Estimation of Multidimensional Temporal Sequences.* Annals of Telecommunications, vol. 64, pages 381–390, 2009.

[Wang 06] A. Wang. *The Shazam music recognition service.* Communications of the ACM, vol. 49, pages 44–48, 2006.

Chapitre 1

État-de-l'art

Ce chapitre décrit les travaux fondateurs liés à l'interrogation de grandes bases de données de séquences. Cette thématique a été abordée sous différents angles par les communautés scientifiques de la recherche d'information et du multimédia. Ces différentes visions structurent la suite de ce chapitre. Après avoir présenté une typologie des recherches, nous introduirons trois grandes familles de méthodes proposées dans ce cadre. Nous commencerons par aborder les approches par alignement dynamique et les optimisations proposées pour les adapter au traitement de grands volumes de données. Nous nous intéresserons ensuite aux méthodes adaptées aux séquences symboliques issues de grands domaines de recherche comme la génomique ou la recherche textuelle. Enfin, nous présenterons les principaux travaux spécifiques aux séquences multimédias qui nous concernent particulièrement ici.

Les principales notations utilisées dans la suite de ce document sont récapitulées

Notation	**Description**
dim	Dimension du $\mathbb{R}$-espace vectoriel dans lequel sont définis les éléments
d	Distance dans l'espace $\mathbb{R}^{\text{dim}}$
$Q = q_1 q_2 \dots q_m$	Séquence requête, supposée de taille m
$C = c_1 c_2 \dots c_l$	Séquence de la base, supposée de taille l
$S_1.S_2$	Séquence issue de la concaténation des séquences S_1 et S_2
$\text{NN}_k(o, E)$	Ensemble des k-plus proches voisins d'un objet o (qui peut être une séquence ou un élément) parmi les éléments d'un ensemble E
$[\![1; n]\!]$	Ensemble des entiers compris entre 1 et n inclus
$x \wedge y$	Conjonction entre les variables booléennes x et y
$x \vee y$	Disjonction entre les variables booléennes x et y

Tableau 1.1 – *Récapitulatif des notations utilisées.*

dans le tableau 1.1.

1.1 Typologie des recherches

Les problèmes adressés dans ce manuscrit s'inscrivent dans le cadre de la recherche d'information dans de grandes bases de données de séquences. De manière générale, on appellera séquence une suite d'éléments. Un élément sera soit symbolique, dans ce cas, on précisera l'alphabet correspondant, soit numérique et alors nous considèrerons qu'il s'agit d'un vecteur de $\mathbb{R}^{\text{dim}}$. Il est à noter que, dans le cas des séquences multimédias qui nous intéresse ici, il est fréquent de considérer des alphabets de grandes tailles, pouvant aller de plusieurs dizaines de milliers à plusieurs millions de symboles [Sivic 03], pour les séquences symboliques. De même, les éléments des séquences numériques que nous considérons sont susceptibles d'être de grande dimension (les descripteurs d'images globaux les plus largement utilisés aujourd'hui ont des dimensions de l'ordre de la dizaine de milliers).

La recherche d'information dans ces séquences peut se décliner sous plusieurs formes.

On peut tout d'abord distinguer les problématiques de *détection* d'éléments répétés et de *recherche*. Dans le premier cas, on dispose d'une longue séquence (ou d'un ensemble de séquences) dans lequel on souhaite retrouver des sous-séquences qui se répètent. Cette approche est notamment utile lorsque l'on souhaite découvrir la structure d'un document. Il est en effet fréquent qu'un flux multimédia soit structuré par l'intermédiaire d'éléments répétitifs, que ce soit dans le cas de chansons organisées en couplets et refrains ou pour les flux de radio ou de télévision qui sont ponctués de publicités et autres génériques. De plus, il existe dans ces séquences des répétitions plus subtiles qui peuvent également être des sources d'information pertinentes, pour une organisation thématique du flux. Si l'on considère par exemple un signal de parole, il peut être intéressant de rechercher les répétitions de certains mots-clé dans le but de mettre en relation des parties de discours traitant de thématiques similaires. Dans le deuxième cas, on souhaite rechercher les occurrences d'une séquence requête dans une base, la taille de la requête étant très petite devant celle de la base.

Dans ce cas, on distingue deux problématiques. On peut vouloir retrouver les k séquences de la base les plus similaires à la requête : on parle alors de recherche de *k-plus proches voisins*. On peut également chercher à retrouver l'ensemble des séquences de la base dont la distance à la requête est plus petite qu'un seuil fixé ε et l'on parle alors de *recherche à ε près*.

La base considérée peut, selon les cas, exister sous des formes plus ou moins organisées. En absence totale de segmentation, la base considérée sera vue comme un flux et le but d'une recherche sera alors de trouver les positions dans le flux où apparaît la requête. Pour pouvoir appliquer certains algorithmes, il peut toutefois être utile de segmenter arbitrairement ledit flux et ainsi chercher les segments de la base qui correspondent à la requête. Dans le cas de séquences multimédias, il peut exister une organisation sous-jacente (flux sonore divisé en chansons, flux vidéo découpé en scènes, par exemple) et, si l'on a accès à ces informations, il peut être judicieux de les utiliser afin d'obtenir une segmentation plus sémantique.

De manière générale, il est fréquent de rechercher des correspondances incomplètes. Si l'on considère par exemple le cas de l'utilisation d'un logiciel comme Shazam [Wang 06], la requête est un sous-ensemble bruité de la chanson recherchée. Dans le cas d'une base segmentée arbitrairement, si les segments considérés sont de petite taille, il est possible d'avoir des requêtes qui seront des *sur-séquences* des segments leur correspondant dans la base. De même, le *recalage* (c'est-à-dire l'alignement entre les instants de début et de fin des séquences à comparer) entre la requête et ses correspondances dans la base est souvent nécessaire, dans le cas où la segmentation considérée est arbitraire ou peu fiable.

Pour permettre la recherche dans des bases de séquences, trois grandes familles d'approches peuvent être distinguées. Tout d'abord, il est possible d'élaborer des métriques de similarité adaptées aux données séquentielles et de construire des systèmes de recherche autour de ces mesures de similarité. Ensuite, on peut chercher à utiliser la redondance temporelle des données sans pour autant conserver l'ordre des vecteurs dans la séquence. On représentera alors une séquence comme un ensemble de vecteurs non ordonnés. Une dernière solution consiste à encapsuler la dimension temporelle des données dans leur description. Par exemple, de nombreux travaux proposent de représenter les séries temporelles par un point dans le domaine fréquentiel, notamment à l'aide de transformées de Fourier par exemple [Faloutsos 94]. Une fois ce changement de représentation effectué, on se ramène à un problème classique de recherche de plus proches voisins dans un $\mathbb{R}$-espace vectoriel. Il n'est alors pas utile de développer des méthodologies de recherche propres et c'est pour cela que nous n'aborderons pas ces techniques dans ce document.

1.2 Alignement dynamique

Un moyen communément utilisé pour comparer deux séquences numériques est de chercher l'ensemble de déformations de coût minimal nécessaire à l'alignement d'une séquence sur l'autre. Les déformations locales autorisées (ou opérations d'édition) sont l'insertion ou la suppression d'un élément dans une séquence et le remplacement d'un élément d'une séquence par un autre. Celles-ci définissent un modèle de déformations locales et nous verrons plus loin des moyens de l'adapter à un cadre applicatif spécifique.

Nous présentons dans cette section un algorithme classique, appelé alignement dynamique temporel, ou *Dynamic Time Warping* (DTW), qui permet d'obtenir efficacement l'ensemble de déformations de coût minimal, ainsi que différentes optimisations envisageables pour adapter cet algorithme à des besoins spécifiques.

1.2.1 Présentation

Sakoe et Chiba [Sakoe 78] posent le problème du calcul de l'ensemble de déformations de coût minimal entre deux séquences. Le coût correspondant est une mesure de la dissimilarité entre les séquences considérées.

Pour évaluer la similarité entre Q et C, on commence par construire une matrice

S de correspondance entre les deux séquences, de taille $m \times l$, que l'on définit comme :

$$\forall (i,j) \in [\![1;m]\!] \times [\![1;l]\!], S_{i,j} = d(q_i, c_j). \tag{1.1}$$

On définit un *chemin* $W = w_1 \cdots w_K$ comme une suite de paires $w_k = (i,j) \in [\![1;m]\!] \times [\![1;l]\!]$ d'indices de S vérifiant les contraintes aux bords :

$$w_1 = (1,1), w_K = (m,l). \tag{1.2}$$

Dans le modèle de déformations locales évoqué plus haut, un chemin est dit *continu* s'il vérifie, pour tout $k \in [\![1;K-1]\!]$:

$$(w_k = (i,j)) \wedge (w_{k+1} = (i',j')) \Rightarrow (|i-i'| \leq 1) \wedge (|j-j'| \leq 1). \tag{1.3}$$

On s'intéresse également à la propriété de *monotonie* qui s'énonce comme suit :

$$\forall k \in [\![1;K-1]\!], (w_k = (i,j)) \wedge (w_{k+1} = (i',j')) \Rightarrow (i' \geq i) \wedge (j' \geq j). \tag{1.4}$$

On associe à chaque chemin un coût :

$$c(W) = \left(\sum_{k=1}^{K} S_{w_k} \right) \tag{1.5}$$

et l'on appelle *chemin optimal* le chemin continu monotone de coût minimal :

$$\hat{W} = \underset{W}{\operatorname{argmin}} \left(c(W) \right). \tag{1.6}$$

La DTW est alors définie comme le coût du chemin optimal :

$$\mathrm{DTW}(Q,C) = c(\hat{W}) = \min_{W} \left(c(W) \right). \tag{1.7}$$

On appelle *transition* un ensemble de deux paires d'indices consécutifs au sein d'un chemin. Dans la suite de ce document, on fera référence aux transitions via le vecteur de déplacement qui leur est associé. On peut noter que les propriétés de continuité et de monotonie ci-dessus énoncées limitent les chemins acceptables à n'utiliser que des transitions horizontales (notée $(1,0)$), verticales (notée $(0,1)$) et diagonales (notée $(1,1)$). Les transitions $(0,0)$, si elles ne sont pas interdites, ne seront pas considérées car elles ne peuvent que pénaliser le chemin en ajoutant à son coût un certain nombre de fois la même valeur positive $S_{i,j}$.

Il en découle la formule de récurrence suivante :

$$\forall (i,j) \in [\![1;m]\!] \times [\![1;l]\!], \gamma_{i,j} = \min \begin{cases} S_{i,j} + \gamma_{i-1,j-1} \\ S_{i,j} + \gamma_{i-1,j} \\ S_{i,j} + \gamma_{i,j-1} \end{cases}, \tag{1.8}$$

où $\gamma_{i,j}$ représente le coût du chemin optimal associé à la sous-matrice de S constituée de ses i premières lignes et j premières colonnes ou, en d'autres termes, le coût de l'alignement des séquences $q_1 \ldots q_i$ et $q_1 \ldots q_j$. Pour respecter la contrainte au bord $w_1 = (1,1)$, on fixe :

$$\gamma_{0,0} = 0, \tag{1.9}$$

$$\forall j \in [\![1;l]\!], \gamma_{0,j} = +\infty, \tag{1.10}$$

$$\forall i \in [\![1;m]\!], \gamma_{i,0} = +\infty. \tag{1.11}$$

$\gamma_{m,l}$ fournit ainsi une mesure de la dissimilarité entre les séquences Q et C.

Pour éviter de calculer plusieurs fois chaque terme d'indice (i, j), Sakoe et Chiba [Sakoe 78] proposent d'utiliser la programmation dynamique. On construit ainsi une matrice Γ dans laquelle on stocke les $\gamma_{i,j}$ au fur et à mesure des calculs. L'algorithme 1.1 illustre le remplissage de cette matrice ligne par ligne et de la gauche vers la droite (de cette manière, on est sûr, lorsque l'on considère le terme d'indice (i, j), d'avoir pré-calculé les termes d'indices $(i-1, j)$, $(i, j-1)$ et $(i-1, j-1)$ nécessaires au calcul du terme courant).

Algorithme 1.1 *Calcul de DTW.* Cet algorithme retourne le coût du chemin optimal étant donnée la matrice S de similarité entre les séquences considérées.

Entrées : S[1..m][1..l]
  float Γ[0..m][0..l]
  int i, j
  Γ[0][0] ← 0
  Γ[0][1..l] ← $+\infty$
  Γ[1..m][0] ← $+\infty$
  pour i = 1 à m **faire**
    pour j = 1 à l **faire**
      Γ[i][j] ← $S[i][j]$ + min(Γ[i-1][j], Γ[i][j-1], Γ[i-1][j-1])
    fin pour
  fin pour
  renvoie Γ[m][l]

Cet algorithme a une complexité temporelle en $O(m \times l)$ et une complexité spatiale en $O(m \times l)$. Cette dernière peut toutefois aisément être réduite à $O(l)$ car il suffit, lorsque l'on traite la ligne d'indice i, de stocker les lignes d'indices $i-1$ et i de la matrice Γ, les précédentes n'étant plus utiles. De la même manière, si l'on a $m \ll l$, il sera judicieux de remplir la matrice Γ colonne par colonne et limiter la complexité spatiale à $O(m)$.

1.2.2 Variantes

La suite de cette section décrit des modifications fréquemment appliquées à l'algorithme décrit plus haut dans le but d'adapter le modèle de déformations utilisé.

Pondérations

On peut tout d'abord remarquer que la formule de récurrence définie par l'équation 1.8 a tendance à favoriser les alignements contenant de nombreuses transitions diagonales dans la mesure où celles-ci rapprochent plus de l'objectif – qui est l'élément d'indice (m, l) de la matrice Γ – qu'une simple transition horizontale ou verticale. Pour influer sur l'importance relative accordée aux diverses transitions, il est possible de les pondérer, pour obtenir une formule de récurrence de la forme :

$$\gamma_{i,j} = \min \begin{cases} \alpha_{(1,1)} S_{i,j} + \gamma_{i-1,j-1} \\ \alpha_{(0,1)} S_{i,j} + \gamma_{i,j-1} \\ \alpha_{(1,0)} S_{i,j} + \gamma_{i-1,j} \end{cases} . \tag{1.12}$$

Sakoe et Chiba [Sakoe 78] proposent, par exemple, d'utiliser le jeu de pondérations

$$\alpha_{(0,1)} = \alpha_{(1,0)} = \frac{\alpha_{(1,1)}}{2}, \tag{1.13}$$

effaçant ainsi la préférence naturellement accordée aux chemins diagonaux. Il faut toutefois noter que l'utilisation de ce jeu de pondérations pénalise fortement les transitions diagonales, ce qui, pour les applications où les déformations locales sont faibles (telles que la recherche de quasi-réplicat, comme nous le verrons au dernier chapiter de ce manuscrit), est peu réaliste.

Transitions

Un autre moyen d'influer sur l'allure générale du chemin obtenu est de modifier l'ensemble des transitions autorisées. Comme illustré en figure 1.1a, l'ensemble de transitions autorisées $\{(0,1),(1,0),(1,1)\}$ découlant des propriétés de continuité et de monotonie énoncées plus haut permet en effet de considérer des chemins assez éloignés de la diagonale. Or, dans certains cas, la nature des documents considérés nous permet d'écarter de telles hypothèses. Par exemple, dans le cas de documents multimédias, il est fréquemment admis que les vitesses des séquences considérées sont similaires. Cela implique que les chemins trop éloignés de la diagonale, qui impliquent des accélérations et décélérations importantes, sont à exclure. Sakoe et Chiba [Sakoe 78] proposent de modifier la contrainte de continuité pour confiner les chemins considérés autour de la diagonale en forçant chaque transition horizontale ou verticale à être combinée à une transition diagonale, limitant alors les transitions autorisées à l'ensemble $\{(1,2),(2,1),(1,1)\}$, comme illustré en figure 1.1b. Cela mène à la redéfinition de la relation de récurrence :

$$\gamma_{i,j} = \min \begin{cases} \alpha_{(1,1)} S_{i,j} + \gamma_{i-1,j-1} \\ \alpha_{(2,1)} S_{i,j} + \gamma_{i-2,j-1} \\ \alpha_{(1,2)} S_{i,j} + \gamma_{i-1,j-2} \end{cases}. \tag{1.14}$$

Contraintes globales

Une autre modification consiste à appliquer une contrainte globale aux chemins admissibles. La contrainte globale la plus utilisée est une bande de largeur fixe $(2r+1)$ autour de la diagonale, appelée *bande de Sakoe-Chiba*. Cette contrainte équivaut à ne remplir la matrice Γ que pour les termes d'indices (i,j) tels que :

$$|i-j| \in I_R = [\![0; r]\!]. \tag{1.15}$$

La figure 1.2b illustre l'altération d'un chemin découlant de cette contrainte en fixant r à 1.

Une autre contrainte globale communément utilisée est le *parallélogramme d'Itakura*, qui consiste à restreindre les chemins dans un parallélogramme dont l'une des diagonales correspond au chemin diagonal. Cette contrainte globale peut être imposée par une modification des transitions autorisées. Par exemple, les transitions proposées dans l'équation 1.14 impliquent une contrainte globale similaire, comme

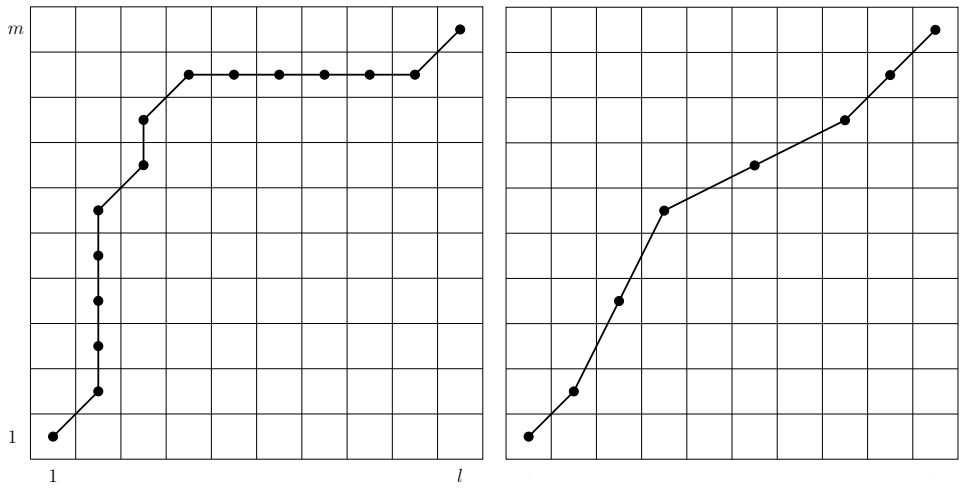


FIGURE 1.1 – *Alignement dynamique et transitions.* Deux exemples de chemins pour l'alignement dynamique : le premier correspond aux contraintes classiques telles qu'énoncées par l'équation 1.8, alors que le second correspond aux contraintes découlant de l'équation 1.14. Dans les deux cas, les points correspondent aux paires d'indices constitutives du chemin optimal retenu.

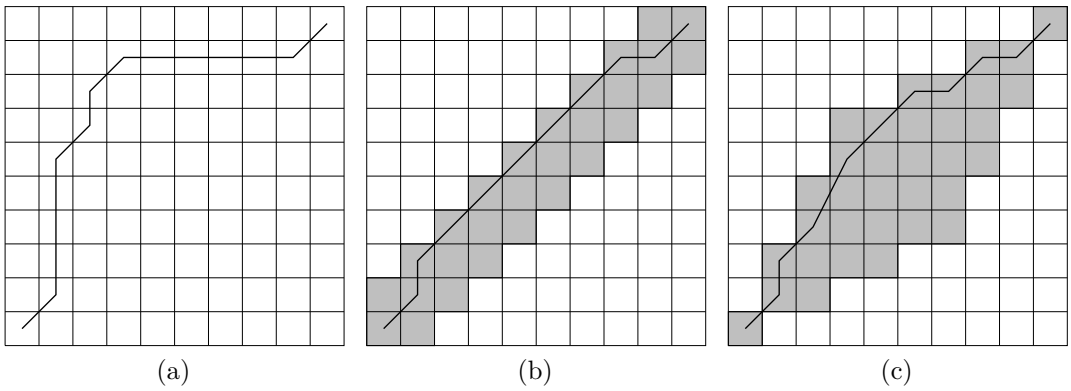


FIGURE 1.2 – *Alignement dynamique et chemins.* Trois exemples de chemins pour l'alignement dynamique : le premier correspond aux contraintes classiques telles qu'énoncées par l'équation 1.8, les deuxième et troisième correspondent à des chemins contraints globalement, respectivement par une bande de Sakoe-Chiba de largeur $r = 1$ et par un parallèlogramme d'Itakura.

illustré en figure 1.1b. On peut également noter que ce parallélogramme peut être vu comme une bande de largeur variable, r et I_R devenant dépendants de i et étant alors notés $r(i)$ et $I_R(i)$.

Dans les deux cas, l'application de telles contraintes globales a un impact positif sur la complexité temporelle qui devient $O\left(\max_i r(i) \times (m + l)\right)$ en économisant le calcul des termes éloignés de la diagonale.

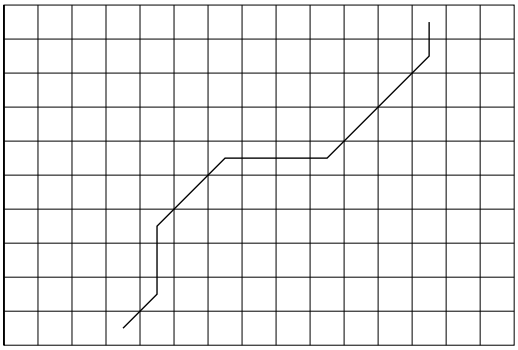


FIGURE 1.3 – *Alignement dynamique et sous-séquence.*

Sous-séquences

Enfin, si l'on souhaite rechercher l'alignement optimal dans le cas où Q est une sous-séquence de C, il convient de relaxer les contraintes aux bords pour que $w_1 = (1, j_1)$ et $w_K = (m, j_K)$ où j_1 et j_K sont dans l'intervalle $[\![1; l]\!]$. L'algorithme 1.1 doit alors être modifié en deux points :

- les éléments de la première ligne de Γ sont initialisés avec la valeur 0 ;
- la valeur de retour devient min(Γ[m][1..l]).

La complexité de cet algorithme est en $O(m \times n)$. Un exemple d'un tel chemin est présenté en figure 1.3.

Plus longue sous-séquence commune

Si l'on souhaite trouver la plus longue sous-séquence commune entre Q et C, il suffit d'adapter l'algorithme de DTW.

Tout d'abord, il convient de binariser la matrice S utilisée pour définir les éléments des séquences considérés comme similaires. Un moyen communément utilisé pour cela est de fixer un seuil T et de définir la matrice S^T comme :

$$\forall (i,j) \in [\![1;m]\!] \times [\![1;l]\!], S^T_{i,j} = (S_{i,j} < T)\,. \tag{1.16}$$

Reste alors à chercher la séquence d'éléments similaires la plus longue. Pour cela, il suffit de modifier la formule de récurrence de la manière suivante :

$$\forall (i,j) \in [\![1;m]\!] \times [\![1;l]\!], \gamma_{i,j} = \begin{cases} 1 + \max\left(\gamma_{i-1,j-1}, \gamma_{i,j-1}, \gamma_{i-1,j}\right) & \text{si } S^T_{i,j} \\ 0 & \text{sinon} \end{cases}, \tag{1.17}$$

la longueur de la plus longue sous-séquence commune étant alors égale au maximum des $\gamma_{i,j}$.

Il existe une variante de cet algorithme, appelée *plus longue sous-séquence commune pondérée* dans laquelle la matrice S^T n'est pas binaire mais numérique. Dans ce cas, les valeurs qui remplissent cette matrice ne sont pas des distances comme c'est le cas pour la DTW classique mais des mesures de similarités qui peuvent être négatives. La formule de récurrence alors utilisée est :

$$\gamma_{i,j} = \max\left(\gamma_{i-1,j-1} + S^T_{i,j}, \gamma_{i,j-1} + S^T_{i,j}, \gamma_{i-1,j} + S^T_{i,j}, 0\right). \tag{1.18}$$

Pour ces deux variantes de l'algorithme de plus longue sous-séquence commune, on obtient une complexité quadratique.

1.2.3 Recherche de séquences au sein d'une base

Nous présentons ici des adaptations de la DTW visant à diminuer son coût computationnel ou à limiter le nombre d'alignements nécessaires pour interroger une base de séquences. Nous considèrerons donc dans la suite une séquence requête que l'on souhaite comparer à un ensemble de séquences candidates.

Minoration de la DTW

Un moyen de limiter le nombre de calculs de DTW est d'en estimer une version approchée qui soit moins coûteuse et de ne calculer la valeur exacte de la dissimilarité que pour les candidats les plus proches de la requête au sens de cette version approchée. Si, en outre, cette version approchée est une borne inférieure de la DTW, elle peut être utilisée pour rejeter les séquences de la base trop dissimilaires à la requête. En effet, dans le cas d'une recherche à ε près dans une base, si on note LB la borne inférieure considérée, on a :

$$(\mathrm{LB}(Q, C) > \varepsilon) \Rightarrow (\mathrm{DTW}(Q, C) > \varepsilon)\,. \tag{1.19}$$

La DTW peut alors n'être calculée que sur les séquences de la base pour lesquelles $LB(Q, C) \leq \varepsilon$. De la même manière, il est possible d'utiliser une telle borne inférieure dans le cadre d'une recherche de k-plus proches voisins. Le principe sera alors, pour une requête, d'ordonner la liste des candidats par valeur de LB croissante et de calculer la DTW sur les premiers éléments de cette liste. Ainsi, si après un certain nombre k_0 (variable) de calculs exacts, on trouve k valeurs de DTW toutes plus petites que la $(k_0 + 1)^{\text{ième}}$ valeur de LB, on est assurés d'avoir les k-plus proches voisins exacts. Ce principe est illustré en figure 1.4.

Le but est alors de trouver une borne inférieure qui soit à la fois peu coûteuse en temps de calcul et proche de la DTW.

LB_Yi, LB_Keogh et LB_PAA La figure 1.5 illustre les bornes inférieures présentées ici. Il est intéressant de noter que, pour deux séquences à comparer fixées (et donc une valeur unique renvoyée par la DTW), on obtient des bornes inférieures assez variables. L'objectif étant de se rapprocher au plus près de la DTW, plus l'aire grisée est grande, meilleure est la borne inférieure considérée.

Yi, Jagadish et Faloutsos [Yi 98] proposent une borne inférieure, notée LB_Yi, qui s'intéresse aux extrema des deux séquences. La DTW entre deux séquences est ainsi approchée par la somme, pour chacune des deux séquences à considérer, des éléments supérieurs au maximum de l'autre séquence et des éléments inférieurs au minimum de l'autre séquence. Chacun des éléments de cette somme est un minorant d'un élément différent de la somme présentée en équation 1.7, ce qui montre que LB_Yi est bien une borne inférieure de la DTW.

Cette borne a été affinée par Keogh et Ratanamahatana [Keogh 05] dans le cas où l'on applique une contrainte globale au calcul de DTW. En utilisant une bande

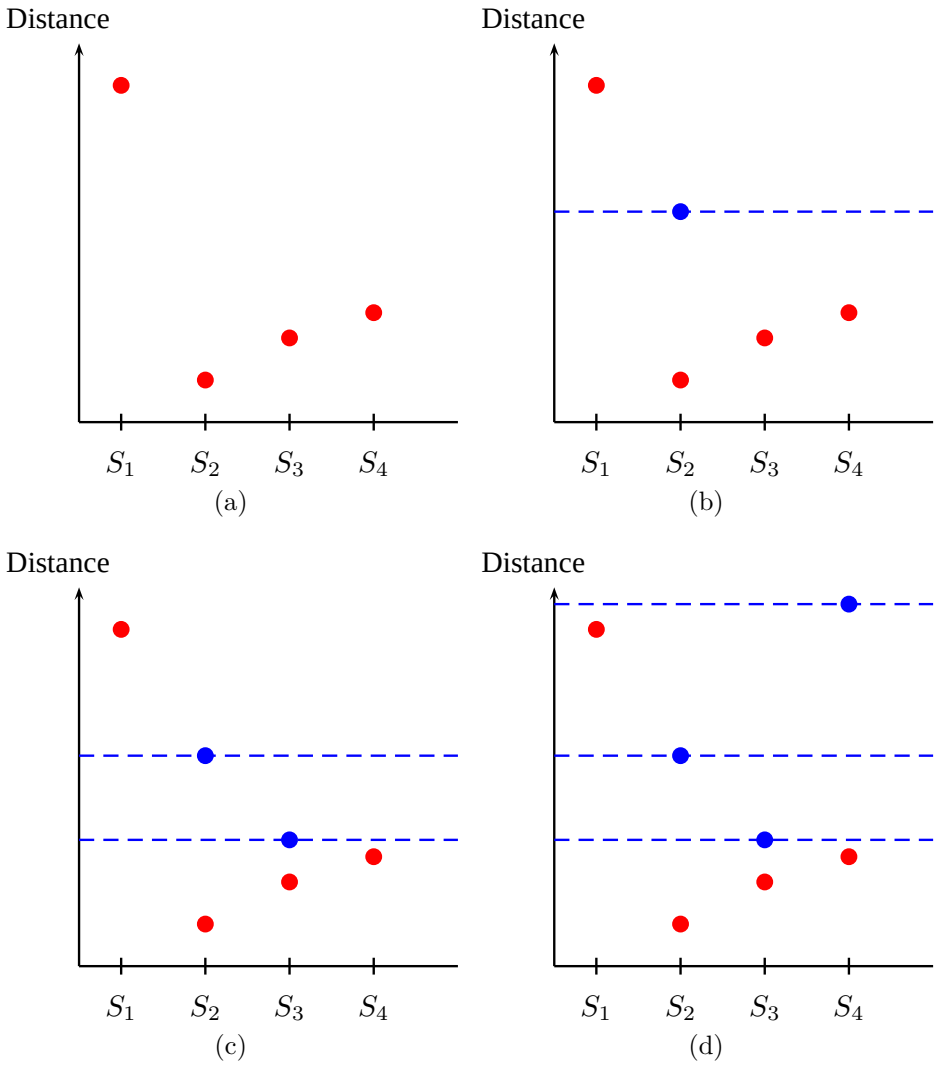
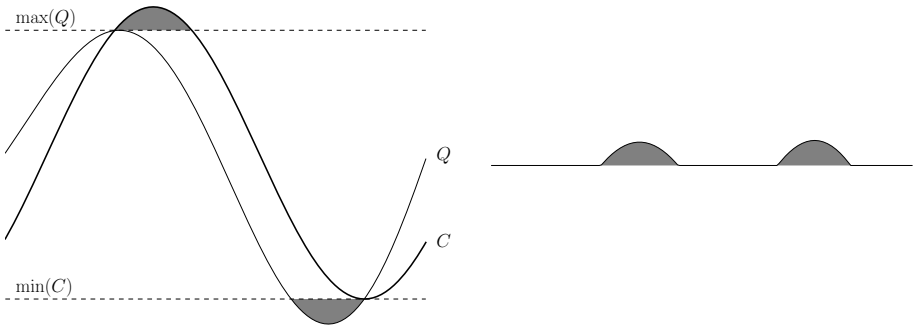
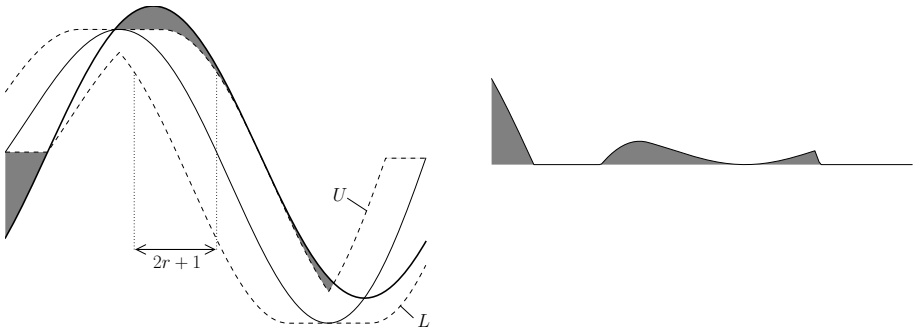


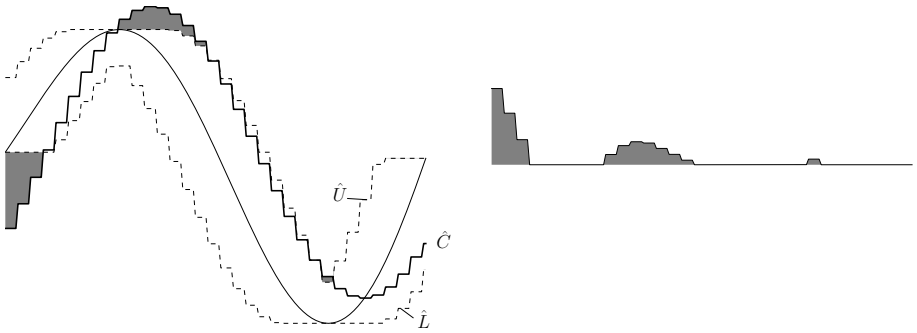
FIGURE 1.4 – *Utilisation des bornes inférieures.* On considère ici une séquence requête qui est comparée aux séquences S_1 à S_4 de la base et pour laquelle on recherche le plus proche voisin exact. On commence, comme illustré en figure (a), par calculer une borne inférieure à la DTW, ici représentée par un point rouge, pour chaque séquence de la base. On considère ensuite les séquences de la base par ordre croissant de borne inférieure, on calcule donc la DTW exacte pour la séquence S_2 (figure (b)). On remarque dès lors que la séquence S_1 ne peut pas correspondre à une valeur de DTW plus faible que la plus petite distance calculée (qui correspond ici, pour le moment, à S_2) car sa borne inférieure est elle-même plus grande que cette distance. On peut alors d'ores et déjà exclure S_1 de la suite de la recherche. Les calculs s'arrêtent lorsqu'aucune des séquences restantes n'a une borne inférieure plus petite que la plus petite distance exacte calculée. Dans notre cas, on a effectué 3 calculs exacts de DTW au lieu de 4.



(a) LB_Yi



(b) LB_Keogh



(c) LB_PAA

FIGURE 1.5 – *Bornes inférieures pour la DTW.* La requête est dessinée en trait fin et la séquence de la base à laquelle elle est comparée (ou sa version sous-échantillonnée dans le cas de LB_PAA) en trait gras. Dans chacun des cas, l'aire colorée en gris correspond à la mesure renvoyée par la borne inférieure considérée. Elle est reproduite sur les figures de droite pour plus de lisibilité. Pour LB_Keogh et LB_PAA, la contrainte globale considérée est une bande de Sakoe-Chiba, menant à une enveloppe de largeur fixe $2r + 1$ telle qu'indiquée en (b). On voit notamment ici l'intérêt de se restreindre à une bande autour de la diagonale pour LB_Keogh qui offre ainsi une mesure plus élevée que LB_Yi. En outre, les approximations introduites dans LB_PAA, si elles permettent de gagner en temps de calcul, font perdre de la précision dans l'estimation de la DTW. Cette figure est inspirée de [Keogh 05].

de Sakoe-Chiba suffisamment étroite, les minimum et maximum de

$$\{q_j, |i-j| \in I_R(i)\}, \tag{1.20}$$

qui est l'ensemble des éléments de Q susceptibles d'être mis en correspondance avec c_i, sont proches et alors l'alignement entre Q et C est bien estimé par l'écart entre C et l'enveloppe de Q formée par les séquences :

$$U : \forall i \in [\![1; m]\!], u_i = \max_{|i-j| \in I_R(i)} q_j, \tag{1.21}$$

$$L : \forall i \in [\![1; m]\!], l_i = \min_{|i-j| \in I_R(i)} q_j. \tag{1.22}$$

Pour pouvoir comparer les séquences C, U et L à l'aide d'une distance de Minkowski, celles-ci doivent être de même taille. Pour rendre cela possible, les auteurs proposent de ré-échantillonner la séquence C. Ainsi, dans la suite de cette partie, la séquence C sera supposée de longueur m. Il faut toutefois noter que les bornes inférieures proposées par la suite minorent la DTW appliquée non pas entre les séquences Q et C mais entre la séquence Q et la version de la séquence C ré-échantillonnée. Il vient alors [1] :

$$\text{LB\_Keogh}(Q, C) = \sum_{i=1}^{m} \Phi(c_i, u_i, l_i), \tag{1.23}$$

où, pour tout $y > z$, on a :

$$\Phi(x, y, z) = \begin{cases} d(x, y) & \text{si } x > y \\ d(x, z) & \text{si } x < z \\ 0 & \text{sinon} \end{cases}. \tag{1.24}$$

La démonstration de la minoration de la DTW (contrainte globalement) par LB_Keogh est similaire à celle évoquée pour LB_Yi.

Que ce soit pour LB_Yi ou pour LB_Keogh, le coût de calcul de la borne inférieure est linéaire, mais la seconde présente l'avantage de proposer, pour la DTW contrainte globalement, une estimation plus fine.

Pour diminuer encore le coût du calcul de cette borne inférieure, les auteurs proposent de sous-échantillonner les séquences à considérer en utilisant la méthode présentée dans [Keogh 01], appelée *Piecewise Aggregate Approximation* (PAA). Le principe est de transformer une séquence $S = s_1 \cdots s_m$ en $\hat{S} = \hat{s}_1 \cdots \hat{s}_M$ telle que :

$$\forall i \in [\![1; M]\!], \hat{s}_i = \text{moyenne}(c_{\frac{m}{M}(i-1)+1}, \cdots, c_{\frac{m}{M}i}). \tag{1.25}$$

Il en découle une mesure LB_PAA qui minore la norme L_2 entre les séquences mais qui ne fournit aucune garantie quant à la minoration de la DTW. On a :

$$\hat{U} : \forall i \in [\![1; M]\!], \hat{u}_i = \max(u_{\frac{m}{M}(i-1)+1}, \cdots, u_{\frac{m}{M}i}), \tag{1.26}$$

$$\hat{L} : \forall i \in [\![1; M]\!], \hat{l}_i = \min(l_{\frac{m}{M}(i-1)+1}, \cdots, l_{\frac{m}{M}i}), \tag{1.27}$$

$$\text{LB\_PAA}(Q, C) = \sum_{i=1}^{M} \frac{m}{M} \Phi(\hat{c}_i, \hat{u}_i, \hat{l}_i). \tag{1.28}$$

1. Il est à noter que la formule ici présentée varie légèrement de celle présentée dans [Keogh 05] qui définit LB_Keogh comme la racine carrée de la version ici proposée. Cette différence se justifie par une formule différente pour la DTW et n'implique pas de changement dans le raisonnement. La même modification est appliquée à LB_PAA, présentée plus tard.

***i*SAX** Shieh et Keogh [Shieh 08] ont proposé des travaux dans le domaine de la recherche de séquences numériques. La méthode proposée passe par une étape de symbolisation des données, qui utilise le schéma proposé par Lin, Keogh, Wei et Lonardi [Lin 07]. Celui-ci consiste en deux étapes : un sous-échantillonnage des séquences utilisant PAA, suivi d'une quantification des séquences obtenues. Les bornes des intervalles de quantification utilisées sont fixées de manière à assurer l'équiprobabilité des différents symboles sous une hypothèse de distribution normale centrée réduite. Ainsi, chaque séquence numérique peut être représentée par une séquence de symboles qui constitue l'entrée associée dans la structure d'index (qui est un fichier inversé, comme présenté en section 1.3.1).

Les auteurs remarquent qu'une telle utilisation mène à des déséquilibres très importants entre les populations associées aux entrées de l'index et proposent d'augmenter la granularité de la quantification pour certains symboles des entrées les plus peuplées. Autrement dit, au moment de peupler la structure de fichier inversé, si une entrée e dépasse la population maximale autorisée, on la scinde en deux en ajoutant un bit de précision à l'un des symboles de la séquence et on obtient alors deux nouvelles entrées dans le fichier inversé qui se partageront la population de l'entrée e, comme illustré en figure 1.6. Pour choisir le symbole qui sera représenté plus finement, les auteurs proposent de prendre soit le premier symbole si tous les symboles sont décrits avec le même nombre de bits, soit le premier symbole qui soit décrit avec moins de bits que son prédecesseur.

Ce choix a été affiné par Camerra *et al.* [Camerra 10], dans le cadre d'*i*SAX 2.0, qui se démarque de *i*SAX en deux points. Tout d'abord, les auteurs proposent un mécanisme de construction qui limite le nombre d'accès disque nécessaire lors de la construction de l'arbre. Ensuite, ils modifient la procédure de choix du symbole à raffiner lors de la scission d'une feuille. Leur algorithme commence par calculer la moyenne μ et l'écart-type σ de chaque élément de la représentation PAA pour l'ensemble des séquences stockées dans la feuille considérée. Ensuite, pour chaque élément de la représentation PAA, ils s'intéressent à la frontière qui serait induite par une représentation plus fine du symbole SAX qui leur est associé. Parmi les éléments dont les frontières sont comprises dans l'intervalle $[\mu - 3\sigma; \mu + 3\sigma]$ correspondant, ils choisissent de retenir l'élément dont la frontière est la plus proche de μ et raffinent le symbole SAX associé.

L'interrogation de la base se résume ensuite à une symbolisation de la séquence requête (pour être sûr d'avoir toutes les informations nécessaires, chaque symbole de la séquence requête obtenue doit avoir la précision de l'entrée de la base la plus précise), suivie d'un parcours de l'arbre d'indexation obtenu jusqu'à arriver à une feuille (ou à un ensemble de feuilles candidates). Le choix des feuilles se fait de manière à considérer tout d'abord celles pour lesquelles la mesure MINDIST est la plus petite. Cette dernière est une mesure de la similarité entre une séquence Q et un nœud représenté par son rectangle englobant $R = (B, H)$:

$$\mathrm{MINDIST}(Q, R) = \sqrt{\frac{n}{N} \sum_{i=1}^{N} \begin{cases} (\bar{q}_i - H_i)^2 & \text{si } \bar{q}_i > H_i \\ (B_i - \bar{q}_i)^2 & \text{si } \bar{q}_i < B_i \\ \min(H_i - \bar{q}_i, \bar{q}_i - B_i)^2 & \text{sinon} \end{cases}}. \tag{1.29}$$

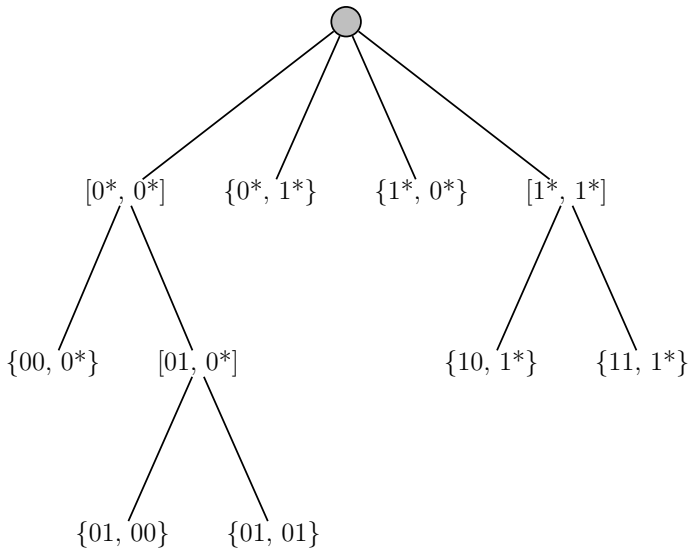


FIGURE 1.6 – *Exemple d'arbre d'indexation produit par iSAX.* Les nœuds de l'arbre sont ici représentés avec des crochets et les feuilles avec des accolades. On remarque que n'importe quelle séquence de taille 2 codée sur 2 bits a une feuille associée dans cet arbre (le symbole * remplace indifféremment 0 ou 1) et que, pour une feuille donnée, la précision n'est pas forcément la même pour chacun des deux symboles, comme dans le cas de la feuille {00, 0*}. Cette figure est inspirée de [Shieh 08].

Il est à noter que MINDIST, contrairement aux mesures présentées plus haut, n'est pas une borne inférieure pour la DTW, mais pour la distance euclidienne.

Enfin, la DTW est calculée entre la séquence numérique requête et chacune des séquences stockées dans la ou les feuilles considérées.

*i*SAX présente un intérêt certain par rapport aux approches par minoration présentées plus haut puisque l'utilisation d'une structure d'indexation permet de considérer des recherches dans des volumes de données très importants.

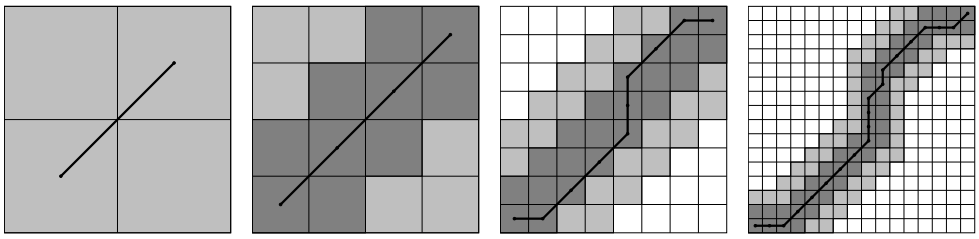
DTW et inégalité triangulaire Il est à noter que la DTW telle que présentée jusqu'ici n'est pas une distance car elle ne vérifie pas l'inégalité triangulaire. Chen et Ng [Chen 04] proposent de modifier légèrement la définition de la DTW pour faire disparaître cette limitation dans le but d'utiliser la borne inférieure fournie par l'inégalité triangulaire.

La formule de récurrence alors utilisée est la suivante :

$$\gamma_{i,j} = \min \begin{cases} S_{i,j} + \gamma_{i-1,j-1} \\ d(q_i, e_0) + \gamma_{i-1,j} \\ d(c_j, e_0) + \gamma_{i,j-1} \end{cases} . \tag{1.30}$$

Ainsi, le coût des transitions diagonales n'est pas modifié alors que le coût d'une transition horizontale ou verticale ne dépend plus de $S_{i,j}$ mais de la distance entre l'élément à insérer et un élément neutre e_0.

Les auteurs utilisent ensuite l'inégalité triangulaire pour fournir une borne inférieure à la DTW. En effet, pour toutes séquences C_1 et C_2 que l'on souhaite



(a) 1/8 (b) 1/4 (c) 1/2 (d) 1/1

FIGURE 1.7 – *DTW multi-échelle.* On voit sur cette figure 4 itérations de l'algorithme *Fast DTW*. Les cellules colorées en gris foncé correspondent aux cellules que le chemin à l'échelle précédente implique d'examiner. En pratique, les auteurs recommandent de s'accorder une marge de manœuvre par rapport à ces cellules pour ne pas manquer un chemin. Les cellules ainsi rajoutées sont représentées en gris clair sur les figures. Cette figure est issue de [Salvador 04].

comparer à une requête Q, on a :

$$|\mathrm{DTW}(Q, C_2) - \mathrm{DTW}(C_1, C_2)| \leq \mathrm{DTW}(Q, C_1). \tag{1.31}$$

Or, $\mathrm{DTW}(C_1, C_2)$ peut être calculé hors-ligne. Ainsi, en supposant que $\mathrm{DTW}(Q, C_2)$ a déjà été évalué, on obtient une borne inférieure peu coûteuse à la mesure $\mathrm{DTW}(Q, C_1)$ et l'on peut utiliser les mêmes raisonnements que ceux présentés plus haut pour limiter le nombre de calculs exacts de DTW.

Fast DTW

Salvador et Chan [Salvador 04] proposent une méthode d'analyse multi-échelle, appelée *Fast DTW* pour l'alignement dynamique. Le principe est ici de diminuer la complexité d'un calcul de DTW en commençant par chercher le chemin optimal pour des versions sous-échantillonnées des séquences Q et C, puis en raffinant itérativement ce chemin au fur et à mesure que l'on utilise des versions moins grossièrement échantillonnées. Ainsi, comme le montre la figure 1.7, une fois le chemin optimal $W_{1/i}$ calculé pour la résolution $1/i$, on se restreint, lorsque l'on double la résolution, aux chemins cohérents avec $W_{1/i}$. Si cette méthode ne garantit pas de retourner le chemin optimal (il est en effet possible que le chemin optimal à la résolution la plus fine ne soit pas entièrement compris dans les chemins de résolutions plus grossières), elle permet d'obtenir une approximation assez fine de la DTW pour une complexité en $O(m + l)$.

1.3 Séquences symboliques

Dans le cas particulier des séquences symboliques, des méthodes de recherche efficace spécifiques ont été développées qui utilisent la finitude de l'ensemble des symboles possibles.

Dans la suite, nous considérons des séquences de symboles définies sur l'alphabet fini $\Sigma = \{\sigma_1, \cdots, \sigma_t\}$.

1.3.1 n-grammes

Un *n-gramme* est une séquence constituée de n symboles. Le dictionnaire de l'ensemble des n-grammes possibles en utilisant un alphabet de taille t est ainsi de taille t^n. On peut alors chercher à représenter une séquence requête par l'ensemble des n-grammes qu'elle contient et effectuer des recherches isolées de chacun de ces n-grammes dont les résultats seront ensuite fusionnés [Puglisi 06].

Un moyen efficace de rechercher des mots issus d'un dictionnaire fini dans une base est d'utiliser un *fichier inversé*. La notion de fichier inversé a été introduite dans le domaine de la recherche de documents textuels. Le principe est de stocker, pour chaque mot du dictionnaire utilisé, l'ensemble des positions dans les textes de la base où le mot existe. La liste de ces positions est appelée *liste inversée*. Pour permettre des recherches efficaces, on utilise une table de hachage dans laquelle on stocke chaque mot du dictionnaire et un pointeur vers sa liste inversée associée.

Pour interroger une base ainsi indexée avec une requête Q de taille m, on a deux cas de figure. Si m est plus petit que n, on génère tout d'abord l'ensemble des mots de taille n s'écrivant sous la forme $Q.S$ où S est une chaîne de taille $n-m$ (il en existe t^{n-m}). Le résultat de la requête sera alors l'union des listes inversées associées à chacun de ces mots. Si, au contraire, on a $m \geq n$, on commence par générer l'ensemble des n-grammes contenus dans Q (il en existe $m-n+1$) et le résultat de la requête sera l'intersection des listes inversées associées à chacun de ces mots.

La principale faiblesse de cette approche est sa rigité, dans la mesure où aucune déformation temporelle n'est supportée.

1.3.2 Génomique

De par l'intérêt de la recherche de sous-chaînes communes entre séquences, que ce soit pour l'ADN ou pour les séquences d'acides aminés, et le volume de données à traiter, de nombreuses méthodes de recherche efficace ont été développées dans le domaine de la génomique. Les arbres et tableaux de suffixes permettent la recherche exacte de chaînes au sein d'une base organisée en flux. L'approche BLAST utilise ces outils dans le cadre de la recherche de sous-chaînes communes.

Arbres de suffixes

Les *arbres de suffixes* [Weiner 73, McCreight 76, Ukkonen 95] permettent la recherche de séquences symboliques au sein de grandes bases.

On considère ici un alphabet $\Sigma = \{\sigma_1, \cdots, \sigma_t\}$ enrichi d'un élément, noté \$, représentant la fin de chaîne. Ainsi, les chaînes Q et C deviennent respectivement $Q.\$$ et $C.\$$. En pratique, la séquence C représentant la base vue comme un flux, sa taille sera grande devant celle de la requête.

À partir de C, on construit un arbre de suffixes $\mathcal{A}_C$ tel que :
- les arcs de $\mathcal{A}_C$ sont des chaînes non-vides définies sur $\Sigma \cup \{\$\}$;
- chaque nœud intérieur de $\mathcal{A}_C$ (sauf la racine dans le cas particulier où C est de taille au plus 1) a au moins deux fils ;

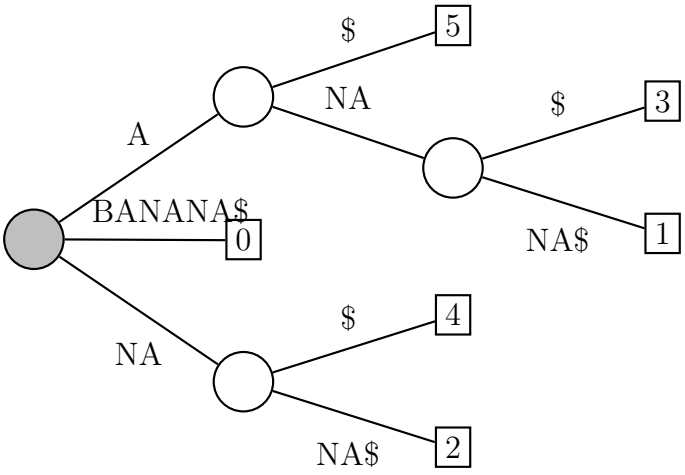


FIGURE 1.8 – *Exemple d'arbre de suffixes.* La chaîne C utilisée ici est la chaîne "BANANA".

- il existe une bijection entre l'ensemble des suffixes de C et l'ensemble des chemins allant de la racine à une feuille de $\mathcal{A}_C$;
- dans chaque feuille de $\mathcal{A}_C$ est stockée la position dans C du suffixe lu le long du parcours depuis la racine.

La figure 1.8 présente un exemple d'arbre de suffixe.

Une fois l'arbre $\mathcal{A}_C$ construit, pour obtenir toutes les occurrences de Q dans C, il suffit de parcourir $\mathcal{A}_C$ depuis le sommet en suivant le chemin indiqué par les symboles de Q. Ce parcours peut être interrompu pour deux raisons. Soit on ne trouve plus de correspondance et alors on peut conclure que C ne contient pas Q. Soit on a consommé tous les symboles de Q. Dans ce dernier cas, on est arrivé à un nœud de $\mathcal{A}_C$ tel que chacune des feuilles du sous-arbre partant de ce nœud correspond à une occurrence différente de Q dans C. Il suffit alors de parcourir ce sous-arbre pour obtenir lesdites occurrences.

Le coût de la recherche du motif Q dans la base se fait avec une complexité temporelle en $O(\min(m \log l, m \log t))$ (ou $O(m \times t)$ si l'on s'autorise une complexité spatiale en $\Theta(l \times t)$ au lieu de $O(l)$). Dans les deux cas, l'influence de la taille de l'alphabet t sur la complexité peut être une limitation importante à l'utilisation des arbres de suffixes pour certaines applications.

Tableaux de suffixes

En se basant sur la remarque que le temps de construction de la structure d'index est moins critique que les deux autres paramètres que sont, d'une part, l'espace occupé par cette structure et, d'autre part, la complexité temporelle d'une requête, Manber et Myers [Manber 90] ont proposé une nouvelle structure, appelée *tableau de suffixes*.

Un tel tableau stocke, à la position i, le $i^{\text{ième}}$ suffixe de C selon l'ordre lexicographique, comme présenté en tableau 1.2. Il est à noter que le symbole \$ est considéré comme précédent tous les symboles de Σ dans l'ordre lexicographique. L'espace occupé par un tel tableau est en $O(l)$.

indice dans $\mathcal{T}_C$	indice dans C	suffixe correspondant
0	5	A$
1	3	ANA$
2	1	ANANA$
3	0	BANANA$
4	4	NA$
5	2	NANA$

TABLEAU 1.2 – *Exemple de tableau de suffixes.* La chaîne C utilisée ici est la chaîne "BANANA". On notera que la troisième colonne du tableau est ici représentée dans un but de clarté, mais que $\mathcal{T}_C$ se limite à mettre en correspondance les deux premières colonnes.

La recherche exacte d'un motif Q dans $\mathcal{T}_C$ utilise l'idée que toutes les occurrences de Q dans C auront des entrées contigües dans $\mathcal{T}_C$, de par l'utilisation de l'ordre lexicographique. Ainsi, il suffit de chercher une entrée débutant par Q dans $\mathcal{T}_C$ (cela se fait, en utilisant la dichotomie, en $O(\log l)$ comparaisons, chaque comparaison ayant un coût au plus égal à m, soit en $O(m \log l)$) pour avoir une première correspondance. Ensuite, il suffit d'étendre la sélection autour de cette correspondance et ainsi, de proche en proche, on obtient toutes les occurrences de Q dans C.

De façon similaire à l'utilisation des n-grammes, la manipulation de tableaux de suffixes, si elle permet des recherches efficaces dans des grands volumes de données, n'autorise pas de déformation temporelle, ce qui justifie de s'intéresser à des approches plus flexibles telles que BLAST.

BLAST

Basic Local Alignment Search Tool (BLAST), introduit dans [Altschul 90], est un ensemble d'heuristiques très largement utilisé dans le domaine de la génomique. Il s'agit d'isoler efficacement les sous-séquences communes entre une requête et une base (supposées ici de tailles comparables) à moindre coût, tout en autorisant des déformations locales.

Pour ce faire, on se fixe un seuil S de similarité entre sous-chaînes et on souhaite retrouver toutes les sous-chaînes de la base ayant une similarité (au sens de la plus longue sous-séquence commune pondérée) au moins égale à S avec une sous-chaîne de la requête. Les pondérations utilisées sont fournies par une table (par exemple, pour les acides aminés, une table couramment utilisée est la matrice *BLOcks of amino acid SUbstitution Matrix* (BLOSUM) introduite dans [Henikoff 92]).

On commence alors par extraire de la requête tous les n-grammes [2] (il en existe $m - n + 1$ pour une requête de taille m). Ensuite, on étend cette liste de mots candidats en y ajoutant les n-grammes du dictionnaire ayant une similarité à une des sous-chaînes extraites au moins égale à T. Pour tous les mots obtenus, on effectue une recherche exacte dans la base (par exemple organisée en tableau de suffixes,

2. dans le cas de l'ADN, la valeur utilisée est 11 alors que pour les protéines, les auteurs utilisent $n = 3$.

mais certains travaux ont montré que l'utilisation d'un fichier inversé peut s'avérer opportune [Puglisi 06]). On obtient ainsi un ensemble de positions candidates de la base. Il suffit alors de chercher à étendre ces correspondances tant que le score ne diminue pas en-dessous de X fois le meilleur score obtenu jusqu'alors et de vérifier si cela suffit à atteindre un score de S.

Le résultat obtenu, s'il n'est pas garanti qu'il contienne l'ensemble des sous-chaînes de la base étant en correspondance avec une sous-chaîne de la requête avec un score au moins égal à S, en fournit une approximation raisonnable.

Altschul, Madden et Schaffer [Altschul 97] proposent trois améliorations à cela. Tout d'abord, partant de la remarque que l'étape la plus coûteuse dans le processus décrit ci-dessus est l'étape d'extension des correspondances de taille n, les auteurs proposent de ne limiter l'extension qu'aux régions où deux correspondances cohérentes (c'est-à-dire avec un même décalage) ont été observées avec un écart plus petit ou égal à A. Cette restriction étant forte, les auteurs préconisent de baisser le seuil T pour compenser la trop grande sélectivité induite par la nouvelle contrainte.

Ensuite, pour les régions où un n-gramme issu de Q est trouvé, ils proposent d'utiliser un algorithme comparable à la DTW pour étendre les correspondances tout en autorisant des insertions et des suppressions. Cette adaptation de BLAST est appelée *Gapped BLAST*.

Enfin, la troisième proposition de l'article est d'utiliser le résultat d'une exécution de BLAST pour avoir une représentation plus fine des symboles qui pourraient être présents à chaque position dans la requête. Plus précisément, après chaque exécution de BLAST, on évalue la probabilité de chaque symbole à chaque position de la requête (on a donc une matrice de taille $m \times t$) à l'aide des correspondances retournées et on construit l'ensemble des mots de taille n à rechercher en se basant sur cette matrice (et non plus sur des tables de similarité telles que BLOSUM), sans modifier la suite du processus. Ce processus peut être itéré pour affiner les probabilités obtenues.

De manière générale, BLAST fournit une méthodologie intéressante car adaptée au passage à l'échelle et relativement flexible. En effet, comme le montre *Gapped BLAST*, il est relativement aisé de modifier la partie fine de la recherche pour l'adapter à un problème particulier.

1.4 Séquences multimédias

Au-delà des deux grandes familles d'approches présentées plus haut, de nombreux algorithmes de recherche de *media* séquentiels dans de grandes bases ont été développés pour des applications précises. Nous présentons dans cette partie des cadres d'application dont les caractéristiques propres justifient les approches proposées.

1.4.1 Images clés

Une première approche, largement utilisée dans le domaine de la recherche de vidéos, et notamment dans [Sivic 03], est de résumer une séquence multimédia par un ensemble d'éléments saillants appelés, dans le cas de la vidéo, *images clés*.

Ainsi, on transforme une base de données de vidéos en une base d'images fixes, chaque image étant associée à la vidéo à laquelle elle appartient. La requête se fait

alors en deux temps. Tout d'abord, on résume le document requête avec le même processus que celui utilisé pour la base. Ensuite, pour chaque image clé du document requête, on interroge la base en utilisant les techniques issues du domaine de la recherche d'images fixes.

Les approches basées sur ce paradigme utilisent généralement une vérification *a posteriori* pour s'assurer de la cohérence temporelle des résultats fournis. Ainsi, un document de la base qui ne partage qu'une image clé avec la requête sera pénalisé par rapport à un document qui partage un certain enchaînement des images clés. L'intérêt de ces méthodes est que la robustification temporelle, qui est coûteuse, ne sera effectuée que pour un très petit sous-ensemble des documents de la base.

Un exemple de robustification temporelle est proposé dans [Douze 08], où les auteurs utilisent un *histogramme de Hough*. L'idée est de s'intéresser, pour une vidéo requête, à l'ensemble des triplets $(C, \delta t, s)$ retournés par le système de recherche d'images fixes, où δt est le décalage entre les positions temporelles dans Q et C de deux images mises en correspondance avec un score s. L'espace des δt est ensuite quantifié et, pour chaque vidéo candidate C, on regroupe les triplets par indice de quantification de δt, ce qui fournit un score pour chaque couple $(C, i_{\delta t})$ où $i_{\delta t}$ est l'indice de quantification considéré et c'est ce score qui sera considéré comme une mesure de la similarité entre Q et C.

Utiliser des images clés ne réduit pas l'ordre de grandeur de la complexité de l'algorithme de recherche, car le nombre d'images clés extraites d'un signal de longueur n est en $\Theta(n)$. Par contre, la constante qui n'apparaît pas dans les notations de Landau est, elle, beaucoup plus petite.

Lorsque l'on traite de l'audio, néanmoins, la notion de trame clé a moins de sens car la dynamique du signal est plus élevée et, ainsi, il est difficile de trouver une trame qui soit représentative d'un voisinage temporel de plusieurs secondes.

1.4.2 Identification de chansons

De la même façon, les deux systèmes commerciaux d'identification de chansons les plus répandus se basent sur une recherche de trames (sans sélection de trame clé) suivie d'une robustification temporelle.

Le premier, développé par Philips [Haitsma 01], propose de structurer la base de données à l'aide d'une table de hachage. La fonction de hachage correspondante est la quantification en une signature binaire de 32 bits des descripteurs extraits sur chaque trame. Au moment de la requête, on retourne toutes les chansons qui partagent au moins une clé de hachage avec la requête et la robustification temporelle consiste à considérer une plage de taille fixe (3 secondes dans leur cas) autour de la correspondance trouvée et de comparer bit à bit les signatures obtenues pour la requête et pour le candidat. Cette approche peut être vue comme une version simplifiée de la robustification temporelle opérée dans BLAST.

Le second, appelé Shazam [Wang 06], organise également sa base à l'aide d'une table de hachage. La robustification temporelle utilisée est ici un système de vote basé sur un histogramme de Hough, tel que présenté plus haut.

Si ces méthodes sont particulièrement adaptées au passage à l'échelle, elles ne sont utilisables que dans les cas où les déformations temporelles considérées sont

faibles car elles ne proposent pas de modèle de déformations temporelles.

1.4.3 Mélanges de gaussiennes

Une autre possibilité pour aborder le problème sans se soucier du caractère temporel des documents traités est de considérer chaque séquence comme un ensemble de réalisations d'une variable aléatoire multidimensionnelle. Le problème est alors, étant donné un ensemble d'observations, de retrouver la loi de probabilité de cette variable. Un modèle couramment utilisé dans ce cadre est le modèle de mélange de gaussiennes, ou *Gaussian Mixture Model* (GMM). Pour comparer deux séquences, il convient alors de comparer les deux modèles de distribution qui leur sont attribués. Pour ce faire, on aimerait pouvoir calculer une divergence symétrique de Kullback-Leibler entre les deux modèles :

$$KL2(p_Q, p_C) = KL(p_Q \| p_C) + KL(p_C \| p_Q). \tag{1.32}$$

Or, il n'existe pas de forme analytique permettant de calculer cette quantité pour des GMMs. Ben, Betser, Bimbot et Gravier [Ben 04] en proposent une approximation définie par :

$$\hat{KL2}(p_Q, p_C) = \sum_{k=1}^{K} \sum_{d=1}^{\dim} w_k \cdot \frac{(m_Q^{k,d} - m_C^{k,d})^2}{\sigma_{k,d}^2}, \tag{1.33}$$

où $m_Q^{k,d}$ et $m_C^{k,d}$ sont les $d^{\text{ième}}$ coordonnées des centres des $k^{\text{ième}}$ gaussiennes de chacun des deux modèles et où les $\sigma_{k,d}^2$ sont les éléments de la matrice de covariance de la $k^{\text{ième}}$ gaussienne des mélanges (on utilise ici l'hypothèse simplificatrice que les $k^{\text{ième}}$ gaussiennes des deux modèles partagent les mêmes matrices de covariance et que celles-ci sont diagonales). Cette métrique a donc pour avantage d'être calculable directement à partir des paramètres des modèles retenus.

Toutefois, une limitation majeure de ce type d'approches est l'absence totale de prise en compte de l'aspect temporel des documents considérés qui mène à une perte d'information importante.

1.4.4 Recherche de reprises musicales

Dans le domaine de la recherche de reprises de chansons, la notion de similarité est très particulière. En effet, deux versions d'un même morceau peuvent à la fois avoir des tonalités différentes et ne pas être structurées de la même façon. Ainsi, il est nécessaire d'utiliser des descripteurs adaptés que sont les *vecteurs de chroma*.

Dans leur forme la plus couramment utilisée, ces descripteurs sont de dimension 12, chaque dimension correspondant à l'un des demi-tons de la gamme chromatique, comme illustré en figure 1.9. Ces descripteurs encodent, pour une fenêtre de temps donnée (souvent de l'ordre de quelques millisecondes), l'énergie des différentes bandes de fréquence correspondant aux demi-tons, avec un repliement à l'octave. De plus, pour cette application, les algorithmes considérés doivent rechercher des sous-séquences similaires entre les deux séquences considérées plutôt qu'un alignement global d'une séquence sur l'autre.

L'algorithme fournissant la mesure de similarité la plus pertinente actuellement a été proposé par Serrà, Serra et Andrejzak [Serrà 09].

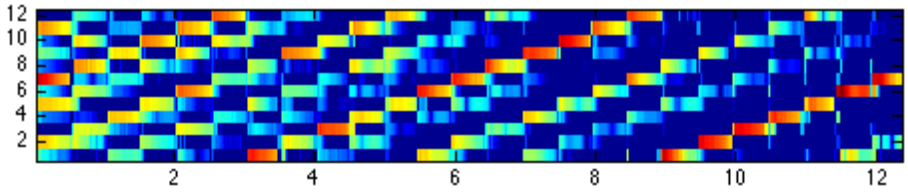


FIGURE 1.9 – *Exemple de chromagramme.* Le fichier sonore correspondant est un enregistrement de piano dans lequel un même accord est répété régulièrement en étant décalé d'un demi-ton. On note ici que ce décalage d'un demi-ton correspond à une permutation circulaire des dimensions du chromagramme.

Pour comparer deux chansons, les auteurs commencent par transposer l'un des deux morceaux dans la tonalité de l'autre. Pour ce faire, ils proposent de calculer la moyenne des vecteurs de chromas pour chacune des chansons considérées. Ensuite, le vecteur de chromas de la requête est conservé tel quel et l'on estime, pour chacune des 12 permutations circulaires possibles sur les dimensions des descripteurs, le produit scalaire entre le vecteur de chromas moyen de la requête et le vecteur de chromas moyen du candidat de la base auquel on aurait appliqué la permutation considérée. La permutation qui maximise le produit scalaire est considérée comme celle permettant de transposer la chanson candidate dans la tonalité de la requête.

Le calcul de similarité, lui, recherche la plus longue sous-séquence commune entre les séquences en utilisant la méthode présentée en section 1.2.2 avec une matrice binarisée définie comme :

$$S^k_{i,j} = (q_i \in \mathrm{NN}_k(c_j, Q)) \wedge (c_j \in \mathrm{NN}_k(q_i, C)). \tag{1.34}$$

Autrement dit, cette matrice encode, à l'indice (i, j), l'information selon laquelle c_j est un des k-plus proches voisins de q_i et réciproquement. Cette réciprocité, appelée *Reverse Nearest Neighbour* (R-NN) en anglais est connue comme étant un frein important au passage à l'échelle pour la recherche de plus proches voisins [Yang 01].

De plus, on souhaite absorber de légers sauts, d'une manière similaire à ce que supporte *Gapped BLAST* par exemple. La formule de récurrence utilisée est alors la suivante :

$$\gamma_{i,j} = \begin{cases} 1 + \max(\gamma_{i-1,j-1}, \gamma_{i-1,j-2}, \gamma_{i-2,j-1}) & \text{si } S^k_{i,j} \\ \max(0, \tilde{\gamma}_{i-1,j-1}, \tilde{\gamma}_{i-1,j-2}, \tilde{\gamma}_{i-2,j-1}) & \text{sinon} \end{cases}, \tag{1.35}$$

avec

$$\tilde{\gamma}_{i,j} = \gamma_{i,j} - \lambda S^k_{i,j}. \tag{1.36}$$

Dans le cas où λ tend vers $+\infty$, cette formule de récurrence peut être simplifiée et l'on se ramène à un algorithme classique de plus longue sous-séquence commune entre deux séquences sans tolérance aux sauts. Au contraire, plus λ est petit et moins les sauts sont pénalisés.

Dans tous les cas, la similarité entre deux séquences est définie comme le maximum des $\gamma_{i,j}$.

Cette méthode, si elle fournit les meilleurs résultats actuellement présentés dans la communauté de la recherche de reprises, est extrêmement coûteuse et les auteurs ne proposent notamment pas de méthode d'indexation associée.

Pour pallier ce problème, Di Bucchio, Montecchio et Orio [Di Buccio 10] proposent d'utiliser une stratégie appelée *sac de descripteurs*, ou *Bag Of Features* (BOF). Le principe est ici de segmenter arbitrairement chaque chanson et de représenter chacun des segments obtenus par le multiensemble de ses descripteurs quantifiés. À chacun des indices de quantification correspond une entrée dans le fichier inversé utilisé pour indexer la base. Ainsi, au moment d'effectuer une requête, la *similarité cosinus* entre le multiensemble M_{Q_i} d'un segment Q_i de la requête et le multiensemble M_{C_j} d'un segment C_j de la base est calculée en interrogeant le fichier inversé pour chacun des éléments de M_{Q_i} :

$$\cos(Q_i, C_j) = \sum_{e \in M_{Q_i}} \frac{\text{card}(e, M_{Q_i})}{\|M_{Q_i}\|} \cdot \frac{\text{card}(e, M_{C_j})}{\|M_{C_j}\|} \tag{1.37}$$

où $\text{card}(e, M)$ est la cardinalité du symbole e dans le multiensemble M et où $\|M\|$ est la norme du vecteur de fréquence associé au multiensemble M. Ensuite, plus les séquences Q et C contiennent des paires de segments (Q_i, C_j) similaires, plus leur similarité sera considérée élevée.

Si le temps de calcul de cette approche est largement inférieur à celui de l'approche proposée par Serrà, Serra et Andrejzak [Serrà 09], la qualité des résultats retournés est significativement moins bonne.

1.4.5 Recherche de mots parlés

Un autre domaine où la variabilité entre différentes occurrences d'un même motif est grande est la recherche de mots parlés. La requête utilisée ici est donc un enregistrement sonore correspondant à l'énonciation d'un mot et la recherche consiste à retrouver, dans une base d'enregistrements sonores, les répétitions de ce mot, possiblement énoncées par un autre locuteur et avec des dynamiques différentes.

Fraihat et Glotin [Fraihat 10] proposent pour cela d'utiliser une version binarisée des descripteurs MFCC classiquement utilisés pour ce type d'applications. La comparaison des séquences se fait en suite à l'aide d'un algorithme d'alignement dynamique tel que présenté en section 1.2. Les auteurs suggèrent une pénalisation des chemins diagonaux, arguant qu'une telle application implique le recours intensif aux transitions horizontales et verticales à cause de la grande variabilité existant entre les différentes occurrences d'un même mot. Nous reviendrons sur ce point dans le chapitre 4 de ce manuscrit.

Les expériences présentées montrent que la modification de la description comme la pondération des transitions améliore la qualité des résultats de la recherche.

1.5 Bilan

Nous avons présenté ici un éventail de méthodes issues de divers domaines et ayant pour but de permettre la recherche dans des bases de données de séquences. On peut classer les approches présentées selon deux grands paradigmes, tous deux basés sur la remarque que la comparaison fine entre séquences est trop coûteuse pour

que l'on puisse comparer une requête à l'ensemble des séquences de la base avec ce type d'outils.

Certaines approches font le choix d'estimer une version approchée peu coûteuse de ce que renverrait une comparaison fine telle que la DTW. Dans ce cadre, *i*SAX est la méthode la plus aboutie pour permettre le passage à l'échelle. D'autres méthodes proposent d'effectuer un filtrage basé sur un alignement rigide entre sous-séquences, puis d'effectuer une comparaison fine entre la requête et les sous-séquences pour lesquelles la première étape a conclu à une possibilité de correspondance. L'éventail des propositions est ici plus large, allant des propositions issues de la génomique telles que BLAST à d'autres venant du multimédia utilisant les techniques déjà développées en recherche de plus proches voisins dans les $\mathbb{R}$-espaces vectoriels.

Dans la suite de ce document, nous nous concentrerons sur ces deux grandes familles de méthodes. Nous proposerons tout d'abord un algorithme basé sur *i*SAX affichant des gains significatifs en termes de coût de calcul. Nous présenterons ensuite une méthode d'identification de reprises musicales utilisant un premier filtrage des régions temporelles susceptibles d'être mises en correspondance avec la requête, avant d'appliquer un algorithme de mise en correspondance plus fin permettant de robustifier les résultats. Enfin, nous discuterons de la nécessité d'utiliser des métriques de comparaison fines dans les principaux sous-domaines issus de la recherche de séquences multimédias.

Bibliographie

[Altschul 90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers & D. J. Lipman. *Basic Local Alignment Search Tool.* Journal of Molecular Biology, vol. 215, no. 3, pages 403–410, 1990.

[Altschul 97] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller & D.J. Lipman. *Gapped BLAST and PSI-BLAST : a New Generation of Protein Database Search Programs.* Nucleic Acids Research, vol. 25, no. 17, pages 3389–3402, 1997.

[Ben 04] M. Ben, M. Betser, F. Bimbot & G. Gravier. *Speaker Diarization Using Bottom-up Clustering Based on a Parameter-derived Distance Between GMMs.* In Proceedings of the International Conference on Spoken Language Processing, pages 2329–2332, 2004.

[Camerra 10] A. Camerra, T. Palpanas, J. Shieh & E. J. Keogh. *iSAX 2.0 : Indexing and Mining One Billion Time Series.* In Proceedings of the IEEE International Conference on Data Mining, 2010.

[Chen 04] L. Chen & R. Ng. *On the Marriage of Lp-norms and Edit Distance.* In Proceedings of the International Conference on Very Large Data Bases, pages 792–803, 2004.

[Di Buccio 10] E. Di Buccio, N. Montecchio & N. Orio. *FALCON : FAst Lucene-based Cover sOng identification.* In Proceedings of the ACM International conference on Multimedia, pages 1477–1480, 2010.

[Douze 08] M. Douze, A. Gaidon, H. Jégou, M. Marszałek & C. Schmid. *INRIA-LEARs video copy detection system.* In Proceedings of the TRECVID Workshop, 2008.

[Faloutsos 94] C. Faloutsos, M. Ranganathan & Y. Manolopoulos. *Fast Subsequence Matching in Time-series Databases.* In Proceedings of the ACM SIGMOD Conference, pages 419–429, 1994.

[Fraihat 10] S. Fraihat & H. Glotin. *Indexation rapide de documents audio par traitement morphologique de la parole.* Ingénierie des systèmes d'information, vol. 15, pages 29–48, 2010.

[Haitsma 01] J. Haitsma, T. Kalker & J. Oostveen. *Robust Audio Hashing for Content Identification.* In Proceedings of the International Workshop on Content-Based Multimedia Indexing, 2001.

[Henikoff 92] S. Henikoff & J.G. Henikoff. *Amino Acid Substitution Matrices from Protein Blocks.* Proceedings of the National Academy of Sciences of the United States of America, vol. 89, no. 22, pages 10915–10919, 1992.

[Keogh 01] E. Keogh, K. Chakrabarti, M. Pazzani & S. Mehrotra. *Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases.* Knowledge And Information Systems, vol. 3, no. 3, pages 263–286, 2001.

[Keogh 05] E. Keogh & C.A. Ratanamahatana. *Exact Indexing of Dynamic Time Warping.* Knowledge And Information Systems, vol. 7, no. 3, pages 358–386, 2005.

[Lin 07] J. Lin, E. Keogh, L. Wei & S. Lonardi. *Experiencing SAX : a Novel Symbolic Representation of Time Series.* Data Mining and Knowledge Discovery, vol. 15, no. 2, pages 107–144, 2007.

[Manber 90] U. Manber & G. Myers. *Suffix Arrays : a New Method for On-line String Searches.* In Proceedings of the annual ACM-SIAM symposium on Discrete algorithms, pages 319–327, 1990.

[McCreight 76] E. McCreight. *A Space-Economical Suffix Tree Construction Algorithm.* Journal of the ACM, vol. 23, pages 262–272, 1976.

[Puglisi 06] S. Puglisi, W. Smyth & A. Turpin. *Inverted Files Versus Suffix Arrays for Locating Patterns in Primary Memory.* In Proceedings of the International Conference on String Processing and Information Retrieval, pages 122–133, 2006.

[Sakoe 78] H. Sakoe & S. Chiba. *Dynamic Programming Algorithm Optimization for Spoken Word Recognition.* IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 26, pages 43–49, 1978.

[Salvador 04] S. Salvador & P. Chan. *FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space.* In Proceedings of the KDD Workshop on Mining Temporal and Sequential Data, pages 70–80, 2004.

[Serrà 09] J. Serrà, X. Serra & R.G. Andrzejak. *Cross Recurrence Quantification for Cover Song Identification.* New Journal of Physics, vol. 11, 2009.

[Shieh 08] J. Shieh & E. Keogh. *iSAX : Indexing and Mining Terabyte Sized Time Series.* In Proceedings of the ACM International Conference on Knowledge Discovery and Data mining, pages 623–631, 2008.

[Sivic 03] J. Sivic & A. Zisserman. *Video Google : a Text Retrieval Approach to Object Matching in Videos.* In Proceedings of the International Conference on Computer Vision, volume 2, pages 1470–1477, 2003.

[Ukkonen 95] E. Ukkonen. *On-line Construction of Suffix Trees.* Algorithmica, vol. 14, pages 249–260, 1995.

[Wang 06] A. Wang. *The Shazam music recognition service.* Communications of the ACM, vol. 49, pages 44–48, 2006.

[Weiner 73] P. Weiner. *Linear Pattern Matching Algorithms.* In Proceedings of the Annual Symposium on Switching and Automata Theory, pages 1–11, 1973.

[Yang 01] C. Yang & K. I. Lin. *An Index Structure for Efficient Reverse Nearest Neighbor Queries.* In Proceedings of the International Conference on Data Engineering, pages 485–492, 2001.

[Yi 98] B. Yi, H. V. Jagadish & C. Faloutsos. *Efficient Retrieval of Similar Time Sequences Under Time Warping.* In Proceedings of the International Conference on Data Engineering, pages 201–208, 1998.

Chapitre 2

Alignement dynamique et recherche approximative

La complexité temporelle du calcul de DTW rend cette dernière très coûteuse et impossible à utiliser telle quelle pour des recherches exhaustives dans de grandes bases de données de séquences. Comme présenté au chapitre 1, un moyen de contourner cette limitation est d'utiliser des bornes inférieures à la DTW peu coûteuses dans le but de réserver les calculs exacts de DTW aux séquences candidates les plus prometteuses.

Les deux contributions centrales de ce chapitre sont, d'une part, la proposition de bornes inférieures approximatives permettant d'accélérer la recherche dans les bases de séquences au moyen de la DTW et, d'autre part, un nouveau schéma d'indexation permettant une répartition équilibrée des séquences au sein d'un arbre d'indexation inspiré de *i*SAX.

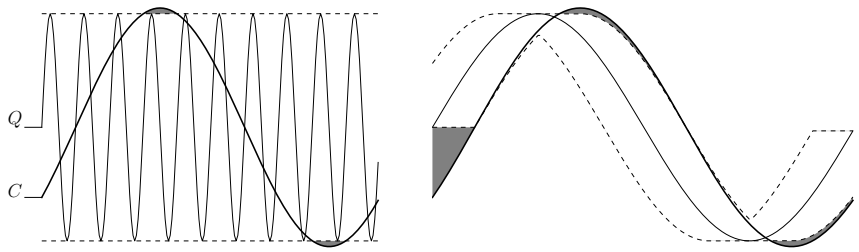
Une partie de ces travaux a été publiée dans [Tavenard 11].

2.1 Bornes inférieures approximatives

On peut remarquer que même LB_Keogh, réputée pour être la borne inférieure fournissant l'estimation de la DTW la plus précise, sous-estime la DTW d'au moins 20% pour les expériences présentées dans [Keogh 05]. Or, moins l'estimation de la DTW est précise, plus le nombre de calculs de DTW nécessaire est grand, comme expliqué en section 1.2.3.

Il est donc nécessaire d'imaginer de nouvelles bornes inférieures qui soient plus proches de la véritable valeur de DTW. En effet, Ratanamahatana et Keogh [Ratanamahatana 04] montrent qu'une borne inférieure imaginaire, qui renverrait toujours une valeur égale à 99% de la véritable valeur de DTW permettrait d'effectuer 200 fois moins de calculs de DTW que LB_Keogh.

La figure 2.1 explique une des limitations fortes des bornes inférieures existantes :



(a) (b)

FIGURE 2.1 – *DTW et enveloppes.* Enveloppes obtenues pour deux séquences de même taille et une contrainte de Sakoe-Chiba utilisant la même largeur de bande. On voit ici que la figure de droite, présentant une séquence Q qui varie plus lentement que celle de gauche, cette séquence mieux approximée par son enveloppe. Il en résulte que l'estimation de la DTW entre Q et C fournie par LB_Keogh, représentée par l'aire grisée, est plus fine pour le cas de droite.

elles se basent sur une enveloppe de l'une des deux séquences, or, dans le cas de séquences ayant une dynamique forte, cette enveloppe peut s'avérer une estimation très peu précise de la séquence qu'elle représente. Considérons la séquence candidate représentée sur les deux figures. Si l'on utilise LB_Keogh, du fait de l'enveloppe utilisée, cette séquence sera considérée comme plus similaire à la séquence de la figure 2.1a (comme le montre l'aire grisée moins importante) qu'à celle de la figure 2.1b, ce qui est à la fois contraire à l'intuition et aux résultats retournés par la DTW.

Cette faiblesse explique que l'on n'ait pas réussi à proposer de borne inférieure plus efficace que LB_Keogh se basant sur l'enveloppe des séquences. Or, l'avantage de ces approches utilisant les enveloppes des séquences est qu'elles ont une complexité linéaire qui justifie leur utilisation.

Nous proposons ici de relaxer l'hypothèse, énoncée dans [Faloutsos 94], selon laquelle l'étape de filtrage qui précède le calcul exact de similarité ne doit introduire aucun faux négatif (c'est-à-dire que l'on ne doit en aucun cas, à aucune étape du processus, être susceptible d'écarter une séquence qui est en fait celle recherchée). Nous chercherons alors à utiliser des fonctions qui, si elles ne sont pas des bornes inférieures pour la DTW, ont la bonne propriété de lui être inférieure avec une grande probabilité tout en approchant au mieux la valeur exacte de DTW. Nous qualifierons ces fonctions de *bornes inférieures approximatives*.

En conséquence, nous montrerons que les bornes inférieures approximatives introduites permettent de diminuer le phénomène observé en figure 2.1.

Dans la suite de cette partie, nous nous concentrerons sur des bornes inférieures approximatives dérivées de LB_Keogh. Pour obtenir de telles bornes, nous étudierons dans un permier temps la possibilité de multiplier LB_Keogh par un facteur constant, ce qui ne change rien à sa complexité. Les limites de cette méthode nous mèneront à l'introduction d'une borne inférieure approximative plus fine, qui utilise l'information apportée par une borne supérieure dérivée de LB_Keogh, le coût de calcul de cette

Fonction	Valeur de T visée	Valeur médiane de T observée	I_T
LB_Keogh	—	0,916	[0,500 ;0,994]
αLB_Keogh	0,99	1.239	[0,677 ;1.345]
LBUB_Keogh	0,99	0,993	[0,929 ;1.000]

TABLEAU 2.1 – *Bornes inférieures approximatives visant un ajustement de 99%.* Nous présentons ici les valeurs médianes et les intervalles inter-quartiles observés pour T, notés I_T.

borne supérieure étant du même ordre de grandeur que celui de LB_Keogh. Nous étendrons ensuite nos propositions aux cas des autres bornes inférieures issues de l'état-de-l'art que sont LB_Yi et LB_PAA.

2.1.1 αLB_Keogh

Une façon simple d'obtenir une borne inférieure approximative à la DTW est d'apprendre sur un jeu de données le rapport moyen entre la DTW et la borne inférieure considérée, ici LB_Keogh. Ce rapport est appelé *ajustement* (on parle de *tightness* en anglais) et noté T. On peut ensuite calculer un facteur multiplicatif α à appliquer à LB_Keogh dans le but d'augmenter son ajustement. On note la fonction résultante αLB_Keogh. L'ajustement étant une valeur moyenne, il est alors possible que la fonction résultante soit supérieure à la valeur retournée par la DTW pour certains couples (Q, C) de séquences car la seule inégalité que l'on puisse garantir est la suivante :

$$\forall (Q, C), \alpha\text{LB\_Keogh}(Q, C) \leq \alpha \cdot \text{DTW}(Q, C). \tag{2.1}$$

On notera que le coût de calcul résultant de cette approche est égal à celui du calcul de LB_Keogh.

Pour valider cette idée, nous utilisons, rassemblées dans une seule et même base, l'ensemble des séquences contenues dans le jeu de données noté DS_1 par la suite (voir en section 2.1.8 pour une description précise de ce jeu de données). Le fait de rassembler ces séquences dans une même base permet d'avoir un comportement plus proche de celui de grandes bases de données, où les données stockées sont fortement hétérogènes. Le paramètre α est fixé de manière à atteindre un ajustement de 99% sur le jeu d'apprentissage.

Les deux premières lignes du tableau 2.1 montrent que LB_Keogh est, en médiane, 8% plus faible que la DTW. Ensuite, on peut noter que αLB_Keogh a tendance à surestimer la DTW, sa médiane étant supérieure à celle de la DTW de 24%. Remarquons enfin que les intervalles inter-quartiles pour ces deux méthodes sont étendus, ce qui implique, pour LB_Keogh, un nombre important de paires de séquences pour lesquelles la DTW est fortement sous-estimée et, dans le cas de αLB_Keogh, un nombre important de paires pour lesquelles la valeur retournée est une surestimation de la DTW, montrant ainsi qu'un simple facteur multiplicatif n'est pas suffisant pour obtenir une borne inférieure approximative fiable. En effet, appliquer aveuglément cette valeur unique α n'augmente que légèrement l'ajustement

dans les cas où le rapport entre DTW et LB_Keogh est grand, alors que cela l'augmente trop fortement dans les cas où ce rapport est proche de 1.

On voit ici qu'il est primordial de distinguer les cas où (i) la valeur retournée par la borne inférieure est petite car la valeur de DTW est elle-même petite des cas où (ii) cette valeur est petite car la borne inférieure n'est pas ajustée. Dans ce dernier cas, de larges facteurs multiplicatifs peuvent être appliqués à la borne inférieure tout en conservant un ajustement plus petit que 1 alors que dans le premier cas, il est nécessaire de ne pas trop augmenter la valeur fournie par la borne inférieure, sous peine de voir l'ajustement exploser.

Pour permettre cette distinction, nous décrivons dans la suite une *borne supérieure* pour la DTW inspirée de LB_Keogh permettant d'estimer la largeur de l'intervalle entre ces deux bornes et, par contrecoup, l'ajustement de la borne inférieure exacte. Cette fonction est utilisée pour estimer l'ajustement de LB_Keogh. Nous présentons ensuite l'utilisation d'un paramètre contrôlant le taux attendu de distances surestimées pour la définition d'une borne inférieure approximative. Nous montrons que ce paramètre est en lien étroit avec l'ajustement souhaité et permet de fournir des bornes inférieures approximatives couvrant une large gamme de compromis entre qualité des résultats et nombre de calculs de DTW à effectuer. Nous proposons enfin des algorithmes inspirés de ceux présentés dans [Keogh 05, Shieh 08], l'un pour la recherche séquentielle et l'autre utilisant un arbre d'indexation produit par *i*SAX.

2.1.2 UB_Keogh

LB_Keogh est calculée en approximant l'une des deux séquences, que nous notons Q, par son enveloppe. Le principe est d'estimer la similarité entre les deux séquences par la somme des distances entre chacun des éléments c_i de C et *le plus proche point* dans l'intervalle $[l_i; u_i]$, qui est le $i^{\text{ième}}$ élément de l'enveloppe de Q, comme décrit en section 1.2.3.

Nous proposons de définir une borne supérieure utilisant le même principe, en sommant les distances entre les éléments c_i et *le point le plus éloigné* dans $[l_i; u_i]$.

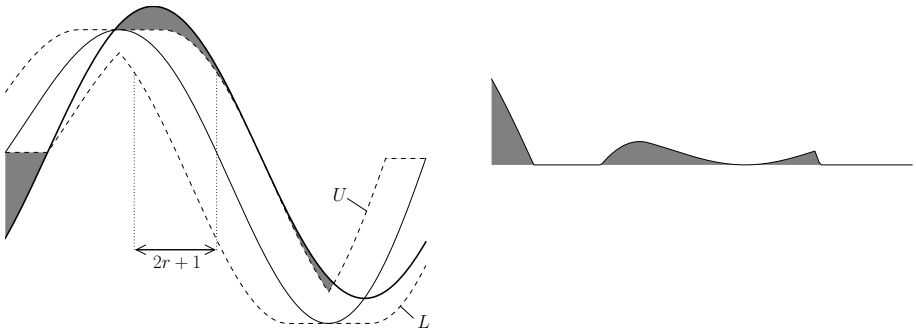
On définit :

$$\begin{aligned} \text{UB\_Keogh}(Q, C) &= \sum_{i=1}^{n} \begin{cases} (c_i - L_i) & \text{si } c_i > U_i \\ (U_i - c_i) & \text{si } c_i < L_i \\ \max(U_i - c_i, c_i - L_i) & \text{sinon} \end{cases} \qquad (2.2) \\ &= \sum_{i=1}^{n} \max(U_i - c_i, c_i - L_i) \qquad (2.3) \end{aligned}$$

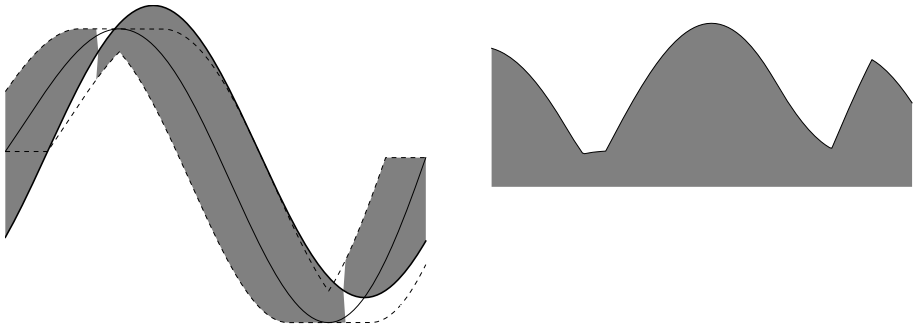
De par sa nature, cette borne supérieure a la même complexité de calcul que LB_Keogh. La figure 2.2 illustre graphiquement la définition de UB_Keogh en la mettant en regard de LB_Keogh.

Une fois UB_Keogh définie, lorsque l'on considère deux séquences Q et C, il est désormais possible de calculer à la fois une borne inférieure et une borne supérieure à la DTW.

L'étendue de l'intervalle obtenu fournit une indication quant à l'ajustement de la borne inférieure : si l'intervalle est petit, la borne inférieure est une bonne approximation de la DTW. Dans le cas contraire, les bornes inférieures et supérieures basant



(a) LB_Keogh(Q, C)



(b) UB_Keogh(Q, C)

FIGURE 2.2 – *Exemples de LB_Keogh et UB_Keogh.*

leurs calculs sur le même principe d'approximer une séquence par son enveloppe, il est probable que la borne inférieure soit peu ajustée.

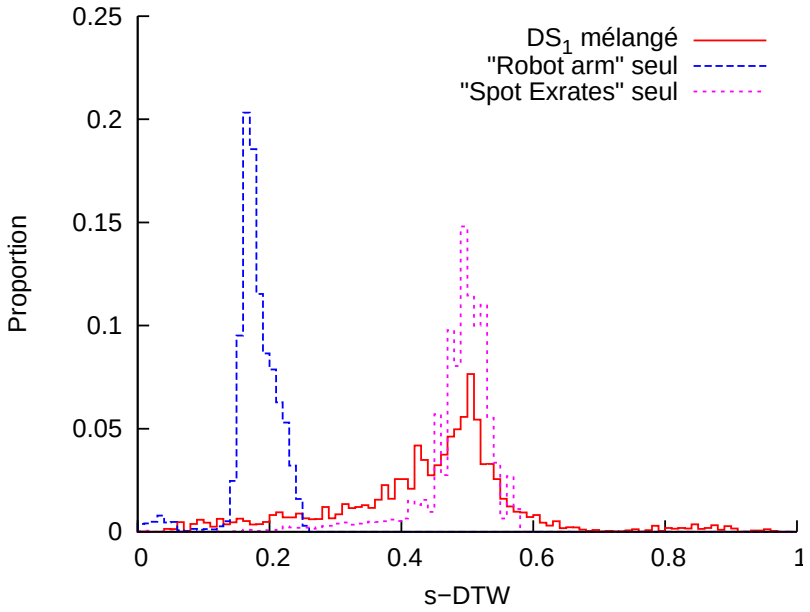
Une première option pourrait être, par exemple de toujours choisir comme borne inférieure approximative le milieu de l'intervalle défini par les bornes inférieure et supérieure exactes. Toutefois, cela ne nous donne aucune garantie quant au fait que la borne inférieure approximative obtenue soit inférieure à la DTW avec une grande probabilité, ce que nous recherchons. Pour atteindre cet objectif, nous introduisons dans la section suivante une méthode simple d'apprentissage de l'ajustement.

2.1.3 Apprentissage de l'ajustement

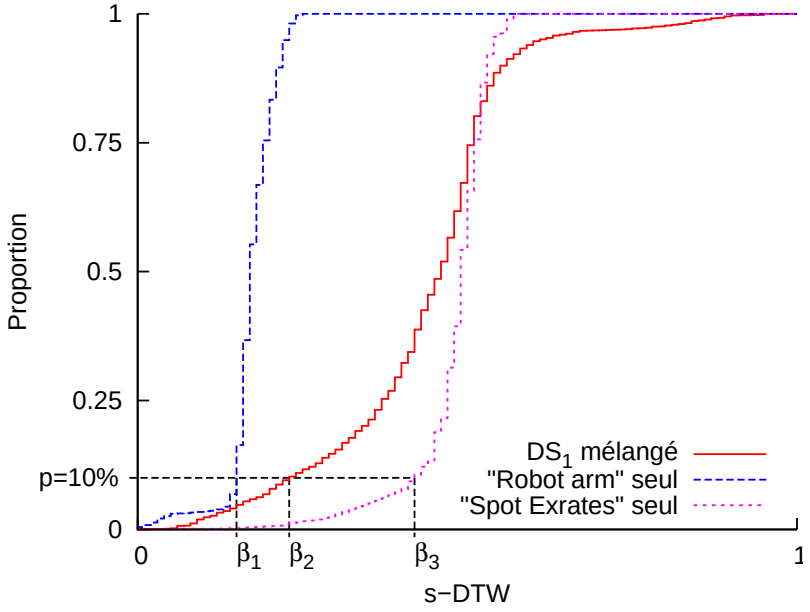
Il est possible d'apprendre l'allure de la fonction de répartition de la DTW dans l'intervalle formé par ses bornes inférieure et supérieure. Pour pouvoir comparer les valeurs obtenues, on remet celles-ci à l'échelle de la manière suivante :

$$\text{s-DTW}(Q, C) = \frac{\text{DTW}(Q, C) - \text{LB\_Keogh}(Q, C)}{\text{UB\_Keogh}(Q, C) - \text{LB\_Keogh}(Q, C)}. \tag{2.4}$$

L'allure de la distribution des valeurs de s-DTW est en lien étroit avec l'ajustement observé pour la borne inférieure sur le jeu d'apprentissage : plus la courbe est décalée vers la droite, moins la borne inférieure considérée est ajustée. La figure 2.3a présente la distribution des valeurs de s-DTW pour le jeu de données hétérogène présenté plus haut ainsi que deux sous-parties de ce jeu de données plus homogènes.



(a)



(b)

FIGURE 2.3 – *Fonction de densité de probabilité et fonction de répartition pour la DTW.* Les bornes inférieure et supérieure utilisées sont LB_Keogh et UB_Keogh.

Dans les trois cas, les courbes mettent en avant le manque d'ajustement de la borne inférieure utilisée. De plus, on note un décalage significatif entre les différentes courbes qui montre le peu de pertinence de l'utilisation d'un facteur α fixe à appliquer à la borne inférieure exacte.

Dans la suite, nous présentons un moyen d'utiliser la distribution des valeurs de s-DTW pour fixer un taux attendu de distances surestimées.

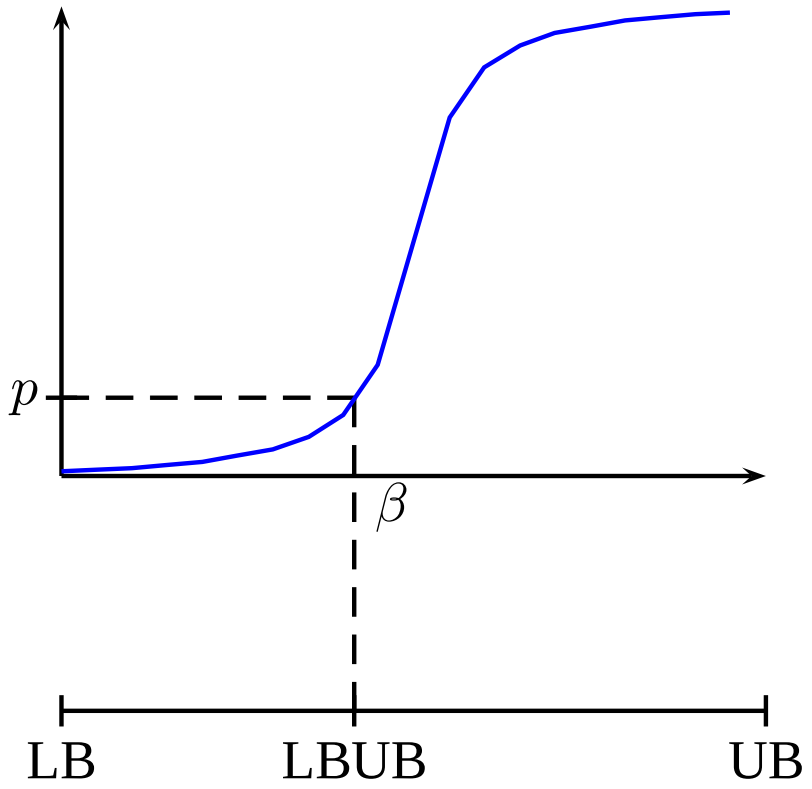


FIGURE 2.4 – *Principe d'estimation de la borne inférieure approximative.* Une fois le paramètre p fixé, la fonction de répartition apprise permet de calculer β et ainsi, en fonction des valeurs des bornes inférieure LB et supérieure UB, de trouver la valeur LBUB à donner à la borne inférieure approximative.

2.1.4 Utilisation du taux de distances surestimées

Supposons qu'un utilisateur souhaite accélérer une recherche dans une base de séquences au prix de potentiels faux négatifs. Cet utilisateur décide d'autoriser p =10% de surestimations lors de l'approximation des distances. Étant donnée la fonction de répartition de s-DTW, il est possible de déterminer la valeur β sur l'axe des abscisses qui correspond à la valeur p sur l'axe des ordonnées, comme présenté par les traits pointillés sur la figure 2.3b (on note qu'alors, selon la distribution observée sur le jeu de données considéré, la valeur de β varie, prenant les valeurs β_1 à β_3). La valeur de β retenue est alors utilisée pour calculer la valeur retournée par la borne inférieure approximative :

$$\text{LBUB\_Keogh} = \text{LB\_Keogh} + \beta(\text{UB\_Keogh} - \text{LB\_Keogh}) \tag{2.5}$$

qui sera alors utilisée pour filtrer les séquences les moins susceptibles d'être proches de la requête.

Ce principe est illustré en figure 2.4.

Il est important de noter que si l'on fixe $p = 0$, on a alors $\beta = 0$ et on retrouve les bornes inférieures exactes.

Pour certains utilisateurs, néanmoins, il est possible que la détermination du seuil p à utiliser soit peu intuitive. Il est alors préférable de fixer un ajustement médian T

à atteindre. Dans ce cas, sachant que la fonction f telle que $T = f(p)$ est croissante, il suffit d'inverser numériquement f (par exemple en utilisant la dichotomie) sur le jeu d'apprentissage pour obtenir la valeur de p à utiliser (et donc celle de β) suivant le processus décrit plus haut.

Le même raisonnement peut être utilisé dans le cas où l'on souhaite imposer une contrainte de complexité, c'est-à-dire de calculs de DTW attendus. On peut alors fixer la valeur p à utiliser en inversant numériquement la fonction qui exprime la complexité en fonction de p.

La dernière ligne du tableau 2.1 montre qu'en visant un ajustement de 99%, on obtient un résultat proche (99,3%) avec un intervalle inter-quartiles peu étendu, montrant la fiabilité de cette mesure.

Avant d'introduire les algorithmes correspondant à l'utilisation de ces bornes inférieures approximatives, nous présentons les bornes supérieures UB_Yi, UB_PAA et MAXDIST et démontrons que UB_Keogh, UB_Yi et UB_PAA majorent effectivement la DTW.

2.1.5 Démonstrations

Nous montrons ici que UB_Keogh est une borne supérieure pour la DTW contrainte à une bande de Sakoe-Chiba, et introduisons les bornes supérieures relatives à LB_Yi, LB_PAA et MINDIST ainsi que les preuves associées.

Proposition 2.1. *Pour toutes séquences Q et C de taille n, on a :*

$$UB\_Keogh(Q, C) \geq DTW(Q, C)$$

où la DTW considérée est contrainte à une bande de Sakoe-Chiba de largeur r.

Lemme 2.2. *Pour toutes séquences Q et C de taille n, on a :*

$$L_1(Q, C) \geq DTW(Q, C)$$

où la DTW considérée est contrainte à une bande de Sakoe-Chiba de largeur r.

Démonstration. Soient Q et C deux séquences de taille n. La distance L_1 correspond à un alignement pour lequel le $i^{\text{ième}}$ élément de Q est toujours mis en correspondace avec le $i^{\text{ième}}$ élément de C. Ainsi, cette distance correspond à l'un des chemins considérés par l'alignement dynamique, car il est toujours dans la bande de Sakoe-Chiba, quelle que soit la valeur de r. Or, la DTW sélectionnant le chemin de coût minimal, la distance L_1 est toujours supérieure ou égale à la DTW, ce qui conclut la preuve du lemme 2.2. Pour ce qui est de la preuve de la proposition 2.1, il suffit de remarquer que chaque terme de la somme intervenant dans la définition de UB_Keogh correspond exactement à un terme de la somme pour la distance L_1. La seule différence est alors que pour UB_Keogh, on somme les distances entre un point de la séquence candidate et le point le plus lointain correspondant dans l'enveloppe, alors que pour L_1 on s'intéresse à la distance entre un point de la séquence candidate et son correspondant dans la séquence requête, qui est, par définition, dans l'enveloppe. Il est alors clair que l'on a :

$$\text{UB\_Keogh}(Q, C) \geq L_1(Q, C) \geq \text{DTW}(Q, C) \tag{2.6}$$

□

Définition 2.1. *Nous définissons UB_Yi comme :*

$$UB\_Yi(Q,C) = \begin{cases} \min\left(\sum_i q_i - \min(C), \sum_i \max(Q) - c_i\right) & \textbf{(C1)} \\ \sum_i \max(q_i - \min(C), \max(C) - q_i) & \textbf{(C2)} \end{cases} \quad (2.7)$$

où les conditions C1 et C2 sont les suivantes :

(C1) $\min(Q) \geq \min(C)$,

(C2) $\min(Q) < \min(C)$.

On suppose, dans les deux cas, que $\max(Q) \geq \max(C)$. *Les cas où cette inégalité n'est pas vérifiée peuvent être traités de manière parfaitement symétrique.*

Proposition 2.3. *Pour toutes séquences Q et C de taille n, on a :*

$$UB\_Yi(Q,C) \geq DTW(Q,C)$$

où la DTW considérée n'est pas contrainte.

Démonstration. Tout comme pour UB_Keogh, il nous suffit de montrer que UB_Yi est une borne supérieure à la distance L_1 et la proposition sera démontrée. Soient Q et C deux séquences de taille n et soit W le chemin de coût minimum emprunté pour le calcul de la DTW non contrainte entre Q et C. Du lemme 2.2, on a :

$$\sum_{k=1}^{K} w_k \leq \sum_{i=1}^{n} S_{(i,i)} \quad (2.8)$$

Deux cas de figure peuvent alors se présenter.

Premier cas : (C1) On a donc ici $\max(Q) \geq \max(C)$ et $\min(Q) \geq \min(C)$.

Or, pour tout i dans $[\![1;n]\!]$, on a les deux inégalités suivantes :

$$S_{(i,i)} \leq q_i - \min(C) \quad (2.9)$$
$$S_{(i,i)} \leq \max(Q) - c_i. \quad (2.10)$$

En réinjectant ces inégalités dans le membre droit de l'inégalité 2.8, on obtient :

$$\sum_{k=1}^{K} w_k \leq \sum_{i=1}^{n} (q_i - \min(C)) \quad (2.11)$$
$$\sum_{k=1}^{K} w_k \leq \sum_{j=1}^{n} (\max(Q) - c_j) \quad (2.12)$$

et la propriété souhaitée est bien vérifiée.

Second cas : (C2) On a donc ici $\max(Q) \geq \max(C)$ et $\min(Q) < \min(C)$.

Or, pour tout i dans $[\![1;n]\!]$, on a au moins une des deux inégalités suivantes qui est vérifiée :

$$S_{(i,i)} \leq q_i - \min(C) \quad (2.13)$$
$$S_{(i,i)} \leq \max(C) - q_i. \quad (2.14)$$

On a donc toujours :

$$w_k \leq \max(q_i - \min(C), \max(C) - q_i). \tag{2.15}$$

En réinjectant cette inégalité dans le membre droit de l'inégalité 2.8, on obtient :

$$\sum_{k=1}^{K} w_k \leq n \cdot \sum_{i=1}^{n} \max(q_i - \min(C), \max(C) - q_i), \tag{2.16}$$

ce qui conclut notre preuve. □

Définition 2.2. *Nous définissons UB_PAA comme :*

$$UB\_PAA(Q,C) = \frac{n}{N} \cdot \left(\sum_{i=1}^{N} \max(\hat{U}_i - \bar{c}_i, \bar{c}_i - \hat{L}_i) \right) \tag{2.17}$$

$$+ \frac{n}{N} \cdot (\max(C) - \min(C)). \tag{2.18}$$

Proposition 2.4. *Pour toutes séquences Q et C de taille n, on a :*

$$UB\_PAA(Q,C) \geq DTW(Q,C)$$

où la DTW considérée est contrainte à une bande de Sakoe-Chiba de largeur r.

Lemme 2.5. *Pour toutes séquences Q et C de taille n, on a :*

$$UB\_PAA(Q,C) \geq UB\_Keogh(Q,C).$$

Démonstration. Il est clair qu'il suffit de démontrer le lemme 2.5 pour que, en utilisant la proposition 2.1, la proposition 2.4 soit établie.

Soient Q et C deux séquences de taille n et soit W le chemin de coût minimum emprunté pour le calcul de la DTW restreinte à une bande de Sakoe-Chiba de largeur r entre Q et C. Pour montrer :

$$\sum_{i=1}^{n} \max(U_i - c_i, c_i - L_i) \tag{2.19}$$

$$\leq \frac{n}{N} \cdot \left(\sum_{i=1}^{N} \max(\hat{U}_i - \bar{c}_i, \bar{c}_i - \hat{L}_i) + \max(C) - \min(C) \right) \tag{2.20}$$

il suffit de montrer :

$$\forall i \in [\![1; N]\!], \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} \max(U_j - c_j, c_j - L_j) \tag{2.21}$$

$$\leq \frac{n}{N} \cdot \left(\max(\hat{U}_i - \bar{c}_i, \bar{c}_i - \hat{L}_i) + (\max(C) - \min(C)) \right) \tag{2.22}$$

Soit $i \in [\![1; N]\!]$. En utilisant, pour tous $j \in [\![\frac{n}{N}(i-1)+1; \frac{n}{N}i]\!]$, la notation :

$$c_j = \bar{c_i} + \Delta c_j, \tag{2.23}$$

il vient :

$$\sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} \max(U_j - c_j, c_j - L_j) \tag{2.24}$$

$$= \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} \max(U_j - (\bar{c_i} + \Delta c_j), \bar{c_i} + \Delta c_j - L_j) \tag{2.25}$$

$$\leq \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} \max(\hat{U_i} - (\bar{c_i} + \Delta c_j), \bar{c_i} + \Delta c_j - \hat{L_i}) \tag{2.26}$$

$$\leq \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} \max(\hat{U_i} - \bar{c_i}, \bar{c_i} - \hat{L_i}) + |\Delta c_j| \tag{2.27}$$

$$\leq \frac{n}{N}\Big(\max(\hat{U_i} - \bar{c_i}, \bar{c_i} - \hat{L_i}) + \max(C) - \min(C)\Big) \tag{2.28}$$

ce qui conclut notre preuve. □

Dans le cas de MINDIST, la définition d'une borne supérieure correspondante n'est pas tout à fait directe. En effet, certaines bornes des intervalles considérés dans le calcul de MINDIST peuvent être infinies. Cela ne pose aucun problème tant que l'on s'intéresse à la distance entre un symbole d'une séquence requête et le plus proche point de l'intervalle en question. Par contre, lorsqu'il s'agit, comme dans le cas des bornes supérieures, de calculer la distance au point le plus éloigné de l'intervalle, on peut obtenir des valeurs infinies. Nous proposons donc de stocker, pour chaque nœud de l'arbre d'indexation, dans un tableau de taille N (où N est le nombre de symboles PAA utilisés pour le sous-échantillonnage) la valeur minimale ou maximale de l'ensemble des séquences stockées dans le nœud courant ou dans l'un de ses descendants, selon que c'est la borne inférieure ou supérieure qui pose problème. Finalement, la borne supérieure considérée fait référence au rectangle englobant d'un nœud (qui est alors bien défini puisque l'on a supprimé les bornes infinies) $R = (B, H)$. On rappelle ici que les éléments B_i et H_i sont respectivement les bornes inférieure et supérieure de l'ensemble des $i^{\text{ième}}$ éléments des séquences stockées dans le nœud correspondant.

Définition 2.3.

$$MAXDIST(Q, R) = \sqrt{\frac{n}{N}\sum_{i=1}^{N} \begin{cases} (\bar{q_i} - B_i)^2 & si\ \bar{q_i} > H_i \\ (H_i - \bar{q_i})^2 & si\ \bar{q_i} < B_i \\ \max(H_i - \bar{q_i}, \bar{q_i} - B_i)^2 & sinon \end{cases}} \tag{2.29}$$

$$= \sqrt{\frac{n}{N}\sum_{i=1}^{N} \max(H_i - \bar{q_i}, \bar{q_i} - B_i)^2} \tag{2.30}$$

Algorithme 2.1 *Algorithme de recherche séquentielle approximative.*

Entrées : Q[1..n], BDD[1..n_{db}][1..n], β

```
(L, U) ← enveloppe(Q)
pour i = 1..n_db faire
    lb ← LB(L, U, BDD[i])
    ub ← UB(L, U, BDD[i])
    lbub[i] ← lb + β (ub - lb)
fin pour

arg_lbub ← argsort(lbub)

meilleur_score ← ∞
indice_du_meilleur ← -1
pour i dans arg_lbub faire
    si meilleur_score ≤ lbub[i] alors
        sortir
    fin si
    si DTW(Q, BDD[i]) < meilleur_score alors
        meilleur_score ← DTW(Q, BDD[i])
        indice_du_meilleur ← i
    fin si
fin pour

renvoie (meilleur_score, indice_du_meilleur)
```

De même que MINDIST n'est pas une borne inférieure de la DTW, MAXDIST n'en est pas une borne supérieure.

Jusqu'à maintenant, nous avons, en partant de bornes inférieures exactes, défini des bornes supérieures exactes ayant même complexité de calcul. Nous avons proposé un moyen d'utiliser l'étendue de l'intervalle entre ces deux bornes pour fixer, en fonction de données d'apprentissage, un taux de distances surestimées acceptable. Étant donné un tel taux, nous avons défini des bornes inférieures approximatives qui permettent d'atteindre des valeurs d'ajustement plus élevées qu'avec les bornes inférieures exactes, au prix de possibles surestimations, lesquelles sont ensuite susceptibles d'induire des faux positifs.

Dans la suite, nous présentons les algorithmes permettant d'utiliser les bornes inférieures approximatives introduites et évaluons leur impact sur la recherche de séquences.

2.1.6 Algorithme de recherche séquentielle

Cette section présente un algorithme utilisant une borne inférieure approximative dans le but d'accélérer le processus de recherche séquentielle d'une séquence Q parmi un ensemble `BDD` de séquences. Cet algorithme fonctionne indifféremment que l'on se base sur LB_Yi, LB_Keogh ou LB_PAA.

On suppose ici que la base `BDD` contient n_{db} séquences échantillonnées pour être de taille n. Une phase d'apprentissage est effectuée hors-ligne pour apprendre une table de correspondance $\mathcal{C}$ entre valeurs de p et de β.

Au moment de la requête, l'utilisateur fournit une séquence Q de taille n et un taux p de distances surestimées (éventuellement calculé à partir de l'ajustement T voulu ou de la complexité souhaitée). p est utilisé pour interroger $\mathcal{C}$ et obtenir la valeur de β à utiliser.

La recherche séquentielle qui suit est décrite par l'algorithme 2.1. Il est important de remarquer que, en fixant une valeur β strictement positive, l'instruction de sortie de la boucle peut être déclenchée avant d'avoir trouvé le plus proche voisin exact. Cet algorithme peut trivialement être adapté au cas de la recherche de k-plus proches voisins : il suffit alors de maintenir un tableau des k meilleurs candidats à chaque instant, ainsi que leur similarité à la requête.

2.1.7 Algorithme basé sur un arbre de recherche

Cette section présente un algorithme qui utilise les bornes inférieures approximatives pour accélérer la recherche d'une séquence Q au sein d'une base de données organisée en un arbre d'indexation *i*SAX. Il repose sur l'algorithme précédemment présenté en ceci qu'il utilise la recherche séquentielle pour trouver la plus proche séquence au sein d'une feuille de l'arbre quand celle-ci est susceptible de contenir des séquences similaires à la requête.

L'indexation est effectuée hors-ligne et suit le principe énoncé dans [Camerra 10] pour *i*SAX 2.0, à la différence que, pour que MAXDIST puisse être calculé, chaque nœud est représenté par un véritable rectangle englobant, n'ayant que des côtés finis, comme expliqué en 2.1.5.

De la même manière que pour l'algorithme précédent, une phase d'apprentissage est effectuée hors-ligne pour construire une table de correspondance $\mathcal{C}$. Dans le cas de cet algorithme, cette table fournit, pour une valeur de p donnée, une paire $(\beta_{\text{LB\_Keogh}}, \beta_{\text{MINDIST}})$. Le premier élément de cette paire sera utilisé lors de l'appel à l'algorithme de recherche séquentielle alors que le second servira lors du calcul de MINMAXDIST. Ici encore, l'utilisateur fournit en entrée une séquence requête Q et un taux de distances surestimées p, qui permet d'obtenir les quantiles $\beta_{\text{LB\_Keogh}}$ et β_{MINDIST}) à partir de $\mathcal{C}$.

L'interrogation de l'index est détaillée dans l'algorithme 2.2. Notons que la fonction `recherche_approximative_isax`, qui est appelée pour initialiser le processus, renvoie la feuille de l'arbre d'indexation ayant une signature SAX qui correspond à celle de Q. Par construction de l'arbre, il existe une unique telle feuille.

Dans la suite, nous ferons référence à cette méthode sous la dénomination MINMAXDIST. Une autre façon de faire serait d'appliquer notre méthode d'approximation de borne inférieure au cas de la recherche approximative correspondant à la fonction `recherche_approximative_isax` et proposée dans [Keogh 05]. En effet, une fois le nœud approprié sélectionné, cette méthode effectue une recherche séquentielle parmi les séquences de ce nœud, laquelle peut être accélérée par l'utilisation d'une borne inférieure approximative. Dans la suite, nous noterons “LB Approximative” la recherche approximative issue de l'état-de-l'art et “LBUB Approximative” sa version

Algorithme 2.2 *Algorithme de recherche approximative utilisant iSAX.*

Entrées : Q[1..n], $\beta_{\text{LB\_Keogh}}$, β_{MINDIST}

```
(L, U) ← enveloppe(Q)
meilleur_nœud ← recherche_approximative_isax(Q)
(id_seq, dist_seq) ← scan_seq(Q, meilleur_nœud, β_LB_Keogh)
qp ← nouvelle_queue_de_priorité()
qp.insère(0, racine)

tant que qp n'est pas vide faire
   (dist_min, nœud_min) ← qp.minimum ()
   si dist_min ≥ dist_seq alors
      sortir
   fin si
   si nœud_min est une feuille alors
      (id, dist) ← scan_seq(Q, nœud_min, β_LB_Keogh)
      si dist_seq > dist alors
         dist_seq ← dist
         id_seq ← id
      fin si
   sinon
      sortir
      pour tout nœud fils de nœud_min faire
         mind ← MINDIST(Q, nœud)
         maxd ← MAXDIST(Q, nœud)
         dist ← mind + β_MINDIST (maxd - mind)
         qp.insère(dist, nœud)
      fin pour
   fin si
fin tant que

renvoie (id_seq, dist_seq)
```

utilisant des bornes inférieures approximatives.

2.1.8 Validation Expérimentale

Cette section présente l'évaluation expérimentale des bornes proposées. Leur ajustement est ici rapporté et nous détaillons leur efficacité pour le filtrage de séquences non similaires. Nous présentons également leur effet sur la qualité des résultats retournés, c'est-à-dire la quantité de faux négatifs engendrés. Toutes les expériences mettent en regard LB_Keogh et LBUB_Keogh. Certaines présentent également une comparaison de LB_PAA avec LBUB_PAA. Enfin, une dernière série d'expériences s'intéresse au cas particulier de *i*SAX et des bornes inférieures associées.

Dans un souci de reproductibilité des expériences, ces évaluations sont toutes

conduites en utilisant des jeux de données disponibles publiquement [Keogh 06] et largement utilisés [Keogh 05, Shieh 08]. Pour toutes ces expériences, la mesure de similarité de référence est la DTW contrainte par une bande de Sakoe-Chiba d'une largeur égale à 10% de la taille des séquences considérées, comme suggéré dans [Keogh 05].

Les expériences se basent sur deux jeux de données. Le premier, noté DS_1, est détaillé dans [Keogh 05] : il s'agit de 32 séries temporelles desquelles on extrait aléatoirement 50 sous-séquences de taille 256 par série temporelle pour générer le jeu de test. Le jeu d'apprentissage servant à construire la table de correspondance $\mathcal{C}$ est constitué en extrayant 100 sous-séquences de taille 256 par série temporelle.

Le second jeu de données, noté DS_2, est disponible à l'adresse [Keogh 06]. Il est constitué de 20 jeux de données, chacun étant divisé à l'avance entre jeu d'apprentissage et jeu de test. Quand nous utilisons ce jeu de données, nous utilisons naturellement les jeux d'apprentissage pour construire les tables $\mathcal{C}$. De plus, ces jeux de données fournissent une étiquette de classification pour chaque séquence. Cette étiquette est liée au comportement temporel des séquences. Par exemple, pour le jeu de données EEG, qui présente des séries temporelles liées à la mesure de l'activité cérébrale de rats, les trois classes correspondent à différents états des rats observés, éveillés ou dans l'une ou l'autre des phases de sommeil étudiées. Quand nous évaluons la qualité de la classification, nous utilisons le schéma simpliste où l'étiquette associée à une séquence est celle de son plus proche voisin retourné par l'algorithme de recherche.

Pour les deux jeux de données présentés plus haut et pour LBUB_Keogh comme pour LBUB_PAA, nous rapportons le taux de 1-plus-proches-voisins (1-PPV) corrects, qui est la proportion des requêtes pour lesquelles leur véritable plus proche voisin (au sens de la DTW) est classé en première position.

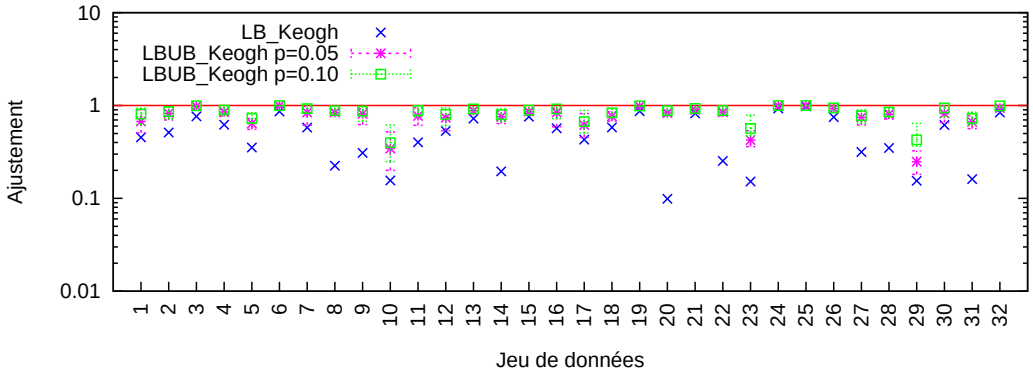
Lorsque c'est nécessaire, les médianes des estimateurs sont fournies avec leur intervalle inter-quartiles.

Ajustement comparé des bornes inférieures

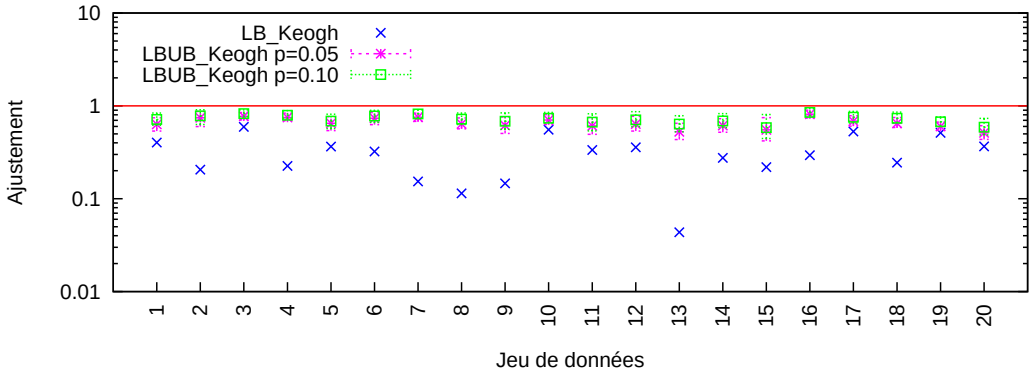
La figure 2.5 présente une comparaison des ajustements obtenus par LB_Keogh et LBUB_Keogh en utilisant les jeux de données DS_1 et DS_2. Le tableau 2.2 présente quelques unes des valeurs issues de cette figure, pour une évaluation plus quantitative. On peut remarquer que l'ajustement pour LB_Keogh varie de manière significative d'une expérience à l'autre : par exemple, pour ce qui est de DS_2, on observe que LB_Keogh est très ajusté pour les expériences #3 et #10, alors que pour les expériences #8 et #13, l'ajustement est très mauvais. Au contraire, l'ajustement de LBUB_Keogh est toujours plus proche de 1. La faible étendue des intervalles inter-quartiles montrent la consistence des valeurs d'ajustement observées pour l'ensemble des paires de séquences considérées. On remarque enfin que, comme prévu par la théorie, plus la valeur de p est grande, plus l'ajustement obtenu est grand.

Similarité des listes ordonnées

L'algorithme 2.1 trie les séquences de la base par ordre croissant de LBUB_Keogh. Cet ordre dépend de la valeur donnée à p. Dans tous les cas, il peut différer de l'ordre



(a) DS_1



(b) DS_2

FIGURE 2.5 – *Ajustements comparés de LB_Keogh et de LBUB_Keogh.* Les médianes d'ajustement sont restituées avec leur intervalle inter-quartiles.

Jeu de données	Méthode	Ajustement médian	Intervalle inter-quartiles de l'ajustement
DS_2 #3	LB_Keogh	0,616	[0,463 ;0,738]
	LBUB_Keogh p =0,05	0,773	[0,706 ;0,848]
	LBUB_Keogh p =0,1	0,826	[0,765 ;0,905]
DS_2 #8	LB_Keogh	0,097	[0,052 ;0,152]
	LBUB_Keogh p =0,05	0,640	[0,567 ;0,740]
	LBUB_Keogh p =0,1	0,719	[0,630 ;0,837]
DS_2 #10	LB_Keogh	0,577	[0,443 ;0,667]
	LBUB_Keogh p =0,05	0,705	[0,630 ;0,805]
	LBUB_Keogh p =0,1	0,736	[0,655 ;0,849]
DS_2 #13	LB_Keogh	0,014	[0,002 ;0,060]
	LBUB_Keogh p =0,05	0,526	[0,433 ;0,649]
	LBUB_Keogh p =0,1	0,636	[0,525 ;0,783]

TABLEAU 2.2 – *Ajustements comparés de LB_Keogh et de LBUB_Keogh pour quelques jeux de données.*

Jeu de données	Méthode	Coefficient de corrélation médian	Intervalle inter-quartiles du coefficient de corrélation
DS_1	LB_Keogh	0,966	[0,874 ;0,996]
	LBUB_Keogh p =0,05	0,967	[0,865 ;0,997]
	LBUB_Keogh p =0,1	0,967	[0,861 ;0,997]
	LB_PAA	0,917	[0,501 ;0,991]
	LBUB_PAA p =0,05	0,927	[0,653 ;0,993]
	LBUB_PAA p =0,1	0,926	[0,653 ;0,993]
DS_2	LB_Keogh	0,914	[0,820 ;0,982]
	LBUB_Keogh p =0,05	0,914	[0,819 ;0,982]
	LBUB_Keogh p =0,1	0,915	[0,819 ;0,982]
	LB_PAA	0,614	[0,310 ;0,875]
	LBUB_PAA p =0,05	0,661	[0,194 ;0,874]
	LBUB_PAA p =0,1	0,663	[0,181 ;0,867]

TABLEAU 2.3 – *Coefficient de corrélation de Spearman pour LBUB_Keogh et LBUB_PAA.*

obtenu en triant les séquences par ordre croissant de DTW. De plus, lorsque p est strictement positif, l'instruction de sortie peut être déclenchée avant d'avoir identifié le plus proche voisin exact. Dans ce cas, il est crucial d'avoir des listes de résultats proches de celles retournées obtenues avec la DTW car, alors, une distance surestimée n'entraine pas forcément un faux négatif (si les listes ordonnées sont identiques, il est même impossible d'observer un faux négatif).

Le coefficient de corrélation de Spearman, noté ρ, est une mesure largement utilisée de la similarité entre deux listes ordonnées. ρ prend ses valeurs dans l'intervalle $[0, 1]$ et, plus il est proche de 1, plus les listes sont similaires. Il est défini par :

$$\rho = 1 - \frac{6 \sum_{i=1}^{n_{\text{db}}} (r_i^1 - r_i^2)^2}{n_{\text{db}}(n_{\text{db}}^2 - 1)}, \tag{2.31}$$

où r_i^1 et r_i^2 sont les rangs de la séquence d'indice i dans chacune des deux listes.

Le tableau 2.3 présente les valeurs médianes de ρ calculées sur les jeux de données DS_1 et DS_2. Dans tous les cas, l'ensemble des r_i^1 est obtenu grâce à la DTW alors que les r_i^2 résultent de la borne inférieure considérée. On remarque tout d'abord que, dans tous les cas, ρ ne varie que peu en fonction de p, pour p compris entre 0 et 0,1.

Un autre point important à remarquer est que, pour LB_PAA (respectivement LBUB_PAA), les listes ordonnées obtenues sont beaucoup moins pertinentes que pour LB_Keogh (respectivement LBUB_Keogh). Cela implique que de petites valeurs de p sont à priviligier lorsque l'on essaye d'approximer LB_PAA, car chaque surestimation aura une probabilité plus grande de provoquer un faux négatif.

Métrique	p	Coût computationnel	Taux de 1-PPV corrects	Taux de classifications correctes
LB_Keogh	—	1,000	1,000	0,918
LBUB_Keogh	0,01	0,165	0,715	0,913
	0,05	0,062	0,410	0,879
	0,10	0,042	0,315	0,853
LB_PAA	—	1,000	1,000	0,918
LBUB_PAA	0,001	0,770	0,942	0,918
	0,01	0,370	0,603	0,906
	0,05	0,132	0,243	0,861

TABLEAU 2.4 – *Pertinence des résultats et coût computationnel de LBUB_Keogh et LBUB_PAA.* Dans tous les cas, la borne inférieure exacte est correspondante est utilisée comme référence pour le coût computationnel.

Pertinence des résultats et coût computationnel

Nous utilisons DS_2 pour évaluer les performances des bornes inférieures que nous proposons à la fois en termes de qualité des résultats et de coût, respectivement évalués par le taux de 1-PPV corrects et la proportion des calculs de DTW induits par la borne inférieure approximative parmis les calculs nécessaires dans le cas de l'utilisation d'une borne inférieure exacte. Ainsi, en fixant $p = 0$, on obtient un coût computationnel de 1.

Le tableau 2.4 présente ces résultats pour LBUB_Keogh et LBUB_PAA. On peut tout d'abord remarquer que, dans tous les cas, le coût computationnel décroît rapidement avec p. De plus, comme suggéré plus haut, il est préférable d'utiliser de faibles valeurs de p pour LBUB_PAA à cause de la mauvaise qualité des listes retournées. On remarque aussi que le taux de classifications correctes décroît beaucoup plus lentement que le taux de 1-PPV corrects. Cela montre que, si le premier plus proche voisin n'est pas classé premier, la séquence qui est classée en première position est souvent de la bonne classe, ce qui est important. En effet, l'une des motivations principales pour l'utilisation de méthodes de recherche approximatives dans le domaine de l'indexation en grande dimension est que, lorsque l'on considère des grands volumes de données, la plupart des objets les mieux classés peuvent être considérés comme similaires et le fait de ne retourner qu'un sous-ensemble des objets les plus similaires est acceptable pour une large gamme d'applications si cela permet de diminuer significativement le coût de calcul.

*i*SAX

Dans cette section, nous nous intéressons aux performances de notre algorithme de recherche approximative pour *i*SAX. Pour ce faire, nous utilisons deux ensembles, l'un pour l'apprentissage et l'autre pour les tests. Chaque ensemble est constitué de

p	Méthode	Coût computationnel	1-PPV	10-PPV	50-PPV
—	MINDIST	1,0000	0,789	0,989	1,000
—	LB Approximative	0,0342	0,127	0,545	0,856
0,01	MINMAXDIST	0,0036	0,146	0,615	0,900
0,05	MINMAXDIST	0,0006	0,045	0,290	0,644
0,1	MINMAXDIST	0,0004	0,037	0,233	0,564
0,01	LBUB Approximative	0,0005	0,060	0,312	0,765
0,05	LBUB Approximative	0,0003	0,034	0,188	0,520
0,1	LBUB Approximative	0,0002	0,031	0,178	0,487

TABLEAU 2.5 – *Performances de la recherche approximative pour iSAX.* Ces performances sont évaluées en termes de qualité des résultats et de coût computationnel.

1000 requêtes qui sont des marches aléatoires de taille 256 telles que :

$$\forall i \in [\![1; 1000]\!], \quad (q_i)_1 = \Delta_{i,1} \tag{2.32}$$

$$\forall (i,j) \in [\![1; 1000]\!] \times [\![2; 256]\!], \quad (q_i)_j = (q_i)_{j-1} + \Delta_{i,j} \tag{2.33}$$

où

$$(\Delta_{i,j}) \sim \mathcal{N}(0,1). \tag{2.34}$$

Chaque ensemble contient également un arbre qui stocke 100 000 séquences ayant les mêmes caractéristiques que les requêtes. Chaque nœud de l'arbre peut stocker jusqu'à 800 séquences. Les arbres sont construits en utilisant l'algorithme *i*SAX2.0 qui permet un meilleur équilibrage de la cardinalité des feuilles. Une vérité terrain est calculée en termes de DTW pour chaque paire requête-séquence de la base. Notons ici que l'algorithme de recherche exacte proposé dans [Shieh 08] et dont s'inspire notre algorithme de recherche approximative ne garantit pas l'absence de faux négatif, puisque MINDIST est un minorant de la distance euclidienne mais pas de la DTW. Cela implique qu'aucune des méthodes considérées ne puisse garantir un taux de 1-PPV corrects de 1. C'est pourquoi nous rapportons également ici le taux de 10-PPV (respectivement 50-PPV) qui est la proportion des séquences retournées faisant partie des 10 (respectivement 50) plus proches voisins exacts de la requête.

Les résultats présentés en tableau 2.5 montrent la supériorité de notre proposition comparée aux méthodes de l'état-de-l'art. Par exemple, la recherche approximative proposée dans [Shieh 08], qui est notée "LB Approximative" dans le tableau, a recours à 10 fois plus de calculs de DTW que notre méthode, notée MINMAXDIST, avec $p = 0,01$, tout en étant toujours moins bonne en termes de qualité des résultats.

On peut également noter que, si l'on souhaite atteindre des coûts computationnels particulièrement bas, il est intéressant de considérer "LBUB Approximative" – qui est une variante de l'algorithme de recherche approximative proposé dans [Shieh 08] pour laquelle LBUB_Keogh est utilisée à la place de LB_Keogh – car, si l'on fixe p =0,01, le coût computationnel obtenu est plus faible que pour MINMAXDIST avec p =0,05, tout en permettant une meilleure qualité des résultats.

2.2 kPAA

Une caractéristique importante de tout arbre d'indexation est son équilibre. En effet, l'espérance du nombre de calculs exacts de similarité est proportionnel au déséquilibre de l'arbre utilisé. C'est pourquoi Camerra *et al.* [Camerra 10] ont introduit dans leur algorithme une heuristique ayant pour but d'équilibrer les feuilles des arbres d'indexation produits par iSAX. Nous proposons ici l'utilisation du célèbre algorithme de classification non supervisée k-means, présenté au chapitre précédent, pour générer un arbre ayant une structure similaire à celle d'un arbre produit par iSAX. La méthode proposée, notée kPAA, permet une meilleure adaptation de la structure aux données que dans le cas de iSAX qui suppose une distribution des données normale centrée réduite. De plus, notre méthode génère des arbres plus équilibrés que lorsque l'on utilise iSAX 2.0 et présente l'avantage d'être paramétrable.

2.2.1 Utilisation de k-means pour l'indexation de séquences

Une hypothèse sur laquelle est basé iSAX est que les représentations PAA des séquences suivent une distribution normale centrée réduite. Pour s'en dispenser, nous proposons d'utiliser l'algorithme de k-means afin d'apprendre des frontières entre nœuds de l'arbre qui dépendent des données.

Présentation de k-means

Nous utilisons ici un quantificateur vectoriel largement répandu (c'est par exemple celui qui est utilisé dans [Sivic 03]), à savoir le k-means.

Le k-means est un algorithme de classification non supervisée. Il prend en entrée un ensemble de points $\mathbf{x}$ et un nombre de classes à former k et propose une heuristique visant à minimiser la variance intra-classe définie par :

$$V = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{c_i}\|^2 \tag{2.35}$$

où $\mathbf{c_i}$ est le centre de gravité de l'ensemble des points associés à la classe C_i, également appelé *centroïde* de la classe C_i.

L'algorithme de k-means est itératif et chaque itération est décomposée en deux phases, sur le modèle des algorithmes dits *Expectation Maximisation* (EM). La première de ces deux phases consiste à associer chacun des points à classifier à son plus proche centroïde. La seconde phase se résume à mettre à jour la position des centroïdes en fonction du nouvel ensemble de points qui leur est associé. On peut montrer que chacune de ces deux phases fait décroître la variance intra-classe et le processus itératif converge donc vers un minimum local de variance intra-classe.

L'initialisation du processus, c'est-à-dire la position initiale à attribuer aux centroïdes, est prépondérante pour obtenir un minimum local de bonne qualité, c'est-à-dire qui soit proche du minimum global. Un procédé largement utilisé pour l'initialisation des centroïdes consiste à tirer aléatoirement des points issus de l'ensemble à classifier. À chaque nouveau centroïde retenu, on met à jour la probabilité

d'être tiré pour chacun des points de l'ensemble non encore sélectionnés comme :

$$p(\mathbf{x}) \propto \min_{\mathbf{c_i}} d(\mathbf{x}, \mathbf{c_i}). \tag{2.36}$$

Ce processus, appelé kmeans++ a été proposé par Arthur et Vassilvitskii [Arthur 07].

Il est important de remarquer que la quantité V que l'on cherche à minimiser peut être vue, dans un contexte de quantification, comme la somme des erreurs de quantifications obtenues en approximant chaque point par le centroïde de la classe à laquelle il est associé. En d'autres termes, pour notre application, un découpage de l'espace en k classes qui minimiserait V serait celui qui garantirait une représentation optimale des points proposés sous la contrainte de n'utiliser que k symboles. Si k-means ne garantit qu'un optimum local, les centroïdes obtenus sont de bons représentants pour les données, comme en témoignent leur utilisation fréquente pour ce type de tâches.

k-means et kPAA

Pour conserver une structure similaire à celle d'un arbre issu de iSAX, nous proposons d'effectuer un k-means hiérarchique, le premier niveau de séparation correspondant à un k-means utilisant $k = w^b$. Les sous-arbres issus des nœuds fils de la racine étant des arbres binaires, les séparations correspondantes sont des k-means pour lesquels $k = 2$.

De plus, alors que l'algorithme iSAX construit l'arbre d'indexation en insérant successivement chacune des séquences dans l'arbre et en rajoutant des nœuds lorsque cela s'avère nécessaire, nous proposons de commencer par stocker toutes les séquences dans la racine et d'effectuer les scissions successives jusqu'à avoir, pour chaque feuille de l'arbre, au plus th séquences associées. Il est ensuite possible d'ajouter des séquences à l'arbre obtenu de manière similaire à ce qui est fait pour iSAX, à la différence près que si l'on a une division à faire parce que l'insertion d'un point nous amène à dépasser la capacité de la feuille considérée, cette division se fera à l'aide d'un k-means.

2.2.2 Équilibrage d'un arbre kPAA

La méthode proposée dans la section précédente permet d'améliorer la représentation des séquences au sein de chacun des nœuds tout en conservant une structure similaire à celle d'iSAX. Nous présentons ici deux méthodes pour équilibrer les classes obtenues par k-means, l'une générique et l'autre spécifique au cas $k = 2$.

Problématique de l'équilibrage des classes de k-means

Comme proposé dans [Jégou 10], nous mesurons le déséquilibre entre les cardinalités des classes issues de k-means par un facteur de déséquilibre γ défini par :

$$\gamma = k \sum_{i=1}^{k} {p_i}^2 \tag{2.37}$$

où p_i est la probabilité d'appartenance à la classe C_i, qui est mesurée en pratique par la fraction des points associés à la classe C_i parmi l'ensemble des points.

[Jégou 10] montre que, pour une valeur de k fixée et dans le cas où, pour une requête, on ne s'intéresse qu'aux éléments de la classe à laquelle elle est associée, le facteur de déséquilibre est en lien direct avec la complexité de la recherche : si l'on a $\gamma = 3$, cela signifie que l'espérance du temps de calcul sera multipliée par 3 par rapport au cas où les classes sont parfaitement équilibrées. C'est pour cette raison que le facteur de déséquilibre nous semble plus approprié dans ce cas que l'entropie, et ce même si les deux métriques atteignent un extremum dans le cas de classes parfaitement équilibrées.

L'équilibrage optimal ($\gamma = 1$) est atteint lorsque $|C_i| = N/k$ pour tout i. Dans ce cas, la variance des temps de requête est nulle, comme le montre la formule suivante :

$$\text{Var} \propto N^2 \sum_{i=1}^{k} p_i \left(p_i - \frac{1}{k} \right)^2 . \tag{2.38}$$

Cas du premier niveau

Nous proposons une procédure d'équilibrage des classes qui s'effectue après convergence de l'algorithme de k-means. L'idée est d'augmenter artificiellement les distances entre les points de données et les centroïdes des classes les plus peuplées dans le but de rétrécir ces classes pour en diminuer la population. Les pénalités appliquées aux distances dépendent de la population des classes considérées. À chaque itération, le contenu des classes et la position de leur centroïde sont recalculés. Les pénalités sont calculées comme suit :

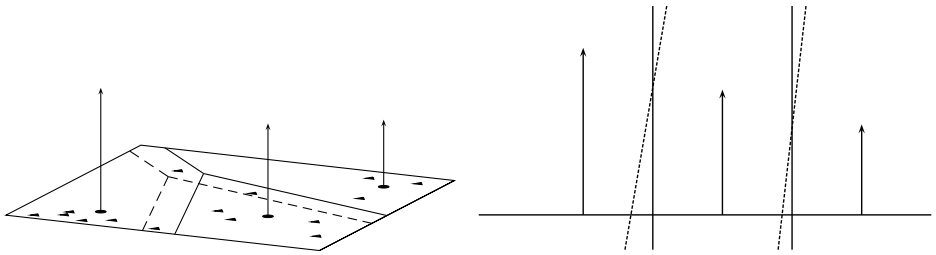
$$\begin{cases} \forall i, h_i^0 = 1 \\ \forall l \leq r, h_i^{l+1} = h_i^l \left(\frac{n_i^l}{n_{\text{opt}}} \right)^\alpha \end{cases} \tag{2.39}$$

où α contrôle la vitesse de convergence et r est le nombre d'itérations effectuées. Une petite valeur pour α permet de garantir que l'équilibrage se fera sans perturber trop fortement la structure des classes générées par l'algorithme de k-means. L'équilibrage nécessitera alors un nombre d'itérations plus grand que pour des valeurs de α plus grandes. Il est important de noter qu'à chaque itération l, les populations $|C_i^l|$ sont mises à jour en prenant en compte les pénalités. Plus précisément, la distance d'un point $\mathbf{x}$ au $i^{\text{ième}}$ centroïde sera calculée en utilisant la formule suivante :

$$d_{\text{bal}}^l(\mathbf{x}, \mathbf{c_i})^2 = d(\mathbf{x}, \mathbf{c_i})^2 + h_i^l. \tag{2.40}$$

La valeur initiale h_i^0 s'explique par cette dernière équation, puisque cela permet d'assigner à l'équilibrage une énergie similaire à celle des autres dimensions des données, sous l'hypothèse de vecteurs normés.

Cette stratégie permet de converger en pratique vers des classes équilibrées. Il est possible de définir plusieurs critères d'arrêt. Les deux critères d'arrêt les plus naturels sont, d'une part, un nombre fixé d'itérations et, d'autre part, un objectif d'équilibrage, exprimé en terme de valeur de γ à atteindre. Notons qu'il peut s'avérer judicieux d'arrêter le processus avant d'avoir atteint un équilibrage optimal, dans la mesure où cet équilibrage se fait au prix d'une distortion des régions de Voronoï initiales.



(a) (b)

FIGURE 2.6 – *Interprétation graphique de l'équilibrage des classes de k-means.* En (a), points de données et centroïdes sont embarqués dans un espace de dimension 3, alors que la dimension de l'espace d'origine est 2. Les données sont représentées par des triangles et les centroïdes sont marqués par des points, avec une coordonnée sur l'axe z non nulle après quelques itérations. Les régions de Voronoï initiales sont délimitées par des lignes pleines alors que les versions translatées de ces frontières, résultant de l'élévation des centroïdes, sont tracées en trait pointillé. La figure (b) montre une vue en coupe de la même situation explicitant le déplacement impliqué sur les frontières par l'élévation des centroïdes.

Interprétation géométrique

Le processus d'équilibrage embarque les vecteurs de dimension dim classifiés par le k-means dans un espace de dimension $(\dim + 1)$. Dans cet espace, les dim premières composantes de ces vecteurs restent inchangées tandis que la $(\dim + 1)^{\text{ième}}$ composante est fixée à 0. Les centroïdes sont embarqués de manière similaire à l'exception de leur dernière composante dont la valeur est fixée à $\sqrt{h_i^0}$. Ensuite, au fur et à mesure des itérations de notre algorithme, cette dernière est mise à jour à la valeur $\sqrt{h_i^l}$ appropriée. L'intuition derrière cela est que les centroïdes sont élevés itérativement de l'hyperplan dans lequel se trouvent les données. Plus une classe est peuplée, plus son centroïde sera éloigné de cet hyperplan.

Cela est illustré par la figure 2.6 où l'axe z correspond à la dimension introduite par notre algorithme. Au fur et à mesure des itérations, les assignations des points sont mises à jour en fonction des coordonnées dans l'espace augmenté. L'élévation artificielle des centroïdes tend à rétrécir les classes les plus peuplées, répartissant leurs points dans les classes voisines moins peuplées.

Il est important de remarquer que les nouvelles frontières des régions de Voronoï sont parallèles aux frontières initiales à chaque étape. On peut le montrer en considérant la définition suivante pour la frontière de Voronoï entre les classes i et j à l'itération l :

$$H_{i,j} = \left\{\mathbf{x}, f_{i,j}(\mathbf{x}) + h_i^l - h_j^l = 0\right\} \tag{2.41}$$

$$f_{i,j}(\mathbf{x}) = d(\mathbf{x}, \mathbf{c_i})^2 - d(\mathbf{x}, \mathbf{c_j})^2 \tag{2.42}$$

où d est la distance dans l'espace de dimension dim et où h_i^l et h_j^l sont indépendants de $\mathbf{x}$. Cela implique que le vecteur $\left(\frac{\partial f_{i,j}}{\partial x_1}, \cdots, \frac{\partial f_{i,j}}{\partial x_{\dim}}\right)$ est orthogonal à $H_{i,j}$. Celui-ci

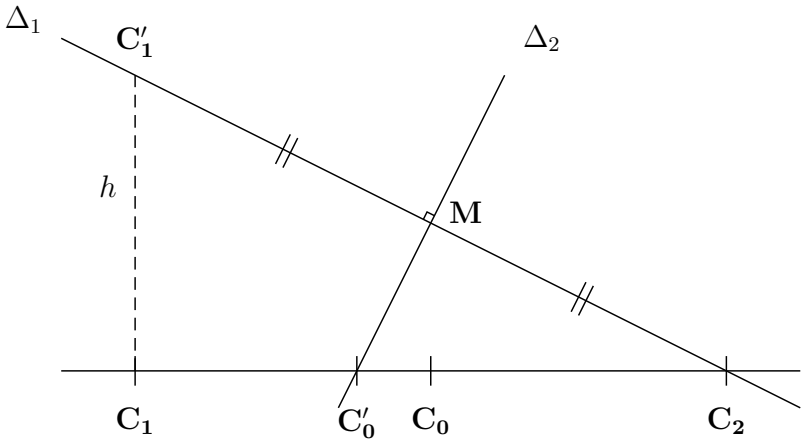


FIGURE 2.7 – *Élévation des centroïdes de k-means dans le cas* $k = 2$.

étant également normal à la frontière initiale, les deux frontières sont parallèles.

Cas des autres niveaux

Chacun des nœuds d'un arbre *i*SAX (hormis la racine dont le cas peut être considéré par la proposition précédente), s'il doit être subdivisé, aura deux fils. Ainsi, le k-means utilisé par kPAA utilisera $k = 2$.

Il est possible, dans ce cas, d'exprimer l'élévation h à appliquer au centroïde le plus peuplé des deux pour que les classes résultantes soient parfaitement équilibrées.

Nous supposons dans la suite, sans perte de généricité, que le k-means a engendré deux centroïdes, $\mathbf{C_1}$ et $\mathbf{C_2}$ et que la classe C_1 est plus peuplée que la classe C_2 et nous basons notre étude sur les notations introduites en figure 2.7. L'intersection entre la droite $(\mathbf{C_1}, \mathbf{C_2})$ et la frontière entre les classes C_1 et C_2 se trouve alors en $\mathbf{C_0}$, milieu du segment $[\mathbf{C_1}, \mathbf{C_2}]$. On cherche à évaluer l'élévation h nécessaire pour que cette frontière se déplace en $\mathbf{C'_0}$, qui est le point médian parmi les projections des points des deux classes. En effet, en utilisant une frontière parallèle à celle définie par le k-means et passant par $\mathbf{C'_0}$, on aura bien des classes équilibrées. Notons que si l'on souhaite atteindre une certaine valeur pour γ, il est possible de définir le quantile à utiliser en lieu et place de la médiane et le reste du raisonnement reste valable.

On obtient le système d'équations suivant :

$$\begin{cases} x_M & = & \frac{x_1+x_2}{2} \\ y_M & = & \frac{h}{2} \\ 0 & = & a_{\Delta_2} x'_0 + b_{\Delta_2} \\ y_M & = & a_{\Delta_2} x_M + b_{\Delta_2} \\ a_{\Delta_1} a_{\Delta_2} & = & -1 \\ h & = & a_{\Delta_1} x_1 + b_{\Delta_1} \\ 0 & = & a_{\Delta_1} x_2 + b_{\Delta_1} \end{cases} \tag{2.43}$$

La résolution de ce système permet de lier h à x'_0, qui est l'intersection entre la nouvelle frontière inter-classes et la droite $(\mathbf{C_1}, \mathbf{C_2})$:

$$h = \sqrt{2(x_2 - x_1)\left(\frac{x_1 + x_2}{2} - x'_0\right)}. \tag{2.44}$$

Arbre	Facteur de déséquilibre γ	
	Arbre de test	Arbre d'apprentissage
iSAX 2.0	1,852	1,855
kPAA	1,040	1,036

TABLEAU 2.6 – *Équilibrages comparés des arbres iSAX et kPAA.*

p	Méthode	Coût computationnel	1-PPV	10-PPV	50-PPV
—	MINDIST	1,0000	0,789	0,989	1,000
—	LB Approximative	0,0342	0,127	0,545	0,856
0,01	MINMAXDIST	0,0036	0,146	0,615	0,900
0,05	MINMAXDIST	0,0006	0,045	0,290	0,644
0,1	MINMAXDIST	0,0004	0,037	0,233	0,564
0,01	LBUB Approximative	0,0005	0,060	0,312	0,765
0,05	LBUB Approximative	0,0003	0,034	0,188	0,520
0,1	LBUB Approximative	0,0002	0,031	0,178	0,487

(a) Rappel des performances de iSAX 2.0

p	Méthode	Coût computationnel	1-PPV	10-PPV	50-PPV
—	MINDIST	3,3168	1,000	1,000	1,000
—	LB Approximative	0,0822	0,486	0,972	1,000
0,01	MINMAXDIST	0,0006	0,100	0,564	0,991
0,05	MINMAXDIST	0,0003	0,056	0,278	0,667
0,1	MINMAXDIST	0,0002	0,053	0,259	0,617
0,01	LBUB Approximative	0,0006	0,100	0,564	0,991
0,05	LBUB Approximative	0,0003	0,056	0,278	0,667
0,1	LBUB Approximative	0,0002	0,053	0,259	0,617

(b) Performances de kPAA

TABLEAU 2.7 – *Performances comparées de la recherche approximative pour kPAA et iSAX2.0.* Ces performances sont évaluées en termes de qualité des résultats et de coût computationnel (la référence utilisée pour le coût computationnel est le coût pour iSAX 2.0 utilisant MINDIST).

2.2.3 Validation expérimentale

Les expériences presentées ici sont effectuées sur les mêmes données que celles utilisées en section 2.1.8, dans le but de permettre la comparaison directe entre iSAX 2.0 et kPAA.

Le tableau 2.6 présente, pour chaque méthode, les facteurs de déséquilibre observés sur les deux arbres utilisés respectivement pour l'apprentissage et le test. On note que, pour un type d'arbre fixé, les valeurs sont assez stables. Par contre, les arbres kPAA obtiennent des valeurs de γ fortement plus basses que les arbres iSAX 2.0, la différence étant de l'ordre de 45%.

Partant de cette observation, il semblerait logique d'observer des coûts de calcul plus faibles pour chacune des méthodes de recherche introduites plus haut. Or, le

tableau 2.7 montre que, pour les méthodes MINDIST et LB Approximative, les coûts sont plus important que ceux observés en tableau 2.5, pour *i*SAX. Il est important de rappeler que, pour ces deux méthodes, la recherche séquentielle au sein d'une feuille de l'arbre se fait en utilisant LB_Keogh. Ainsi, le nombre de calculs de DTW n'est pas forcément proportionnel au nombre de séquences stockées dans la feuille considérée, ce qui explique qu'alors, γ ne soit pas un indicateur fiable de la quantité de calculs de DTW effectués à chaque requête. On remarque par contre que les performances en termes de qualité des résultats retournés sont significativement meilleurs, ce qui valide l'intérêt d'apprendre des frontières entre nœuds de l'arbre qui dépendent des données plutôt que de se fier aux tables de la distribution normale centrée réduite. Si l'on compare maintenant les méthodes utilisant MINMAXDIST en se fixant un coût (on peut par exemple se concentrer sur p =0,05 pour *i*SAX et p =0,01 pour *k*PAA qui ont le même coût), on remarque que les gains en termes de qualité des résultats sont encore une fois élevés, puisqu'on a, pour le score relatif au 1-PPV par exemple, un taux de vrais positifs de 10% au lieu de 4,5% pour *i*SAX. Enfin, on remarque dans ce tableau un artefact qui peut paraître surprenant : les scores et coûts de MINMAXDIST et de LBUB Approximative sont exactement égaux. Cela s'explique par le fait que, pour les valeurs de p rapportées, l'algorithme de recherche se limite toujours à la première feuille proposée, car les autres ont une valeur de MINMAXDIST plus élevée que la séquence la plus similaire issue de la première feuille.

2.3 Bilan

Nous avons présenté ici des méthodes permettant, par l'utilisation d'approximations, de gagner plusieurs ordres de grandeur en termes de coût de calcul tout en retournant des résultats de meilleur qualité que le seul algorithme approximatif de l'état-de-l'art se penchant sur l'indexation de séquence à l'aide de DTW. Nous avons également montré que l'utilisation de l'algorithme de k-means lors de la construction de l'arbre d'indexation permet d'obtenir des arbres mieux adaptés aux données et donc d'améliorer, à coût constant, la qualité des résultats.

Ces résultats encourageants et la suppression des hypothèses sur la distribution des données considérées permettent d'envisager l'utilisation de telles méthodes pour l'indexation de séquences multidimensionnelles naturelles, ce qui constitue une perspective majeure de ce travail. En effet, il est reconnu (voir notamment [Berrani 04]) que les approches utilisant des rectangles englobants ne sont pas utilisables pour l'indexation dans des espaces de grandes dimensions et qu'alors des méthodes comme le k-means sont plus adaptées.

Bibliographie

[Arthur 07] D. Arthur & S. Vassilvitskii. *k-means++ : the advantages of careful seeding*. In Proceedings of the annual ACM-SIAM symposium on Discrete algorithms, pages 1027–1035, 2007.

[Berrani 04] S. A. Berrani. *Recherche approximative de plus proches voisins avec contrôle probabiliste de la précision ; application à la recherche d'images par le contenu*. PhD thesis, Université de Rennes 1, 2004.

[Camerra 10] A. Camerra, T. Palpanas, J. Shieh & E. J. Keogh. *iSAX 2.0 : Indexing and Mining One Billion Time Series*. In Proceedings of the IEEE International Conference on Data Mining, 2010.

[Faloutsos 94] C. Faloutsos, M. Ranganathan & Y. Manolopoulos. *Fast Subsequence Matching in Time-series Databases*. In Proceedings of the ACM SIGMOD Conference, pages 419–429, 1994.

[Jégou 10] H. Jégou, M. Douze & C. Schmid. *Improving bag-of-features for large scale image search*. International Journal of Computer Vision, vol. 87, no. 3, pages 316–336, 2010.

[Keogh 05] E. Keogh & C.A. Ratanamahatana. *Exact Indexing of Dynamic Time Warping*. Knowledge And Information Systems, vol. 7, no. 3, pages 358–386, 2005.

[Keogh 06] E. Keogh, X. Xi, L. Wei & C. A. Ratanamahatana. *The UCR Time Series Classification/Clustering Homepage : www.cs.ucr.edu/~eamonn/time_series_data/*, 2006.

[Ratanamahatana 04] C.A. Ratanamahatana & E. Keogh. *Everything you know about dynamic time warping is wrong*. In Proceedings of the Workshop on Mining Temporal and Sequential Data, 2004.

[Shieh 08] J. Shieh & E. Keogh. *iSAX : Indexing and Mining Terabyte Sized Time Series*. In Proceedings of the ACM International Conference on Knowledge Discovery and Data mining, pages 623–631, 2008.

[Sivic 03] J. Sivic & A. Zisserman. *Video Google : a Text Retrieval Approach to Object Matching in Videos*. In Proceedings of the International Conference on Computer Vision, volume 2, pages 1470–1477, 2003.

[Tavenard 11] R. Tavenard, H. Jégou & L. Amsaleg. *Balancing clusters to reduce response time variability in large scale image search*. In IEEE Workshop on Content-Based Multimedia Indexing, 2011.

Chapitre 3

Recherche de sous-séquences communes : application aux reprises de chansons

Le chapitre précédent propose une approche intégrant une information temporelle forte dans la comparaison de séquences entières. Nous nous focalisons ici sur le cas où l'on ne peut pas aligner l'intégralité des séquences entre elles mais où l'on souhaite trouver des sous-séquences communes entre une base et une requête.

Ce type de problème est exemplaire d'applications comme la recherche de reprises de chansons, que nous abordons dans ce chapitre. Une reprise est l'adaptation d'une chanson par un musicien qui n'en était pas l'interprète original. Il est fréquent que les reprises soient effectuées dans une tonalité différente de la tonalité originale. Cela peut s'expliquer par des motivations artistiques ou pour des raisons plus pragmatiques d'adaptation de la chanson à la voix du nouvel interprète. De plus, ce type d'adaptation implique souvent des changements dans la structure des morceaux, certains parties pouvant être déplacées au sein du morceau, supprimées ou dupliquées. La similarité entre deux versions d'une même chanson s'exprimera donc en termes de sous-parties communes. Enfin, même à l'intérieur de ces sous-parties, la similarité n'est pas parfaite. Il est par exemple fréquent que, là où l'interprète original ne jouait qu'un accord, le musicien qui l'adapte y rajoute des arpèges, et *vice versa*.

Traditionnellement dans ce type de problèmes, une chanson est décrite par une séquence de vecteurs de chromas, qui capturent l'information harmonique du signal audio en indiquant l'énergie dans le signal correspondant à chaque demi-ton de la gamme chromatique. Les bases de données considérées son structurées puisqu'elles consistent en un ensemble de séquences, chacune correspondant à une chanson. Étant donnée une chanson requête décrite elle aussi sous la forme d'une séquence de vecteurs de chromas, on recherche les séquences de la base ayant des sous-parties similaires à une passage de la requête.

La principale méthode de l'état-de-l'art traitant de cette problématique consiste

en une comparaison coûteuse, dérivée de l'alignement dynamique, de chacune des séquences de la base avec la séquence requête. Aucune méthode d'indexation n'est fournie, ce qui limite la possibilité d'une utilisation à grande échelle. De plus, les méthodes d'indexation présentées au chapitre précédent ne sont pas utilisables dans ce cadre puisqu'elles cherchent à aligner les séquences dans leur ensemble, alors que nous cherchons ici à trouver des sous-séquences communes.

Nous proposons dans ce chapitre une approche qui se décompose en deux étapes. Tout d'abord, pour permettre le passage à l'échelle, nous suggérons d'utiliser une recherche locale de plus proches voisins, n'intégrant que peu d'information temporelle, qui permettra d'identifier les séquences de la base les plus susceptibles de correspondre à la requête. Dans un deuxième temps, pour améliorer la pertinence des résultats, nous utilisons les similarités locales extraites à la première étape pour effectuer une recherche de plus longue sous-séquence commune dérivée de l'état-de-l'art. Cette étape coûteuse n'est effectuée que pour un petit sous-ensemble des séquences de la base, grâce au filtrage permis par la première étape. La figure 3.1 présente le principe général de notre proposition en le mettant en regard avec la méthode de référence de l'état-de-l'art.

Une partie de ces travaux a été publiée dans [Jégou 11b].

3.1 Recherche approximative de k-plus proches voisins locaux

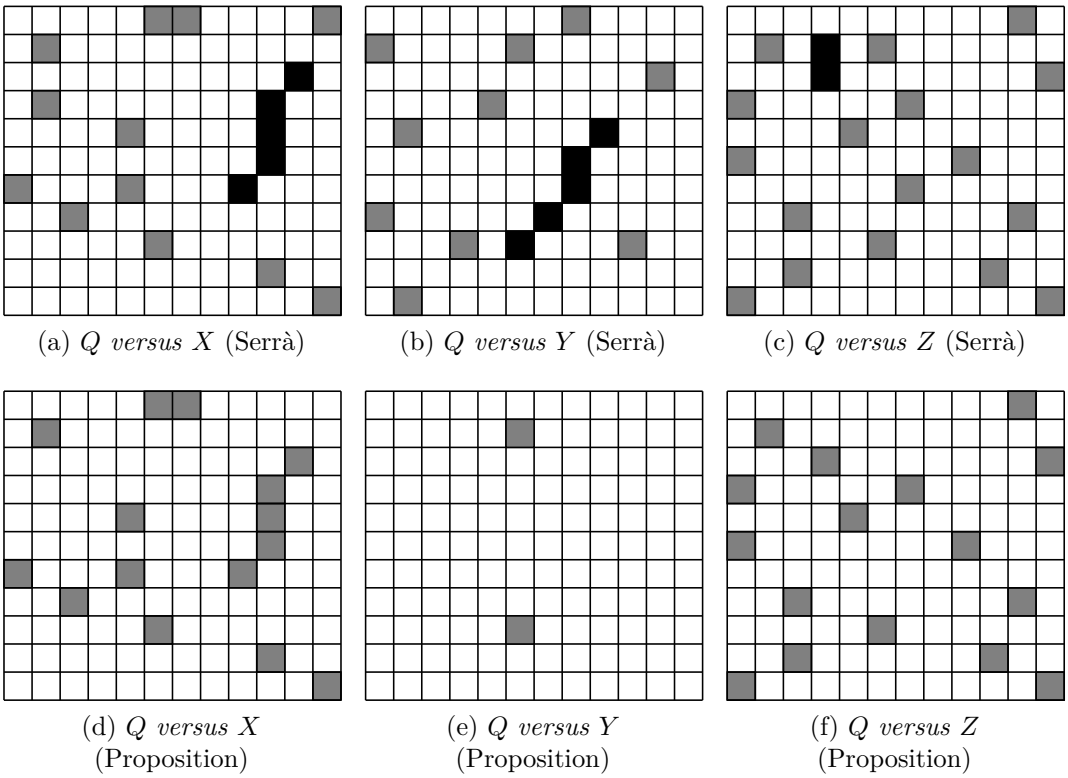
Nous commençons donc par proposer une méthode de recherche efficace de k-plus proches voisins locaux. Afin de capturer une information temporelle locale dans cette étape, nous concaténons, comme le font Serrà, Serra et Andrejzak [Serrà 09], les vecteurs de chromas par groupes de 5, de telle sorte que les vecteurs manipulés dans la suite seront d'une dimension 5 fois égale à la dimension de l'espace d'origine, soit une dimension résultante $\dim = 60$.

La quantité de vecteurs dans la base est grande, puisque les fréquences d'échantillonnage utilisées mènent à des séquences de l'ordre du millier de vecteurs par chanson. Ainsi, pour des bases de données de plusieurs milliers, voire plus, de chansons, les volumes de données considérés sont important et il convient alors de décrire ces vecteurs en utilisant une quantité de mémoire réduite.

Dans ce but, nous proposons ici une méthode inspirée du domaine du codage de source permettant d'approximer les distances entre les descripteurs de la requête et ceux de la base. Nous utilisons ensuite ces distances pour proposer un ensemble de k-plus proches voisins candidats.

3.1.1 Codage de source et indexation

Dans cette section, nous présentons la méthode d'indexation proposée par Jégou, Douze et Schmidt [Jégou 11a], qui utilise une approche de codage de source pour déterminer un ensemble de k-plus proches voisins approximatifs étant donnée une requête. Pour des raisons de clarté de la présentation, nous nous focaliserons ici sur la description du calcul de distance asymmétrique (*Asymmetric Distance Computation*,



(a) Q *versus* X (Serrà) (b) Q *versus* Y (Serrà) (c) Q *versus* Z (Serrà)

(d) Q *versus* X (Proposition) (e) Q *versus* Y (Proposition) (f) Q *versus* Z (Proposition)

FIGURE 3.1 – *Comparaison avec l'algorithme de [Serrà 09].* Ici, la séquence Q, représantant la chanson requête, est comparée à trois séquences X, Y et Z, dont seule la première représente une reprise de la chanson requête, les deux autres correspondant à des chansons sans lien avec la requête. On remarque que la méthode de Serrà a tendance à trouver un nombre important de correspondances (cases grisées), même pour des séquences sans lien, ce qui peut résulter, comme pour la chanson Y, à la découverte de chemins optimaux (cases noires) de taille conséquente. Au contraire, notre approche consiste à calculer des plus proches voisins dans le contexte de l'ensemble de la base des chansons. Ainsi, la chanson Y se retrouve avec un nombre de correspondances fortement diminué. Comme nous utilisons une robustification temporelle similaire à celle de Serrà, nous pouvons également écarter la séquence Z qui a beaucoup de correspondances non cohérentes temporellement.

en anglais, ou ADC) introduit dans cet article. Ce calcul a pour but, étant donnée une représentation compacte en mémoire des vecteurs de la base, d'estimer leur distance à un vecteur requête. Pour les mêmes raisons de fluidité de l'exposé, nous supposons que nous cherchons à retrouver le plus proche voisin pour chaque requête, même si la recherche de k-plus proches voisins avec $k > 1$ est tout à fait possible avec cette méthode.

Soient $\mathbf{x} \in \mathbb{R}^{\dim}$ un vecteur requête et $\mathcal{Y} = \{\mathbf{y_1}, \ldots, \mathbf{y_n}\}$ un ensemble de vecteurs parmi lesquels on souhaite trouver le plus proche voisin $\mathrm{NN}(\mathbf{x})$ de $\mathbf{x}$. L'approche ADC consiste à encoder chaque vecteur $\mathbf{y_i}$ par sa version quantifiée $\mathrm{q_c}(\mathbf{y_i}) \in \mathbb{R}^{\dim}$. Étant donné un quantifieur $\mathrm{q_c}(.)$ utilisant K centroïdes (typiquement, un K-means), chaque vecteur est encodé en utilisant $b_\mathrm{c} = \log_2(K)$ bits, en supposant que K est une puissance de 2.

Cette méthode de calcul de plus proche voisin approximatif considère implicitement l'indexation multidimensionnelle comme un problème d'approximation de vecteurs : un vecteur de la base $\mathbf{y}$ peut être décomposé en :

$$\mathbf{y} = \mathrm{q_c}(\mathbf{y}) + \mathrm{r}(\mathbf{y}), \tag{3.1}$$

où $\mathrm{q_c}(\mathbf{y})$ est le centroïde associé à $\mathbf{y}$ et où $\mathrm{r}(\mathbf{y})$ est le vecteur d'erreur résultant de la quantification, également appelé *vecteur résiduel*.

On peut calculer une distance approximée entre un vecteur requête $\mathbf{x}$ et un vecteur de la base $\mathbf{y_i}$ comme :

$$d_\mathrm{c}(\mathbf{x}, \mathbf{y_i}) = \sqrt{\|\mathbf{x} - \mathrm{q_c}(\mathbf{y_i})\|^2}. \tag{3.2}$$

On obtient alors le plus proche voisin approximatif $\mathrm{NN}_a(\mathbf{x})$ de $\mathbf{x}$ en minimisant l'estimateur de distance :

$$\begin{aligned} \mathrm{NN}_a(\mathbf{x}) &= \arg\min_i d_\mathrm{c}(\mathbf{x}, \mathbf{y_i}) && (3.3)\\ &= \arg\min_i d_\mathrm{c}(\mathbf{x}, \mathbf{y_i})^2 && (3.4)\\ &= \arg\min_i \|\mathbf{x} - \mathrm{q_c}(\mathbf{y_i})\|^2, && (3.5) \end{aligned}$$

qui est une approximation du calcul de distance exact :

$$\mathrm{NN}(\mathbf{x}) = \arg\min_i \|\mathbf{x} - \mathbf{y_i}\|^2. \tag{3.6}$$

Il est important de remarquer que, contrairement à la méthode de Weiss, Torralba et Fergus [Weiss 08], la requête $\mathbf{x}$ n'est pas convertie en un code, ce qui permet d'éviter toute erreur de quantification relative au vecteur requête.

Pour obtenir une bonne représentation des vecteurs de l'espace, il serait toutefois nécessaire d'utiliser des valeurs de K grandes ($K = 2^{64} \simeq 10^{19}$ pour un code de 64 bits). Le problème est alors double. D'une part, il est nécessaire d'apprendre un dictionnaire de taille K, ce qui nécessite une quantité de données d'apprentissage et un temps de calcul prohibitifs. D'autre part, le calcul de la distance entre la requête et chacun des centroïdes, au moment de la requête aura un coût trop élevé pour que cette solution ne soit envisageable.

Pour pallier ce problème, les auteurs proposent d'utiliser un *quantificateur produit*, pour lequel il n'est pas utile d'énumérer l'ensemble des centroïdes explicitement.

Chaque vecteur $\mathbf{y} \in \mathbb{R}^{\dim}$ est tout d'abord décomposé en m sous-vecteurs $\mathbf{y^1}, ..., \mathbf{y^m} \in \mathbb{R}^{\dim/m}$. Un quantificateur produit est alors défini comme une fonction

$$\mathrm{q_c}(\mathbf{y}) = \Big(\mathrm{q}^1(\mathbf{y^1}), ..., \mathrm{q}^m(\mathbf{y^m})\Big), \tag{3.7}$$

qui transforme un vecteur $\mathbf{y}$ en un ensemble d'indices en quantifiant séparément chacun des sous-vecteurs de $\mathbf{y}$. Chaque quantificateur individuel $\mathrm{q}^j(.)$ peut prendre K_s valeurs, apprises par un K_s-means.

Pour limiter la complexité de l'assignement qui est en $O(m \times K_\mathrm{s})$, il est préférable de fixer K_s à une petite valeur (par exemple K_s=256). Toutefois, l'ensemble des K centroïdes induit par le quantificateur produit $\mathrm{q_c}(.)$ demeure grand car on a :

$$K = (K_\mathrm{s})^m. \tag{3.8}$$

Les distances quadratiques dans l'équation 3.5 sont alors calculées en utilisant la décomposition ci-dessus présentée :

$$\begin{aligned} d_\mathrm{c}(\mathbf{x}, \mathbf{y})^2 &= \|\mathbf{x} - \mathrm{q_c}(\mathbf{y})\|^2 && (3.9)\\ &= \sum_{j=1}^{m} \|\mathbf{x^j} - \mathrm{q}^j(\mathbf{y^j})\|^2, && (3.10) \end{aligned}$$

où $\mathbf{y^j}$ est le $j^{\text{ième}}$ sous-vecteur de $\mathbf{y}$.

Les distances quadratiques utilisées dans la somme sont lues dans des tables de correspondance. Ces tables sont construites à la volée pour chaque requête, avant de débuter la recherche dans l'ensemble des codes de quantification. Pour cela, on stocke pour chaque sous-vecteur $\mathbf{x^j}$ de $\mathbf{x}$ et pour chaque indice de quantification i du quantificateur q^j correspondant, la distance entre $\mathbf{x^j}$ et le $i^{\text{ième}}$ centroïde issu de q^j. La complexité de la génération de cette table est en $O(d \times K_\mathrm{s})$. Si l'on a $K_\mathrm{s} \ll n$, cette complexité est négligeable devant la complexité en $O(d \times n)$ du calcul de la somme dans l'équation 3.5.

Enfin, les auteurs montrent que l'erreur quadratique entre la distance et son estimation est bornée en moyenne. Cela garantit que, asymptotiquement, des résultats parfaits seront obtenus en allouant un nombre de bits infini aux quantificateurs utilisés.

Les seuls paramètres de la méthode ADC sont le nombre de sous-vecteurs m et le nombre de bits alloués par quantificateur b_c. Dans la suite, nous fixons $b_\mathrm{c} = 8$ (c'est-à-dire, $K_\mathrm{s} = 256$), comme suggéré dans [Jégou 11a], ce qui implique que chaque vecteur est représenté par exactement m octets.

3.1.2 Codage de source et raffinement de l'estimation des distances

Une telle approximation des distances introduisant une erreur due au bruit de quantification, il est usuellement proposé une étape de post-vérification des distances qui, étant donnée la liste des k' plus proches voisins approximatifs, calcule les distances exactes entre chacun des vecteurs de cette liste et la requête pour fournir une liste finale de k-plus proches voisins [Datar 04, Muja 09]. Dans le cas de grandes bases de

vecteurs où l'ensemble des descripteurs ne peut pas être conservé en mémoire, cela implique des accès disque coûteux.

L'objectif de la méthode que nous proposons ici est d'éviter cette étape de post-vérification. L'idée est de tirer profit de l'information que l'on a sur le point de la base grâce à l'indexation. Cela est possible lorsque l'on utilise la méthode ADC puisque l'on a alors une reconstruction explicite approximative $\mathrm{q_c}(\mathbf{y_i})$ du vecteur de la base $\mathbf{y_i}$.

Nous supposons pour commencer que la première étape de la recherche retourne une liste de k' hypothèses. Ces vecteurs sont ceux pour lesquels une post-vérification est nécessaire.

Pour chaque vecteur $\mathbf{y_i}$ de cet ensemble d'hypothèses, le vecteur résiduel est égal à :

$$\mathrm{r}(\mathbf{y_i}) = \mathbf{y_i} - \mathrm{q_c}(\mathbf{y_i}). \tag{3.11}$$

Nous proposons de réduire l'énergie de ce vecteur résiduel afin de limiter l'influence de l'erreur de quantification sur les distances estimées. Pour ce faire, on encode le vecteur résiduel $\mathrm{r}(\mathbf{y_i})$ à l'aide d'un autre quantificateur produit $\mathrm{q_r}$ défini par ses valeurs de reproduction $\mathcal{C}_\mathrm{r}$:

$$\mathrm{q_r}(\mathrm{r}(\mathbf{y_i})) = \arg\min_{c \in \mathcal{C}_\mathrm{r}} \|\mathrm{r}(\mathbf{y_i}) - c\|^2, \tag{3.12}$$

où le quantificateur produit $\mathrm{q_r}(.)$ est appris sur un ensemble indépendant de vecteurs résiduels. Comme pour $\mathrm{q_c}(.)$, l'ensemble des valeurs de reproduction $\mathcal{C}_\mathrm{r}$ n'est jamais listé exhaustivement grâce à l'utilisation d'un quantificateur produit.

Le vecteur résiduel encodé peut être vu comme un ensemble de "bits de poids faibles" de la représentation du vecteur de la base, à ceci près que le terme "bits" fait généralement référence à la quantification scalaire. Le vecteur $\mathbf{y_i}$ peut alors être estimé par la somme $\mathbf{\hat{y}_i}$ de sa version approximée et de son vecteur résiduel décodé :

$$\mathbf{\hat{y}_i} = \mathrm{q_c}(\mathbf{y_i}) + \mathrm{q_r}(\mathrm{r}(\mathbf{y_i})). \tag{3.13}$$

Comme montré en figure 3.2, cet estimateur est utilisé au moment de la recherche pour mettre à jour les estimations de distances entre la requête $\mathbf{x}$ et les vecteurs de la base $\mathbf{y}$ sélectionnés comme de potentiels voisins :

$$d(\mathbf{x}, \mathbf{y})^2 \approx d_\mathrm{r}(\mathbf{x}, \mathbf{y})^2 = \|\mathrm{q_c}(\mathbf{y}) + \mathrm{q_r}(\mathrm{r}(\mathbf{y})) - \mathbf{x}\|^2. \tag{3.14}$$

Le quantificateur produit de raffinement $\mathrm{q_r}$ est paramétré par son nombre de sous-quantificateurs et le nombre de bits alloués à chaque sous-quantificateur. Comme pour $\mathrm{q_c}$, nous utilisons 1 octet par quantificateur et le seul paramètre à fixer est donc le nombre m' d'octets par vecteur de la base.

Chaque vecteur indexé occuppera finalement $m + m'$ octets en mémoire.

Algorithme

Nous détaillons ici comment les codes de raffinement sont utilisés pour réordonner les hypothèses fournies par l'ADC. Dans la suite, nous noterons l'approche résultante ADC+R. Comme pour la plupart des algorithmes d'indexation, nous distinguons la phase hors-ligne et la phase de requête.

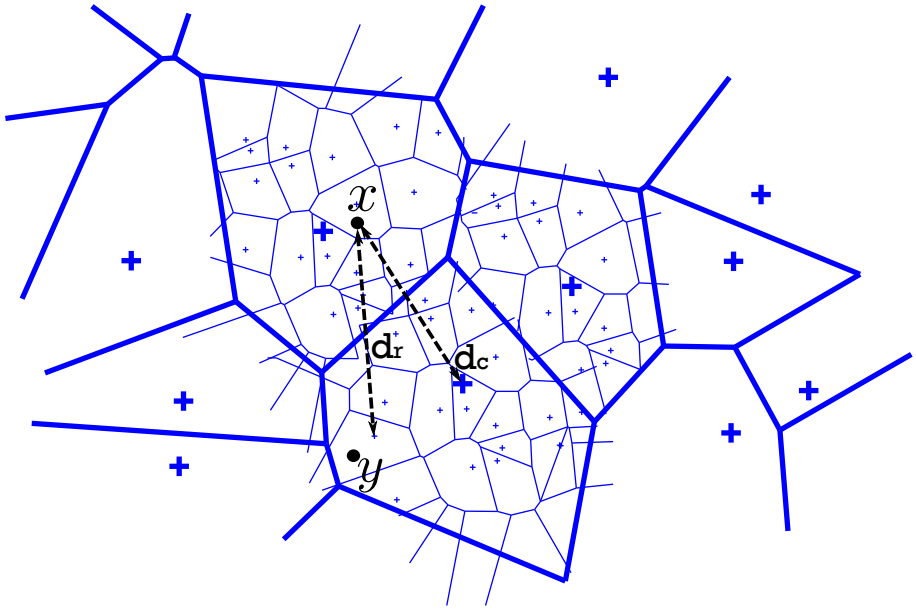


FIGURE 3.2 – *Illustration du processus de raffinement proposé.* Pour chaque vecteur de la base $\mathbf{y}$, la distance $d_c(\mathbf{x}, \mathbf{y}) = d(\mathbf{x}, q_c(\mathbf{y}))$ est calculée pour construire la liste des plus-proches voisins potentiels. Pour les vecteurs $\mathbf{y}$ sélectionnés, la distance est raffinée pour valoir $d_r(\mathbf{x}, \mathbf{y})$, qui est obtenue en calculant la distance entre $\mathbf{x}$ et l'approximation plus fine de $\mathbf{y}$.

La première consiste à apprendre les paramètres d'indexation et à construire la structure associée étant donné un jeu de vecteurs. Elle consiste en les 3 étapes suivantes :

1. les quantificateurs $q_c(.)$ et $q_r(.)$ sont appris sur un jeu d'apprentissage ;
2. les vecteurs de la base $\mathcal{Y} = \{\mathbf{y_1}, \ldots, \mathbf{y_n}\}$ à indexer sont encodés en utilisant q_c, ce qui produit les codes $q_c(\mathbf{y_i})$ pour $i \in [\![1; n]\!]$;
3. les vecteur résiduels sont encodés, produisant les codes $q_r(\mathbf{y_i} - q_c(\mathbf{y_i}))$ associés aux vecteurs indexés.

Ensuite, la recherche d'un vecteur requête $\mathbf{x}$ dans une base ainsi indexée se fait comme suit :

1. la méthode ADC est utilisée pour estimer les distances entre la requête et les vecteurs de la base ;
2. une liste $\mathcal{L}$ de k' hypothèses est calculée à partir de ces distances estimées, les vecteurs sélectionnés étant ceux qui minimisent l'estimateur de l'équation 3.10, lequel est calculé directement dans le domaine compressé ;
3. pour chaque vecteur $\mathbf{y_i} \in \mathcal{L}$, le vecteur approximé $\hat{\mathbf{y}}_\mathbf{i}$ est explicitement reconstruit comme la somme de la première approximation $q_c(\mathbf{y_i})$ et du vecteur résiduel encodé $q_r(\mathbf{y_i})$, afin de calculer la distance estimée $d(\mathbf{x}, \hat{\mathbf{y}}_\mathbf{i})^2$;
4. les vecteurs de $\mathcal{L}$ associés aux k plus petites distances raffinées sont retournés.

Cette dernière étape peut être effectuée de deux manières. Si k' et k sont du même ordre de grandeur, alors, il suffit de trier les distances. Par contre, si l'on a $k' \gg k$, il est plus opportun de maintenir une structure de tas binaire de capacité fixe k.

Dans tous les cas, on obtient en sortie une liste réordonnée de k-plus proches voisins approximatifs. Le choix du nombre k' de vecteurs à conserver à l'issue de la première étape dépend des paramètres m, m' et k et de la distribution des vecteurs. Pour assurer que la complexité de l'étape de post-vérification soit négligeable devant celle de la première étape, nous utilisons typiquement $k' = 2k$.

Variante non exhaustive

Jusqu'à maintenant, nous nous sommes limités à la méthode ADC issue de [Jégou 11a], qui nécessite un parcours de l'ensemble des codes de la base. Il est toutefois important de noter que la méthode de post-vérification proposée ici peut être appliquée à n'importe quelle méthode pour laquelle une reconstruction approximative des vecteurs indexés est possible. En particulier, il est possible de considérer la variante appelée IVFADC dans [Jégou 11a], qui évite le sus-cité parcours de l'ensemble des codes de la base en utilisant une structure de fichier inversé. Cette méthode implique l'utilisation un quantificateur additionnel, grossier, qui permettra de filtrer les vecteurs de la base les moins susceptibles d'être parmi les k-plus proches voisins de la requête.

L'adaptation de notre méthode de post-vérification à IVFADC est directe puisqu'IVFADC fournit une reconstruction explicite des k' vecteurs de la base sélectionnés. L'utilisation d'IVFADC introduit deux paramètres supplémentaires qui sont le nombre c d'indices de quantification du quantificateur grossier d'une part et le nombre v de listes inversées qui sont visitées pour chaque requête. Le principal avatange de l'utilisation de cette variante est de limiter les calculs de distances approchées à une fraction (typiquement de l'ordre de v/c) de la base, au risque de manquer certains plus proches voisins si v/c n'est pas assez grand. Enfin, on peut noter que l'on devra, pour chaque élément de la base, stocker son indice de quantification relatif au quantificateur grossier, ce qui augmentera de $\log_2(c)$ bits (typiquement 4 octets) l'espace mémoire utilisé pour représenter un vecteur de la base. Dans la suite, la méthode utilisant conjointement IVFADC et notre méthode de post-vérification sera notée IVFADC+R.

3.2 Recherche de correspondances étendues

Nous proposons ici d'utiliser les plus proches voisins sélectionnés par la méthode IVFADC+R pour effectuer une robustification temporelle inspirée de la proposition de [Serrà 09].

Soit une séquence requête $\mathbf{x} = \mathbf{x_1} \cdots \mathbf{x_l}$ constituée de l vecteurs de dimension dim. On commence par générer 12 versions de cette séquence correspondant à sa transposition dans chacune des tonalités issues des 12 demi-tons de la gamme. Pour transposer la séquence d'un demi-ton, il suffit d'appliquer la permutation circulaire suivante à chacun des vecteurs $\mathbf{x_i}$ de la requête :

$$\mathbf{x_i} = (x_i^1, \cdots, x_i^{\dim-1}, x_i^{\dim}) \mapsto \mathbf{x_{i,t_1}} = (x_i^2, \cdots, x_i^{\dim}, x_i^1). \tag{3.15}$$

Ensuite, pour chacune de ces 12 versions et pour chaque vecteur de la requête, on interroge la base de données à l'aide de la méthode IVFADC+R. On obtient alors,

pour chaque vecteur $\mathbf{x_{i,t_j}}$, un ensemble de k-plus proches voisins.

3.2.1 Estimation de la similarité par vote

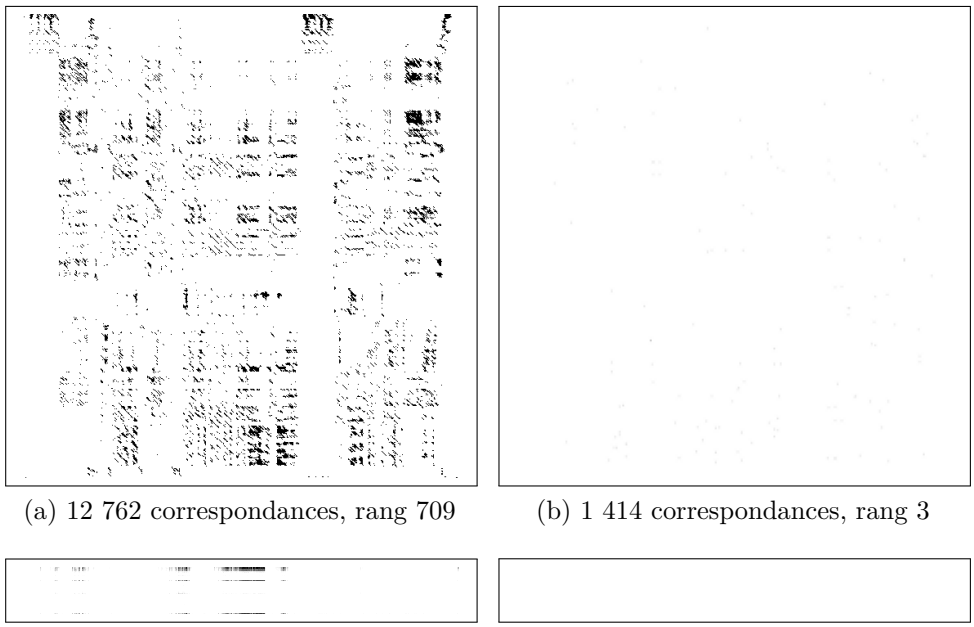
Une fois ces k-plus proches voisins calculés, nous proposons de calculer un score de similarité global entre séquences qui soit égal à la somme, pour tous les vecteurs $\mathbf{x_{i,t_j}}$ de la séquence requête, du nombre d'éléments de la séquence de la base qui sont classés parmis les k-plus proches voisins de $\mathbf{x_{i,t_j}}$. Les séquences de la base sont alors classées, dans une liste $\mathcal{L}_{\text{vote}}(t_j)$, par ordre décroissant de ce score. On obtient une liste ordonnée par indice de transposition. On fusionne ces listes en retenant, pour chaque paire de séquence, l'indice de transposition correspondant au score maximal et le score correspondant. La liste résultante est notée $\mathcal{L}_{\text{vote}}$

3.2.2 Raffinement par programmation dynamique

Nous proposons ici de raffiner cette liste à l'aide des principes énoncés dans [Serrà 09]. Tout d'abord, nous commençons par limiter la liste de séquences candidates en ne conservant que les k_{vote} premières séquences de $\mathcal{L}_{\text{vote}}$. En utilisant cette liste, on obtient également l'indice de transposition optimal pour chaque paire de chansons. Ensuite, pour chacune de ces séquences candidates, on obtient la matrice binaire utilisée par Serrà, Serra et Andrejzak [Serrà 09] à partir des résultats fournis par IVFADC+R dans la transposition considérée. Il est à noter qu'ici, nous n'utilisons pas la recherche de plus proches voisins réciproques, comme c'est le cas dans la méthode présentée dans [Serrà 09]. Ce choix se justifie, d'une part, par la quantité considérable de calculs que nous économisons et, d'autre part, par le fait que les plus proches voisins que nous avons calculés sont contextuels, c'est-à-dire qu'ils sont répartis sur l'ensemble des séquences de la base, ce qui n'est pas le cas dans l'algorithme de [Serrà 09], qui n'effectue que des comparaisons par paires de séquences. Ainsi, pour [Serrà 09], si la séquence requête est très différente de la séquence candidate de la base, le fait d'extraire un nombre fixe de k-plus proches voisins parmis les vecteurs de la séquence de la base peut mener à trouver des correspondances entre un vecteur de la requête, qui se trouverait dans une région peu dense de l'espace, et un vecteur de la séquence de la base qui lui est peu similaire. C'est cet artefact qui motive l'utilisation de la recherche de plus proches voisins réciproques.

L'influence de la qualité des plus proches voisins retournés est illustrée par la figure 3.3, qui présente un comparatif de plus proches voisins retournés par l'algorithme de [Serrà 09] et notre proposition. On remarque que le nombre de correspondances extrait par notre méthode est bien plus faible et que l'on évite notamment de trouver de nombreuses correspondances entre chansons qui ne sont pas des reprises, comme cela peut être le cas avec l'algorithme de [Serrà 09] (voir figure 3.3c).

Le reste du calcul de plus longue sous-séquence commune pondérée suit les principes introduits dans [Serrà 09] et détaillés en chapitre 1 de ce document.



(a) 12 762 correspondances, rang 709

(b) 1 414 correspondances, rang 3

(c) 32 241 correspondances, rang 1

(d) Aucune correspondance, non classée

FIGURE 3.3 – *Plus proches voisins locaux utilisés pour la recherche de reprises musicales.* La colonne de gauche correspond à des matrices de similarité obtenues par [Serrà 09] alors que la colonne de droite correspond aux matrices de similarité obtenues avec la méthode présentée dans ce chapitre. La première ligne correspond effectivement à la mise en correspondance de deux versions d'une même chanson, au contraire de la deuxième ligne.

3.3 Validation expérimentale

Pour évaluer notre proposition, nous utilisons un jeu de données constitué de 2018 chansons décrites à l'aide de vecteurs de chroma calculés toutes les 100 millisecondes sur une fenêtre de 200 millisecondes. Le jeu de requête est constitué de 82 chansons. Ce jeu de données a été introduit dans [Casey 06b] puis utilisé dans [Casey 07, Casey 08]. Comme suggéré dans [Serrà 09], nous concaténons les vecteurs de chroma fournis par groupes de 5, pour obtenir des vecteurs de dimension 60 contenant une information temporelle locale.

3.3.1 Nombre de votants

Nous commençons par estimer le nombre k de plus proches voisins locaux à retenir pour chaque vote. Pour que cette estimation ne dépende pas des autres paramètres utilisés, nous proposons d'utiliser les plus proches voisins exacts plutôt que les plus proches voisins approximés retournés par IVFADC+R.

La figure 3.4 montre les performances obtenues en termes de rappel en ne considérant que la liste $\mathcal{L}_{\text{vote}}$ (l'étape de raffinement est ici ignorée). On remarque que des valeurs faibles pour k donnent les meilleurs résultats. Nous conservons donc, dans la

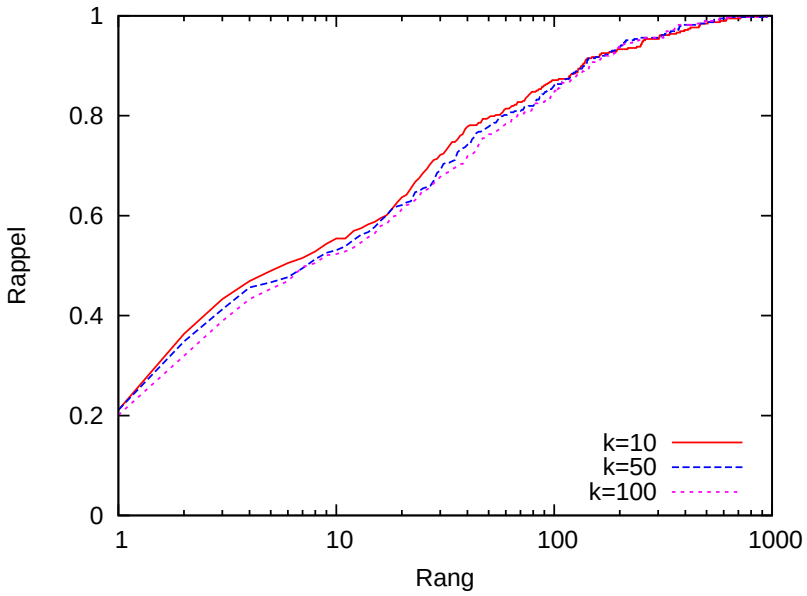


FIGURE 3.4 – *Performances d'un système utilisant la recherche de plus proches voisins exacts.* Le paramètre qui varie entre les différentes courbes est le nombre de plus proches voisins utilisés pour le vote.

suite la valeur $k = 10$.

3.3.2 Quantité de mémoire utilisée

Nous avons vu, plus haut, que les seuls paramètres à fixer pour la méthode IVFADC+R sont m et m' les nombres d'octets utilisés par vecteur de la base correspondant respectivement aux deux étapes d'approximation, ainsi que le rapport v/c correspondant au quantificateur grossier utilisé. Pour cette série d'expériences, nous fixons $c = 1024$ et $v = 4$, ce qui mène à estimer les distances, pour chaque requête, à environ $1/250^{\text{ième}}$ des vecteurs de la base. Dans la suite, nous fixerons $m = m'$ et étudions l'impact de la quantité de mémoire utilisée sur la qualité de la liste ordonnée retournée $\mathcal{L}_{\text{vote}}$.

La figure 3.5 montre que la différence entre un système utilisant la recherche de plus proches voisins exacts et un système utilisant peu de mémoire comme dans le cas d'IVFADC+R avec les paramètres $m = m' = 4$ est très faible. Il existe même des valeurs de rang pour lesquelles le rappel est meilleur avec la version approchée. Cela nous incite à utiliser, dans la suite de nos expériences les paramètres $m = m' = 4$ pour obtenir un système performant à la fois en termes de rappel, de temps de calcul et de mémoire utilisée.

3.3.3 Évaluation du système complet

Dans la suite, on fixe le paramètre k_{vote} à 100. Cela signifie que les listes ordonnées de résultats ne seront pas modifiées au-delà du centième rang et justifie que nous limitions les figures de rappel à ces 100 premiers rangs. Le seul paramètre qui nous

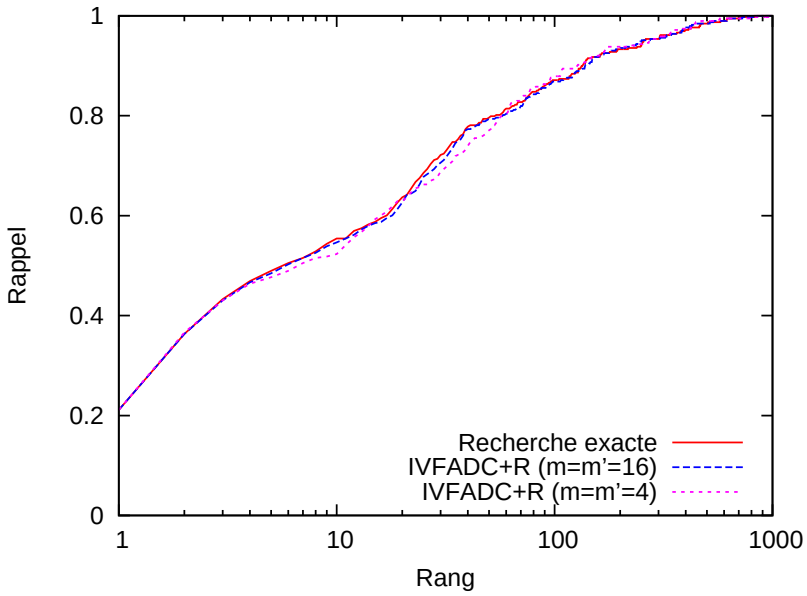


FIGURE 3.5 – *Performances d'un système utilisant la recherche de plus proches voisins approchés IVFADC+R.* Le paramètre qui varie entre les différentes courbes est la quantité de mémoire utilisée ($m + m'$) et la méthode de référence est la recherche de plus proches voisins exacts.

reste alors à fixer est le nombre de plus proches voisins locaux considérés pour peupler la matrice binaire de plus proches voisins.

La figure 3.6 montre que le choix de ce paramètre a finalement une influence assez faible sur la qualité des résultats retournés. De plus, cette figure montre que, non seulement notre méthode est plus rapide que la méthode proposée par [Serrà 09] [1], mais l'utilisation de plus proches voisins contextuels permet d'améliorer les performances en termes de rappel.

On note également ici que l'utilisation de l'alignement dynamique comme étape de robustification temporelle apporte un gain significatif.

3.4 Bilan

Nous avons présenté dans ce chapitre une méthode de recherche de k-plus proches voisins approximatifs pour des vecteurs isolés. Cette méthode permet de représenter les vecteurs d'une base de données par une quantité de mémoire réduite, qui permet de considérer des traitements en mémoire pour des volumes de données jusque-là inaccessibles. Cette méthode a en effet été évaluée sur un jeu de données contenant 1 milliard de vecteurs SIFT [Jégou 11b] qui est le plus grand jeu de données non synthétique proposé jusqu'à maintenant à notre connaissance.

1. Pour notre méthode, le temps de calcul est d'environ 2 minutes par requête contre 4 minutes par requête pour une implémentation non naïve de l'algorithme concurrent. Il est important de noter, sur ce point, que si ces temps de calcul sont importants, notre méthode peut être ajustée par le biais des paramètres $m + m'$, v/c et k_{vote} pour atteindre des points de fonctionnement moins coûteux et ainsi permettre la recherche dans des bases plus grandes.

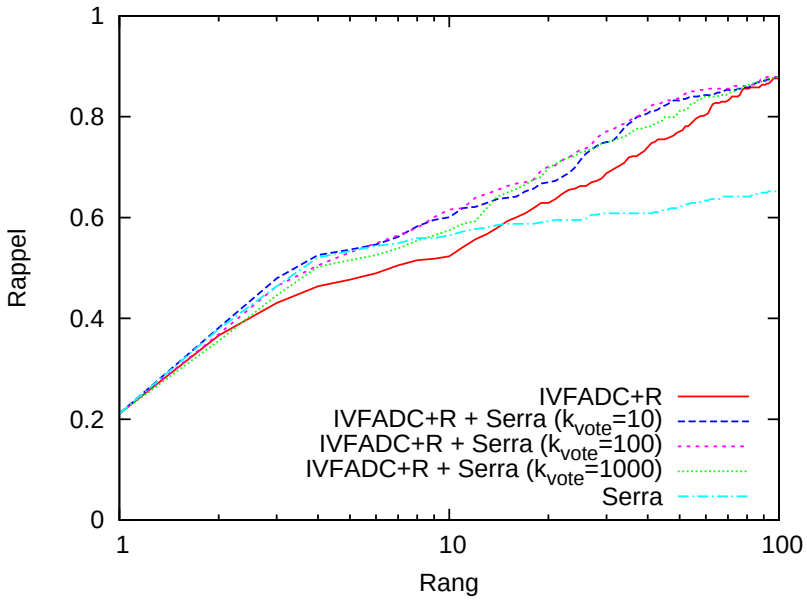


FIGURE 3.6 – *Performances d'un système utilisant la recherche de plus proches voisins approchés IVFADC+R et un raffinement par alignement dynamique.* Le paramètre qui varie entre les différentes courbes est le nombre de plus proches voisins considérés pour l'étape d'alignement dynamique et les deux méthodes étalon sont, d'une part, celle proposée dans [Serrà 09] et notée "Serra" et, d'autre part, notre méthode n'utilisant que le vote sans raffinement par alignement dynamique.

Nous avons ensuite utilisé cette méthode pour accélérer et améliorer les performances de l'algorithme de référence de l'état-de-l'art dans le domaine de la recherche de reprises musicales. Les résultats montrent qu'avec une utilisation de la mémoire très restreinte, il est possible d'obtenir une représentation des vecteurs de la base suffisante pour ce type d'applications.

Ces travaux montrent qu'une étape de pré-traitement n'utilisant que peu d'information temporelle permet d'accélérer les calculs tout en améliorant les résultats grâce à l'utilisation du contexte lors de la comparaison des séquences deux à deux. Toutefois, étant donnée la taille des séquences considérées, le nombre de requêtes de vecteurs locaux à effectuer pour rechercher une séquence dans une base reste un frein à une accélération plus grande des calculs. Il pourrait alors être judicieux d'étudier des moyens d'extraire les régions temporelles saillantes des séquences afin de concentrer les requêtes sur ces zones pertinentes.

Bibliographie

[Casey 06] M. Casey & M. Slaney. *Song intersection by approximate nearest neighbor search.* In Proceedings of ISMIR, pages 144–149, 2006.

[Casey 07] M. Casey & M. Slaney. *Fast recognition of remixed music audio.* In Proceedings of the IEEE International Conference on Acoustics, Speech

and Signal Processing, pages 1425–1428, 2007.

[Casey 08] M. Casey, C. Rhodes & M. Slaney. *Analysis of minimum distances in high-dimensional musical spaces*. IEEE Transactions on Audio, Speech, and Language Processing, vol. 16, no. 5, pages 1015–1028, 2008.

[Datar 04] M. Datar, N. Immorlica, P. Indyk & V.S. Mirrokni. *Locality-sensitive hashing scheme based on p-stable distributions*. In Proceedings of the Symposium on Computational Geometry, pages 253–262, 2004.

[Jégou 11a] H. Jégou, M. Douze & C. Schmid. *Product quantization for nearest neighbor search*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 1, pages 117–128, 2011.

[Jégou 11b] H. Jégou, R. Tavenard, M. Douze & L. Amsaleg. *Searching in one billion vectors : re-rank with source coding*. In IEEE International Conference on Acoustics, Speech and Signal Processing, 2011.

[Muja 09] M. Muja & D. G. Lowe. *Fast approximate nearest neighbors with automatic algorithm configuration*. In Proceedings of the International Conference on Computer Vision Theory and Applications, 2009.

[Serrà 09] J. Serrà, X. Serra & R.G. Andrzejak. *Cross Recurrence Quantification for Cover Song Identification*. New Journal of Physics, vol. 11, 2009.

[Weiss 08] Y. Weiss, A. Torralba & R. Fergus. *Spectral Hashing*. In Proceedings of the Annual Conference on Neural Information Processing Systems, 2008.

Chapitre 4

De l'importance de la notion de séquence

Nous avons introduit, dans les deux chapitres précédents, des méthodes permettant d'utiliser l'alignement dynamique pour la comparaison de séquences dans de grandes bases de séquences multimédias. L'utilisation de telles techniques repose, implicitement, sur l'idée que la notion de séquence, c'est-à-dire l'ordre d'apparition des descripteurs, est une propriété forte de ce type de données, qui nécessite l'utilisation de stratégies de recherche adaptées.

Ce chapitre se propose de discuter l'importance de cette notion en se demandant s'il est toujours nécessaire de préserver l'ordre des descripteurs à chacune des étapes de la recherche ou si, dans certains cas, on peut s'en dispenser. Il existe en effet des domaines où, pour des raisons historiques, l'alignement dynamique est utilisé sans que n'ait été proposée d'évaluation fiable de ses apports.

On peut considérer, comme nous l'avons souligné dans le premier chapitre de ce document, trois options fondamentalement différentes de traitement de l'aspect séquentiel des données. Le but de ce chapitre est d'extraire les propriétés intrinsèques d'une application donnée qui puissent justifier de l'utilisation de l'une ou l'autre de ces options. Le premier choix possible est de délibérément ignorer l'aspect temporel pour considérer une séquence de descripteurs comme un ensemble d'observations non ordonnées. Une autre option est d'ignorer, dans un premier temps, ce caractère temporel pour effectuer une recherche peu coûteuse dont les résultats seront ensuite robustifiés par un post-traitement adapté à l'axe du temps. Enfin, la dernière option est d'intégrer la temporalité des données dès le début du processus, en construisant une méthode d'indexation basée sur une métrique telle que la DTW, comme nous l'avons fait au chapitre 2 de ce document.

Cette discussion autour de l'apport de la notion de séquence se justifie par différents éléments. D'une part, Casey et Slaney [Casey 06a] affirment que le problème de la recherche d'un court extrait musical (comme un refrain par exemple) au sein d'une base de données ne peut pas être résolu avec une représentation d'où

l'information temporelle est absente. Pour arriver à cette conclusion, les auteurs comparent la DTW avec une approche utilisant des sacs de mots en fixant un nombre de mots très faible (64 au plus). Or, si l'on s'intéresse aux travaux autour des sacs de mots issus de la communauté de la vision par ordinateur, l'ordre de grandeur du nombre de mots que l'on doit utiliser pour ce type d'approches est bien plus grand (citons par exemple *Video Google* pour lequel les auteurs utilisent de l'ordre de 10 000 mots visuels). En effet, si l'on utilise peu de mots pour décrire nos données, la phase de quantification des descripteurs en mots mène à une perte d'information trop importante et alors la représentation adoptée est trop peu informative. On peut également remarquer que, si l'on utilise au contraire un nombre de mots trop élevé, on perdra la capacité de généralisation qui est centrale dans l'utilisation de telles techniques. Il nous semble alors opportun d'évaluer, sur des jeux de données aux caractéristiques proches de ceux utilisés par Casey et Slaney, si un réglage plus raisonnable de ce paramètre permet d'obtenir des résultats compétitifs avec ceux obtenus par l'alignement dynamique ou si la faible prise en compte de l'axe temporel par les sacs de mots est un problème intrinsèque les rendant inutilisables pour ce type de recherche.

Tout comme dans le cas de Casey et Slaney, les travaux de Chantamunee et Gotoh [Chantamunee 08] présentés à TRECVID en 2008 pour une tâche de détection de copies vidéos, utilisent la DTW pour aligner des plans deux à deux. Il nous paraît ici important d'évaluer l'apport de techniques d'alignement dynamique pour des tâches pour lesquelles les éventuelles distorsions temporelles sont faibles et peu en rapport avec le modèle de déformations associé à ces techniques.

Dans d'autres cas, où les différentes occurences d'un même motif sont beaucoup plus bruitées, et notamment en ce qui concerne leur dynamique, il nous semble important de quantifier l'apport attendu de techniques d'alignement dynamique. Nous prenons ici pour exemple de cette catégorie d'applications la recherche de mots parlés, pour laquelle Muscariello, Gravier et Bimbot [Muscariello 09] proposent une méthode dérivée de la DTW. Notons également que, pour une application comparable, le choix de Fraihat et Glotin [Fraihat 10] se porte également sur l'utilisation de la DTW, avec un jeu de pondérations particulier, sur lequel nous reviendrons. Il s'agit de retrouver, dans un flux audio, différentes occurences d'un même mot prononcé par des locuteurs variés, chaque occurence ayant sa dynamique propre.

Cette discussion s'attache à tenter de dégager des éléments de compréhension permettant de déterminer les situations pour lesquelles une approche prenant en compte le temps de manière forte, comme la DTW, est nécessaire et quand une approche ayant une prise en compte plus faible de la notion de séquence suffit. Cette discussion a donc pour but de poser des jalons relatifs à la nature des données considérées et aux distorsions temporelles qu'elles sont susceptibles d'avoir subies. Notons que, selon les applications, ces distorsions peuvent avoir une origine artificielle, comme dans le cas de la réutilisation d'extraits vidéos édités et remontés, ou naturelle, lorsque la notion de similarité est plus floue comme par exemple pour la reconnaissance de mots parlés. Nous laissons volontairement de côté ici toute considération de coût de calcul ou de contrainte sur l'espace de stockage nécessaire ou la quantité de mémoire utilisée.

La suite de ce chapitre se structure ainsi : après avoir défini notre cadre et nos

hypothèses de travail, nous présentons les trois grands types de représentation des données que nous considérons et les méthodes de recherche associées. Nous exposons ensuite les performances de ces méthodes sur deux jeux de données représentatifs des problématiques exposées ci-dessus : l'un proposant une recherche de quasi-réplicats et l'autre consistant en une recherche de mots parlés pour lesquels les déformations temporelles sont importantes. Enfin, nous tirons de ces expériences une grille d'analyse de l'adéquation d'une méthode à une situation donnée.

4.1 Cadre et hypothèses

Nous nous intéressons dans la suite à des problématiques de recherche de séquences dans des bases de données multimédias. Les séquences manipulées consistent alors en des suites de vecteurs numériques multidimensionnels. Les bases considérées sont des ensembles de séquences et l'on cherche, pour une séquence requête, à retrouver les séquences de la base qui lui sont les plus similaires.

Dans le but d'illustrer la part d'information temporelle captée par chaque méthode, nous proposons d'utiliser des séquences symboliques jouets. La base illustrative sera composée des séquences {ABCDE, EDCBA, AABBCCDDEE, ABCXXXDF, BXAXCXXFXD} et la séquence requête considérée sera la séquence ABCDF. Le tableau 4.1 résume les rangs obtenus par les différentes séquences de cette base pour les approches détaillées dans la suite de ce chapitre.

4.2 Représentations des données et processus de recherche

Nous présentons ici trois approches possibles pour la comparaison de descripteurs, par ordre croissant de prise en compte de la notion de séquence. Pour chaque approche, nous détaillons la représentation utilisée pour les séquences et le processus de recherche associé.

4.2.1 Représentation sans notion de séquence : des vecteurs ordonnés aux sacs de mots

Le modèle introduit ici est issu de la recherche d'information textuelle. Il permet de représenter les séquences multimédias sous la forme de vecteurs de grande dimension dans lesquels l'ordre des descripteurs est ignoré.

Recherche d'information textuelle et modèle vectoriel

Le modèle vectoriel de Salton, Wong et Yang [Salton 75] est le modèle le plus utilisé en recherche d'information textuelle. Il représente chaque document par un vecteur de fréquences des termes d'indexation appelé *sac de mots*. Ces vecteurs sont de très grande dimension D (égale au nombre de termes du dictionnaire considéré) et sont très creux, puisque seuls quelques termes parmi tous ceux du dictionnaire sont utilisés dans chaque document. Cette éparsité permet de comparer deux documents

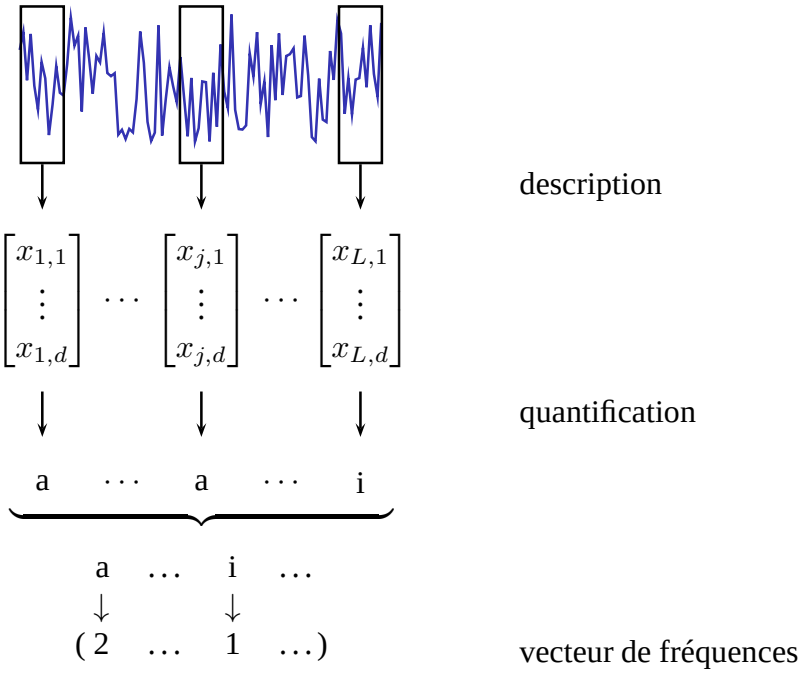


FIGURE 4.1 – *Description d'une séquence en sac de mots.*

avec une complexité restreinte. En effet, pour deux documents représentés par des vecteurs ayant respectivement d_1 et d_2 coordonnées non nulles, avec $d_1 \ll D$ et $d_2 \ll D$, la complexité du calcul de distance cosinus est en $O(d_1 + d_2)$ alors qu'elle serait en $O(D)$ si les vecteurs n'étaient pas creux.

Adaptation aux séquences multimédias

Sivic et Zisserman [Sivic 03] ont proposé d'utiliser ce modèle vectoriel pour représenter les images. Les mots utilisés sont alors des *mots visuels* qui correspondent à des versions quantifiées de descripteurs locaux extraits de l'image représentée.

Nous proposons ici d'utiliser le même type d'approche pour représenter des séquences de descripteurs. Ainsi, à partir de chaque image d'une vidéo (ou de chaque trame d'un enregistrement sonore), nous extrayons un descripteur représentatif de l'ensemble de son contenu. Nous quantifions ensuite tous ces descripteurs pour les transformer en *mots* et représentons une séquence de descripteurs par le vecteur de fréquences des mots qui la composent. La figure 4.1 synthétise les différentes étapes nécessaires à la représentation d'une séquence numérique (possiblement multidimensionnelle même si, sur la figure, pour des raisons de lisibilité, une séquence monodimensionnelle est utilisée) par un sac de mots.

Le seul paramètre introduit par cette quantification des données est le nombre k de mots à utiliser. Nous avons vu plus haut dans ce chapitre qu'il était important de n'utiliser ni une valeur trop basse pour k qui induirait une perte d'information irrémédiable, ni une valeur trop élevée qui ferait perdre aux mots extraits leur bonne propriété de généralisation.

	Sacs de mots	Sacs de mots n-grammes	DTW
ABCDE	1 (*ex aequo*)	1	2 (*ex aequo*)
EDCBA	1 (*ex aequo*)	4 (*ex aequo*)	4
AABBCCDDEE	1 (*ex aequo*)	2	2 (*ex aequo*)
ABCXXXDF	4	3	1
BXAXCXXFXD	5	4 (*ex aequo*)	5

TABLEAU 4.1 – *Récapitulatif des rangs obtenus pour les séquences illustratives avec chaque type de représentation..* La requête utilisée ici est la séquence ABCDF.

Métrique de similarité entre séquences

Pour comparer deux séquences décrites par un vecteur de fréquences, nous utilisons, comme il est de coutume, la distance cosinus entre ces vecteurs. Cette distance peut être vue comme un processus de vote pondéré où les pondérations utilisées correspondent au produit des normes euclidiennes des vecteurs de fréquences considérés. Ainsi, on perd totalement l'information temporelle de dépendance entre les descripteurs. Notamment, nous choisissons ici volontairement de ne pas avoir recours à une étape de robustification temporelle des résultats (comme celles présentées en section 1.4), dans le but d'évaluer les performances d'une approche ignorant totalement l'aspect temporel des données.

Séquences symboliques illustratives

En utilisant les sacs de mots pour effectuer une recherche dans la base de séquences symboliques introduite plus haut, les trois séquences ABCDE, EDCBA et AABBCCDDEE seront considérées comme également similaires à la requête et les séquences BXAXCXXFXD et ABCXXXDF, qui contiennent des symboles absents de la requête, seront légèrement pénalisées, même si leur score restera bon car elles contiennent les symboles A, B, C, D et F. On voit bien ici les problèmes soulevés par l'absence totale d'information temporelle qui implique de considérer comme extrêmement similaire une séquence (EDCBA) ayant une structure totalement incompatible avec celle de la requête.

4.2.2 Représentation avec faible notion de séquence : les modèles n-grammes

La représentation d'une séquence par un sac de mots efface complètement l'information temporelle contenue dans les descripteurs. Il a donc été proposé dans le cadre de la recherche textuelle de reconstituer une partie de cette information à l'aide de n-grammes, qui sont des ensembles de n symboles contigus.

Une première question qui se pose lorsque l'on extrait des n-grammes d'une séquence est de savoir quel recouvrement utiliser. Si l'on n'autorise aucun recouvrement, alors le vecteur de fréquences obtenu sera plus creux, ce qui présente l'avantage de diminuer la complexité de la comparaison de deux séquences entre elles. Par

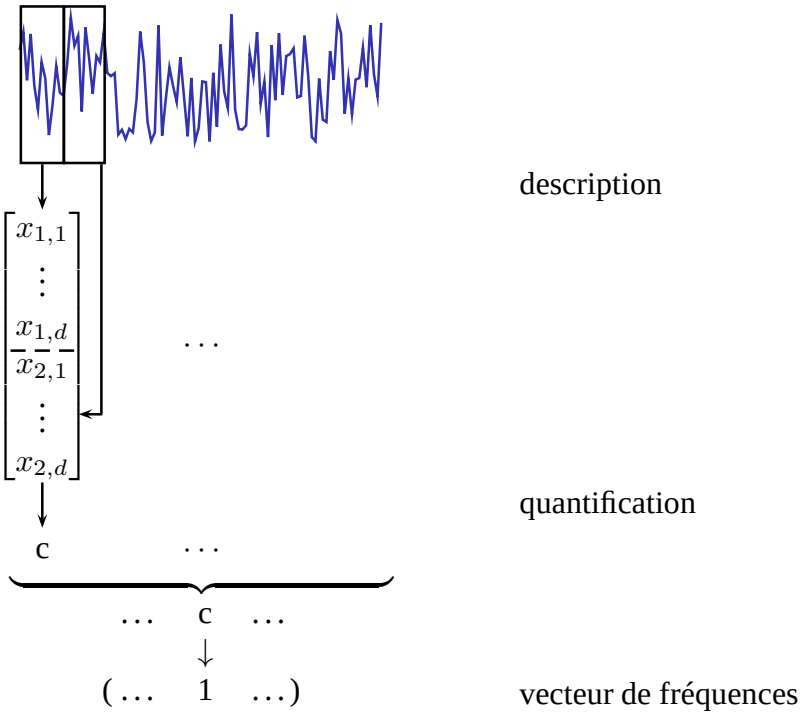


FIGURE 4.2 – *Description d'une séquence en sac de mots n-gramme.*

contre, il est possible de manquer des correspondances. Si l'on considère par exemple les séquences symboliques ABCDEF et BCDEFG que l'on décrit par des 3-grammes sans recouvrement, on n'aura aucune correspondance entre les 3-grammes ABC et DEF d'une part et les 3-grammes BCD et EFG d'autre part, alors que les séquences sont tout de même ressemblantes. Pour éviter cet écueil, il est usuellement admis qu'il est préférable d'extraire tous les n-grammes possibles de chaque séquence en utilisant un recouvrement de $n-1$ symboles, ce qui permet d'avoir une description plus fine de la séquence.

Ensuite, il existe deux façons de transformer des groupes de n descripteurs en mots. On peut soit concaténer les descripteurs avant de quantifier ces versions concaténées, soit effectuer la concaténation après l'étape de quantification. Nous choisissons ici la première option, qui présente l'avantage, pour un nombre fixé k de mots, de laisser toute lattitude au quantificateur utilisé pour répartir les vecteurs concaténés en k classes. En effet, supposons que l'on choisisse la deuxième option. Si l'on souhaite fixer $n = 20$ et que l'on utilise une valeur de k de l'ordre du million, il faut, pour chaque quantificateur utilisé, utiliser un nombre de mots égal à k' tel que $k'^n \simeq k$, ce qui nous donne $k' = 2$. Ainsi chaque quantificateur serait très grossier et peu informatif quant au descripteur qu'il quantifie. Le processus résultant de ce choix est présenté en figure 4.2.

La comparaison entre deux séquences se fait ensuite de manière tout à fait analogue à la comparaison utilisée pour les sacs de mots n'intégrant pas d'information temporelle. Cette méthode est totalement paramétrée par la détermination des valeurs de k et de n à utiliser.

Séquences symboliques illustratives

Pour les séquences symboliques utilisées dans ce chapitre, l'utilisation de bi-grammes (n-grammes avec $n = 2$), classera en première position la séquence ABCDE, suivies des séquences AABBCCDDEE (car celle-ci contient les n-grammes AB, BC et CD, mais aussi d'autres n-grammes qui sont absents de la requête, comme AA) et ABCXXXDF (pour les même raisons), et enfin les séquences EDCBA et BXAX-CXXFXD qui ne contiennent aucun n-gramme commun avec la requête. On voit donc ici les améliorations introduites par rapport aux sacs de mots sans information temporelle qui ne pouvaient pas distinguer les séquences ABCDE et EDCBA.

4.2.3 Représentation avec forte notion de séquence : l'alignement dynamique

Nous avons eu l'occasion dans les chapitres précédents de détailler les principes de fonctionnement de l'alignement dynamique. Il est important de préciser que l'on considère ici la DTW exacte et non une approximation comme on en a présentée au chapitre 2.

La DTW est alors vue comme un moyen d'aligner deux séquences entre elles en autorisant des distorsions locales de l'axe du temps, sous la forme d'insertions et de suppressions. La mesure de similarité résultante correspond à la somme des dissimilarités observées le long du chemin d'alignement optimal.

L'interrogation de la base consiste à comparer la séquence requête avec chacune des séquences de la base en utilisant la DTW et de retourner les séquences de la base dans l'ordre croissant de leurs mesures de DTW.

Séquences symboliques illustratives

Il est important de noter que, si l'on utilise ici des séquences symboliques pour leurs vertus illustratives, la DTW opérant dans le domaine numérique est sensiblement différente de sa version symbolique. Alors que pour la DTW symbolique, il s'agit d'assigner un coût fixe aux opérations d'insertion, de suppression et de remplacement, cette notion disparaît lorsque l'on utilise la DTW numérique qui, elle, cherche un chemin de coût minimal au sein de la matrice des distances entre les séquences. Cette différence est de taille et nous aurons l'occasion d'y revenir dans le bilan de ce chapitre.

L'alignement dynamique classera, pour la requête ABCDF, les séquences de la base dans l'ordre suivant : ABCXXXDF, devant ABCDE, à égalité avec AABBCCDDEE et suivi de EDCBA et de BXAXCXXFXD.

On remarque ici un premier effet de l'utilisation de la DTW numérique : la première séquence retournée contient des insertions de symboles X qui ne sont absolument pas pénalisées par la DTW numérique, alors qu'elles le seraient si l'on utilisait une DTW symbolique. Pour les mêmes raisons, les deux séquences retournées ensuite le sont avec un score égal, alors que si l'on considérait un alignement dynamique basé symbole, la séquence AABBCCDDEE serait pénalisée par les opérations de suppression nécessaires à son alignement avec la séquence requête. Au-delà de ce fait, on note qu'en utilisant cette méthode, l'accent est mis sur la comparaison des

structures temporelles des séquences, les séquences retournées aux premières positions étant celles dont la structure est cohérente avec celle de la requête.

4.3 Expériences

Nous menons la comparaison de ces différentes techniques sur deux jeux de données correspondant, comme précisé plus haut dans ce chapitre, à des domaines où la DTW est utilisée dans la littérature. Dans les deux cas, nous fixons le paramètre k à la valeur 100 000. Cette valeur est justifiée par notre volonté d'éviter les écueils résultant tant de l'utilisation de trop petites valeurs que de trop grandes valeurs, comme expliqué plus haut.

Pour les deux expériences conduites, n'ayant pas de segmentation disponible, nous avons découpé les flux en extraits de 5 secondes, pour correspondre à la problématique de courts extraits exposée par [Casey 06a]. De plus, dans les deux cas, comme dans [Casey 06a], la taille des requêtes est de l'ordre de la taille des séquences de la base. Une séquence de la base est considérée comme similaire à une requête si elle inclut au moins la moitié d'une occurence de cette requête. De plus, n'ayant pas de bornes fiables sur les séquences ainsi obtenues, on utilise la version de la DTW présentée en section 1.2.2 de ce document, pour laquelle les contraites aux bords ont été relaxées.

4.3.1 Recherche de *jingles* vidéo

Cette première expérience consiste à rechercher des quasi-réplicats d'une requête dans un flux, organisé comme précisé plus haut. N'ayant pas accès aux données utilisées dans [Casey 06a], le but de cette expérience est de fournir un cadre d'étude analogue où les différentes occurences d'une requête dans la base ne sont pas altérées temporellement.

Le corpus vidéo utilisé est constitué de 2 jours d'enregistrement de la chaîne de télévision France 2. Les requêtes utilisées correspondent à 33 *jingles* annonçant un espace publicitaire qui se trouvent répétés dans le flux. Pour ce qui est des approches par sacs de mots avec ou sans n-grammes, les vocabulaires sont appris sur un flux d'une journée (distincte des 2 jours considérés pour le jeu de test) de la même chaîne de télévision.

Les enregistrements ayant été réalisés à l'aide d'une carte d'acquisition analogique, les principales variations observées entre différentes occurences d'un même *jingle* se trouvent dans le contenu des images, susceptible d'être bruité à cause d'une mauvaise qualité de réception, plutôt que dans leur enchaînement, puisque l'axe du temps, comme expliqué plus haut, n'a pas été altéré.

Les descripteurs utilisés sont des descripteurs globaux proposés par Naturel et Gros [Naturel 05] dans le cadre de la recherche de séquences dupliquées dans les flux télévisuels. Le calcul de ces descripteurs consiste en la binarisation des coefficients les plus significatifs de la transformée cosinus discrète (ou DCT pour *Discrete Cosine Transform*) d'une image. On obtient alors, pour chaque image, un vecteur binaire de taille 64.

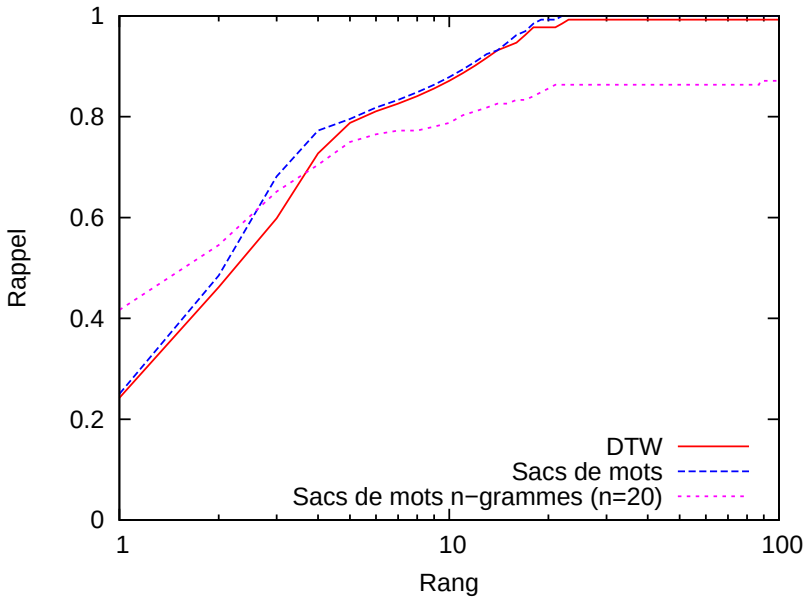


FIGURE 4.3 – *Performances comparées de la DTW et des représentations par sacs de mots pour la recherche de* jingles *vidéo.*

La figure 4.3 montre que l'approche par sac de mots obtient des résultats légèrement supérieurs à la DTW, bien que n'intégrant aucune information temporelle. On remarque en outre que l'utilisation de n-grammes dans ce contexte n'est pas souhaitable, puisque les résultats s'en trouvent détériorés (pour des raisons de lisibilité, nous ne reportons que les résultats pour $n = 20$ mais avons testé pour des valeurs allant de 10 à 100 et en tirons les mêmes conclusions).

Le meilleur comportement des sacs de mots par rapport à la DTW s'explique par le fait que, pour cette application, les déformations temporelles locales sont inexistantes, puisqu'il s'agit de *jingles* répétés à l'identique à plusieurs moments dans le flux. Ainsi, la DTW, peu robuste à l'insertion de grandes valeurs dans la matrice de distance (qui peuvent être dues au bruit d'acquisition des vidéos) est plus sensible aux distorsions existantes ici alors que sa modélisation fine de l'axe du temps est ici de peu d'utilité. Au contraire, pour les approches par sac de mots, le bruit contenu dans les descripteurs est soit absorbé par l'étape de quantification soit, dans le pire des cas, peu influent sur la mesure de similarité finale puisqu'il ne concerne qu'un vote et que tous les votes ont la même pondération.

D'autre part, les moindres performances du modèle utilisant les sacs de mots n-grammes peuvent s'expliquer par le fait que l'on utilise ici un nombre de mots k fixé. Ce choix se justifie dans la mesure où l'on souhaite comparer les deux méthodes utilisant des sacs de mots à quantité d'information égale, et donc à valeur de k fixée. Néanmoins, cela a pour effet, lorsque la dimension des données augmente comme c'est le cas en concaténant des vecteurs pour en faire des n-grammes, de fournir une représentation des données moins précise (l'erreur de quantification obtenue en utilisant k mots dans un espace de dimension $(n \times \text{dim})$ sera nécessairement plus grande que si l'on est dans un espace de dimension dim) et de dégrader les résultats.

On voit donc ici une limite fondamentale de la DTW numérique qui, si elle est

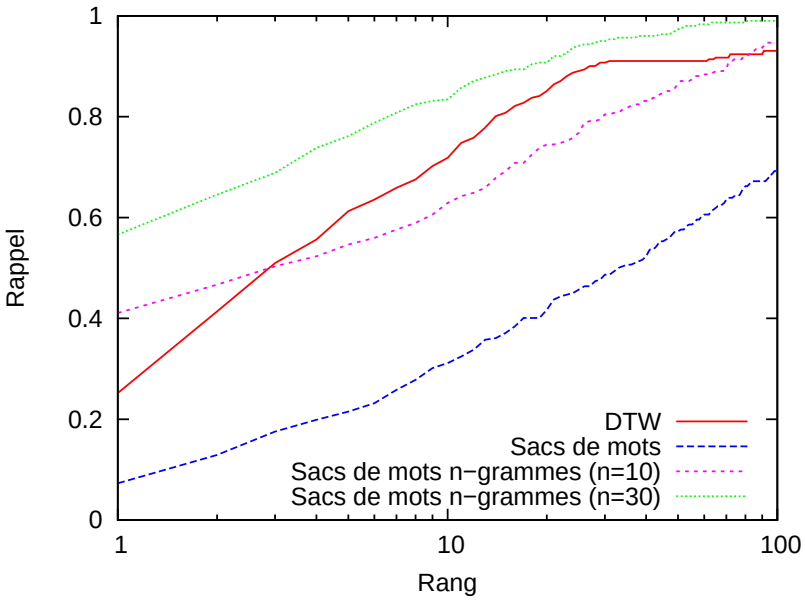


FIGURE 4.4 – *Performances comparées de la DTW et des représentations par sacs de mots pour la recherche de mots parlés.*

adaptée à des opérations d'insertion ou de suppression locales dans les séquences, peut, parce qu'elle effectue une somme le long d'un chemin, retourner une grande valeur à cause d'une sous-partie pour laquelle on ne trouve pas de correspondance, fût-elle courte devant la taille des séquences considérée.

4.3.2 Recherche de mots parlés

Pour cette deuxième expérience, nous utilisons le jeu de données de [Muscariello 09], qui est de même nature que celui de [Fraihat 10]. Celui-ci est donc exemplaire d'un cas où la dimension temporelle des séquences est altérée par la nature même des répétitions dans le flux qui sont des occurences d'un même mot énoncé par différents locuteurs dans différents contextes.

Ce jeu de données est issu d'enregistrements de la chaîne de radio France Info. L'apprentissage des vocabulaires correspondant aux approches par sacs de mots est réalisé sur un ensemble de 8 heures d'enregistrement alors que le jeu de test est fait d'une heure et est totalement distinct du jeu d'apprentissage. Les requêtes considérées sont les enregistrements de 78 mots répétés dans le flux par différents locuteurs.

Les descripteurs utilisés correspondent aux 12 premiers coefficients cepstraux sur une échelle de fréquence Mel (MFCC) [Davis 80], calculés toutes les 10 millisecondes sur une fenêtre de 20 millisecondes.

Les résultats obtenus, présentés en figure 4.4, montrent que, même pour cette application où les différentes versions d'un mot sont susceptibles d'être énoncées avec des dynamiques relativement différentes selon les locuteurs et le contexte, l'utilisation de sacs de mots permet d'obtenir de bons résultats. Toutefois, pour obtenir des résultats comparables (voire meilleurs) à ceux obtenus par la DTW, il est nécessaire de considérer des n-grammes assez longs, permettant de retenir une information

temporelle locale conséquente.

De manière qualitative, on remarque en écoutant les faux positifs produits par l'approche par sacs de mots qu'il s'agit de mots contenant les mêmes syllabes que la requête. Par exemple, pour une requête qui consiste en l'énonciation du prénom "Bernard", l'extrait de la base classé en première position correspond au script "Seul le fou qui débarque, Pierre Vaneck, explose ce confort d'une tristesse inouïe". On remarque que dans cet extrait, tous les phonèmes de la requête sont présents (le nom "Vaneck" étant prononcé de telle façon que le son [v] initial est très difficilement distinguable du son [b]). On comprend alors que l'utilisation de n-grammes permet de se limiter aux mots conservant un certain enchaînement des syllabes, et d'obtenir ainsi de meilleurs résultats. Nous présentons les résultats pour des valeurs de n égales à 10 et 30, correspondant respectivement à des voisinages temporels de 100 et 300 millisecondes.

On note également que la pénalisation de la diagonale, préconisée par Fraihat et Glotin [Fraihat 10] ne semble pas opportune ici, comme le montre la figure 4.5. Cela s'explique par le fait que nous utilisons une DTW dont les bornes ont été relaxées, ce qui n'est pas le cas des travaux de Fraihat et Glotin, pour lesquels l'alignement entre une sous-séquence et un extrait de la base consiste en trois parties : (i) une série d'insertions permettant d'aligner les instants de début de la requête et de son occurence dans le segment de la base, (ii) une mise correspondance des deux occurences du mot considéré qui consiste en un chemin dont l'allure générale est diagonale et enfin (iii) une nouvelle série d'insertions permettant d'aligner les instants de fin de la requête et de son occurence dans le segment de la base. Ainsi, les étapes (i) et (iii) impliquent un grand nombre de transitions non-diagonales auxquelles profitent la pénalisation des transitions diagonales. Ce problème étant réglé dans notre cas par l'utilisation de contraintes aux bords relaxées, il est préferable de ne pas utiliser une telle pénalisation.

On peut enfin noter que les descripteurs utilisés ici ne sont pas identiques à ceux utilisés dans [Fraihat 10] (qui montre qu'une version quantifiée des descripteurs MFCC est plus opportune pour cette application). Nous pensons toutefois que cela n'est pas de nature à altérer nos conclusions dans la mesure où les mêmes descripteurs sont utilisés pour les approches par sacs de mots auxquelles est comparée la DTW.

Ces résultats montrent que, même pour des applications où les variations locales de tempo existent, le grand nombre de déformations temporelles autorisées par la DTW (et donc le grand nombre de chemins possibles au sein de la matrice de similarité utilisée) offre trop de degrés de liberté, permettant d'aligner des séquences qui ne devraient pas l'être. Il semble alors préférable de considérer un modèle vectoriel dans lequel l'information temporelle est incorporée dans les mots utilisés.

4.4 Bilan

Dans les précédents chapitres de ce document, nous nous sommes intéressés à des cadres où il a été montré que l'utilisation de l'alignement dynamique était utile. Nous montrons dans ce chapitre que ce n'est pas toujours le cas et tentons maintenant d'expliquer cette conclusion, pour en extraire des caractéristiques propres

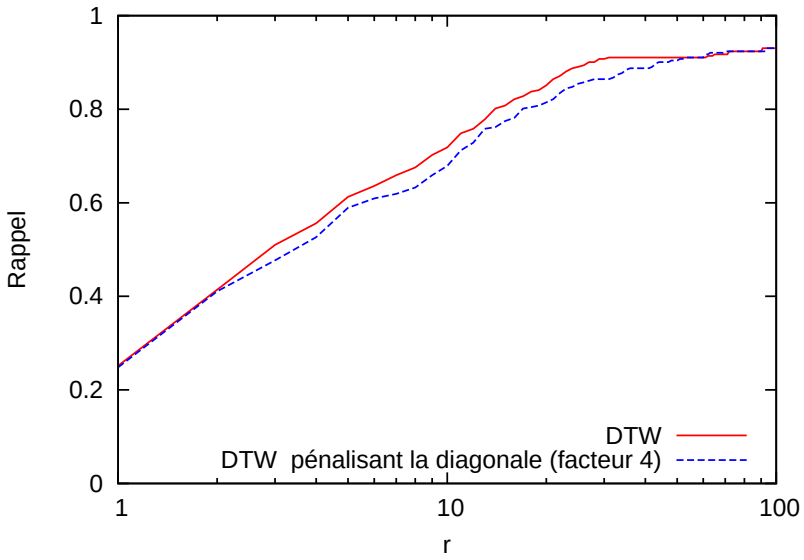


FIGURE 4.5 – *Performances comparées de la DTW avec et sans pénalisation des chemins diagonaux pour la recherche de mots parlés.*

aux problèmes étudiés qui puissent justifier ou non de l'utilisation de la DTW.

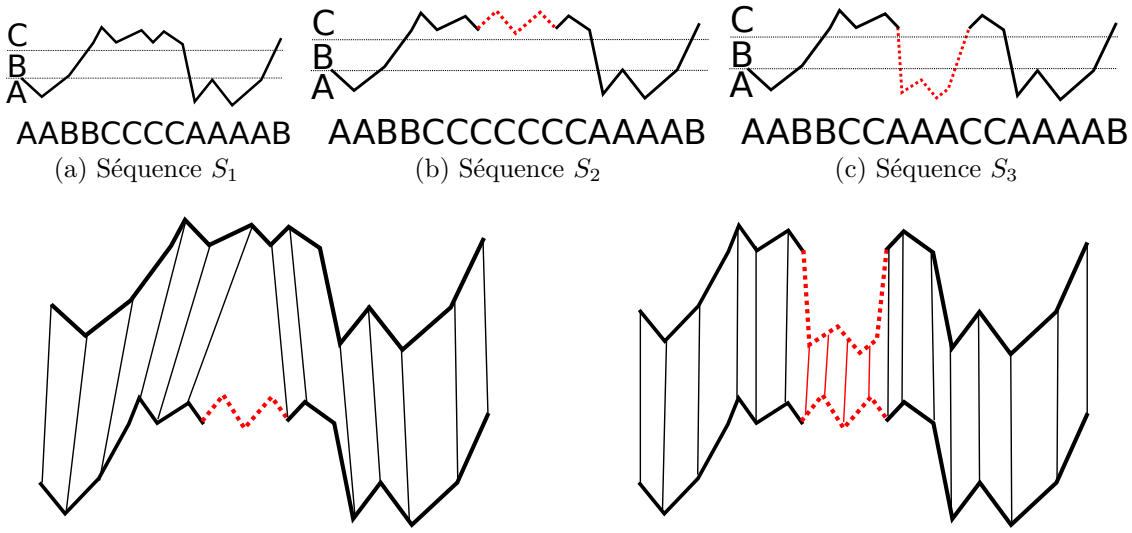
Tout d'abord, il est important de remarquer que la DTW est une mesure de similarité relativement laxiste. En effet, étant données deux séquences on cherche à trouver un chemin dans une matrice de similarité, lequel correspond à un alignement optimal entre les séquences. La recherche de ce chemin offre un grand nombre de degrés de liberté dans le choix des chemins et, ainsi, il est possible de trouver des correspondances entre séquences *a priori* relativement différentes.

De plus, comme illustré en figure 4.6, en cherchant à aligner globalement deux séquences, on obtient un chemin dont le coût est susceptible d'être fortement pénalisé par la présence de sous-parties qui ne se correspondent pas dans les séquences. Ce phénomène est moins problématique lorsque l'on considère les sacs de mots car on ne modélise pas la différence entre les mots. Ainsi, si les sous-séquences en question contiennent des mots différents, la mesure de la similarité entre les séquences sera légèrement bruitée mais ne pourra en aucun cas être aussi fortement impactée par ces sous-séquences différentes que peut l'être la DTW.

On peut extraire de ces expériences le sentiment que, pour des applications qui relèvent de la recherche de quasi-réplicats, l'utilisation de l'alignement dynamique ne semble pas opportune, comme le montrent les résultats obtenus pour la recherche de *jingles* vidéos où une méthode n'intégrant aucune information temporelle obtient de meilleurs résultats que l'alignement dynamique.

On voit, par contre, que lorsque l'axe du temps est plus altéré ou que l'information contenue dans une trame isolée n'est pas suffisamment discriminante, il est nécessaire de prendre la notion de séquence en compte. Toutefois, cette prise en compte ne doit pas nécessairement passer par l'utilisation de la DTW, comme nous le montre l'expérience réalisée sur la recherche de mots parlés, pour laquelle l'utilisation de n-grammes s'avère suffisante.

Il est possible d'extraire une caractéristique commune aux données pour lesquelles



(a) Séquence S_1 (b) Séquence S_2 (c) Séquence S_3

(d) DTW(S_1, S_2) (e) DTW(S_3, S_2)

FIGURE 4.6 – *Comportement de la DTW et des approches par sacs de mots pour deux opérations d'édition.* Dans les deux cas les segments gras correspondent aux séquences et les segments fins illustrent les mises en correspondances opérées par la DTW. Pour la séquence S_2, on a inséré une sous-séquence, représentée en trait pointillé rouge, au milieu de la séquence S_1. On voit qu'alors la DTW est capable de mettre en correspondance les éléments adéquats. En revanche, lorsque l'on compare les séquences S_2 et S_3, elles ont une sous-partie différente, également signalée en trait pointillé rouge. Dans ce cas, la DTW cherche à mettre en relation les deux sous-séquences qui ne se correspondent pas (pour la séquence du haut, les valeurs sont faibles alors que pour la séquence du bas, elles sont élevées) et le coût total du chemin utilisé est alors fortement pénalisé par cet artefact. Si l'on s'intéresse maintenant à une approche par sac de mots sans n-gramme, on a les vecteurs de fréquence suivants pour les trois séquences considérées : $V_f(S_1) = (A : 6, B : 3, C : 4)$, $V_f(S_2) = (A : 6, B : 3, C : 7)$ et $V_f(S_3) = (A : 9, B : 3, C : 4)$. On remarque que si ces vecteurs de fréquence ne sont pas égaux, ils sont relativement proches. De même, si l'on utilise des n-grammes, les sous-mots de taille n extraits au début et à la fin des séquences qui sont similaires pour les 3 séquences se retrouveront dans le vecteur de fréquence et on obtiendra donc une similarité élevée entre les séquences.

l'utilisation de l'alignement dynamique est opportune. Il s'agit là d'applications pour lesquelles les notions d'insertion et de suppression dans les séquences ont un sens physique, comme pour les données issues de la génomique ou pour la recherche de reprises musicales. Dans le premier cas, ces opérations d'édition correspondent à des mutations alors que dans le second, il peut s'agir d'insertions de notes dans un morceau (comme par exemple des arpèges). Quoi qu'il en soit, ces deux cadres se distinguent de la reconnaissance de mots parlés où, d'un locuteur à l'autre, il n'y aura pas de syllabe ajoutée dans un mot.

Ces considérations nous permettent d'étendre les conclusions ici présentées aux cas de quasi-réplicats ayant subi des opérations de post-production. Les opérations de montage opérant à un grain beaucoup moins fin que la description, il nous paraît important de distinguer deux niveaux d'analyse pour de telles applications. Si l'on dispose d'un outil de segmentation, il convient alors de comparer les différents segments obtenus sans alignement dynamique et, éventuellement, d'utiliser l'alignement dynamique ensuite pour aligner les segments entre eux. Si l'on considère maintenant les cas de variations du tempo de l'enregistrement, par exemple dans les cas de ralentis pour les vidéos sportives, l'utilisation de l'alignement dynamique nous paraît inadapté pour les raisons évoquées plus haut. Il convient alors de dériver des métriques de similarité offrant moins de degrés de liberté que la DTW adaptées aux altérations de l'axe du temps que l'on souhaite observer.

Bibliographie

[Casey 06] M. Casey & M. Slaney. *The Importance of Sequences in Musical Similarity*. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2006.

[Chantamunee 08] S. Chantamunee & Y. Gotoh. *University of Sheffield at TRECVID 2008 : Rushes Summurization and Video Copy Detection*. In Proceedings of the TRECVID Workshop, 2008.

[Davis 80] S. Davis & P. Mermelstein. *Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences*. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 28, no. 4, pages 357–366, 1980.

[Fraihat 10] S. Fraihat & H. Glotin. *Indexation rapide de documents audio par traitement morphologique de la parole*. Ingénierie des systèmes d'information, vol. 15, pages 29–48, 2010.

[Muscariello 09] A. Muscariello, G. Gravier & F. Bimbot. *Variability Tolerant Audio Motif Discovery*. In Proceedings of the International Conference on Multimedia Modeling, january 2009.

[Naturel 05] X. Naturel & P. Gros. *A fast shot matching strategy for detecting duplicate sequences in a television stream*. In Proceedings of the International Workshop on Computer Vision Meets Databases, pages 21–27, 2005.

[Salton 75] G. Salton, A. Wong & C. S. Yang. *A vector space model for automatic indexing*. Communications of the ACM, vol. 18, pages 613–620, 1975.

[Sivic 03] J. Sivic & A. Zisserman. *Video Google : a Text Retrieval Approach to Object Matching in Videos*. In Proceedings of the International Conference on Computer Vision, volume 2, pages 1470–1477, 2003.

Conclusion

L'objectif de ces travaux était d'aborder l'utilisation de métriques fines pour la comparaison de séquences.

Nous nous sommes tout d'abord intéressés à des domaines pour lesquels l'état-de-l'art indique que des mesures de similarité dérivées de l'alignement dynamique étaient adaptées. Dans ces domaines, nous avons proposé deux systèmes d'indexation, l'un adapté à l'utilisation de la DTW et l'autre spécifique au cas de la recherche de reprises musicales.

Dans le premier cas, notre proposition consiste à permettre la recherche de plus proches voisins approximatifs dans le but d'accélérer significativement les temps de requête. Dans ce cadre, nous avons également proposé un moyen d'équilibrer les arbres d'indexation considérés en utilisant une représentation qui dépende des données plutôt que de considérer des hypothèses peu réalistes sur la distribution des données, comme il était jusqu'alors proposé.

Notre deuxième système d'indexation est donc appliqué au cas particulier de la recherche de reprises musicales. Notre proposition dans ce domaine est faite de deux étapes. La première consiste à effectuer une recherche efficace de vecteurs isolés alors que la deuxième utilise les résultats de cette première étape pour raffiner les correspondances à l'aide d'un algorithme d'alignement dynamique. Les résultats obtenus montrent que cette approche permet à la fois d'accélérer les temps de calcul et la pertinence des résultats retournés.

Nous avons ensuite cherché à identifier les domaines dans lesquels de telles métriques étaient nécessaires. Nous avons montré que pour certaines applications, comme par exemple la recherche de quasi-réplicats, le trop grand nombre de degrés de liberté offerts par des mesures de similarité comme la DTW avait un impact négatif sur la qualité des résultats et qu'alors, des approches utilisant peu ou pas du tout d'information temporelle étaient plus appropriées.

De manière générale, ces travaux ont permis de mieux comprendre les caractéristiques intrinsèques aux données qui justifient l'utilisation de la DTW. Il en ressort que ces caractéristiques s'expriment en termes de variabilité temporelle des extraits à retrouver. En effet, si les données considérées sont telles que deux occurrences d'une même séquence sont susceptibles de différer par la présence d'insertions locales d'information, il sera alors bon de considérer les méthodes d'alignement dynamique pour la recherche. Dans ce cas, si l'on cherche à étudier la similarité globale entre séquences, il sera bon d'utiliser des techniques telles que celles présentées au chapitre 2, alors que si l'on cherche à isoler des sous-séquences similaires, on devra se tourner vers des techniques qui commencent par localiser les correspondances avant d'autoriser l'utilisation d'algorithmes d'alignement coûteux, comme proposé au chapitre 3. Dans

le cas contraire, il est préférable d'avoir recours aux techniques usuelles de recherche de plus proches voisins locaux, quitte à avoir recours à une post-vérification de la cohérence temporelle des résultats retournés.

Nous avons déjà présenté des perspectives immédiates de nos travaux dans les chapitres correspondants de ce document. Nous tentons ici d'adopter un point de vue plus global pour proposer des pistes de recherche relatives au domaine de la recherche d'information dans les bases de séquences.

Tout d'abord, nous avons mis en évidence au chapitre 4 les limites de l'alignement dynamique pour la comparaison de séquences. Celles-ci ne doivent pas, à notre sens, être interprétées comme la nécessité absolue de se débarrasser de l'information temporelle pour comparer des séquences mais plutôt comme le fait que la DTW n'est peut-être pas la meilleure métrique. Bien que celle-ci soit largement utilisée, il semble nécessaire de développer de nouveaux modèles de déformations plus adaptés à la comparaison de séquences peu altérées. D'une part, si l'on connaît la nature des altérations auxquelles on souhaite être robuste, il est judicieux de construire une métrique de comparaison adaptée, offrant moins de degrés de liberté que la DTW ou, sur le modèle de la DTW symbolique, pénalisant explicitement les opérations locales d'insertion et de suppression. D'autre part, nous avons vu que l'un des problèmes de la DTW est sa sensibilité aux grandes valeurs dans la matrice de distances. Il serait alors intéressant d'étudier les performances d'algorithmes adaptés limitant l'impact de ces grandes valeurs, par exemple en utilisant des fonctions de coût concaves.

Ensuite, au-delà des techniques de recherche de séquence similaire abordées dans ce manuscrit, il existe une problématique importante pour des applications comme l'organisation de flux, qui est la découverte de motifs. Dans ce domaine, il semble important de disposer de plusieurs niveaux d'analyse car il n'est pas possible de comparer finement toutes les sous-parties du flux entre elles. À la lumière des résultats présentés dans ce manuscrit, il semble opportun de débuter avec une phase de filtrage n'intégrant pas ou peu d'information temporelle dans le but d'extraire les paires de segments de flux ressemblantes. Ensuite, si nécessaire, il est possible d'affiner cette information à l'aide de métriques de comparaison intégrant plus finement l'axe temporel.

En outre, comme le montre l'exemple de la recherche de reprises musicales, les cas où l'utilisation de l'alignement dynamique pour la comparaison de séquences est adaptée correspondent à des problématiques parfois très proches de la classification. Il semblerait alors adapté d'étudier la faisabilité d'algorithmes de classification utilisant l'alignement dynamique comme métrique, plutôt que les classiques distances de Minkowski. Il a par exemple été développé des noyaux à base d'alignement dynamique pour les célèbres machines à vecteurs supports. Toutefois, leur utilisation reste très marginale, notamment à cause du coût de calcul prohibitif lié à leur apprentissage. Une option envisageable pour faire face à ce problème serait d'adapter les techniques de sélection de candidats pertinents (en l'occurence, de vecteurs support potentiels) présentées au chapitre 2 de ce manuscrit. Si l'on s'intéresse à des techniques d'apprentissage non supervisé, en outre, il serait judicieux de dériver un algorithme tel que le k-means utilisant la DTW comme métrique. Pour ce faire, il est nécessaire de définir une notion de barycentre adapté à cette métrique, ce qui constitue en soi un axe de recherche important à nos yeux.

Table des figures

Liste des tableaux

Bibliographie Générale

[Altschul 90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers & D. J. Lipman. *Basic Local Alignment Search Tool.* Journal of Molecular Biology, vol. 215, no. 3, pages 403–410, 1990.

[Altschul 97] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller & D.J. Lipman. *Gapped BLAST and PSI-BLAST : a New Generation of Protein Database Search Programs.* Nucleic Acids Research, vol. 25, no. 17, pages 3389–3402, 1997.

[Arthur 07] D. Arthur & S. Vassilvitskii. *k-means++ : the advantages of careful seeding.* In Proceedings of the annual ACM-SIAM symposium on Discrete algorithms, pages 1027–1035, 2007.

[Ben 04] M. Ben, M. Betser, F. Bimbot & G. Gravier. *Speaker Diarization Using Bottom-up Clustering Based on a Parameter-derived Distance Between GMMs.* In Proceedings of the International Conference on Spoken Language Processing, pages 2329–2332, 2004.

[Berrani 04] S. A. Berrani. *Recherche approximative de plus proches voisins avec contrôle probabiliste de la précision ; application à la recherche d'images par le contenu.* PhD thesis, Université de Rennes 1, 2004.

[Camerra 10] A. Camerra, T. Palpanas, J. Shieh & E. J. Keogh. *iSAX 2.0 : Indexing and Mining One Billion Time Series.* In Proceedings of the IEEE International Conference on Data Mining, 2010.

[Casey 06a] M. Casey & M. Slaney. *The Importance of Sequences in Musical Similarity.* In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2006.

[Casey 06b] M. Casey & M. Slaney. *Song intersection by approximate nearest neighbor search.* In Proceedings of ISMIR, pages 144–149, 2006.

[Casey 07] M. Casey & M. Slaney. *Fast recognition of remixed music audio.* In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pages 1425–1428, 2007.

[Casey 08] M. Casey, C. Rhodes & M. Slaney. *Analysis of minimum distances in high-dimensional musical spaces*. IEEE Transactions on Audio, Speech, and Language Processing, vol. 16, no. 5, pages 1015–1028, 2008.

[Chantamunee 08] S. Chantamunee & Y. Gotoh. *University of Sheffield at TRECVID 2008 : Rushes Summurization and Video Copy Detection*. In Proceedings of the TRECVID Workshop, 2008.

[Chen 04] L. Chen & R. Ng. *On the Marriage of Lp-norms and Edit Distance*. In Proceedings of the International Conference on Very Large Data Bases, pages 792–803, 2004.

[Datar 04] M. Datar, N. Immorlica, P. Indyk & V.S. Mirrokni. *Locality-sensitive hashing scheme based on p-stable distributions*. In Proceedings of the Symposium on Computational Geometry, pages 253–262, 2004.

[Davis 80] S. Davis & P. Mermelstein. *Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences*. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 28, no. 4, pages 357–366, 1980.

[Di Buccio 10] E. Di Buccio, N. Montecchio & N. Orio. *FALCON : FAst Lucene-based Cover sOng identification*. In Proceedings of the ACM International conference on Multimedia, pages 1477–1480, 2010.

[Douze 08] M. Douze, A. Gaidon, H. Jégou, M. Marszałek & C. Schmid. *INRIA-LEARs video copy detection system*. In Proceedings of the TRECVID Workshop, 2008.

[Faloutsos 94] C. Faloutsos, M. Ranganathan & Y. Manolopoulos. *Fast Subsequence Matching in Time-series Databases*. In Proceedings of the ACM SIGMOD Conference, pages 419–429, 1994.

[Fraihat 10] S. Fraihat & H. Glotin. *Indexation rapide de documents audio par traitement morphologique de la parole*. Ingénierie des systèmes d'information, vol. 15, pages 29–48, 2010.

[Haitsma 01] J. Haitsma, T. Kalker & J. Oostveen. *Robust Audio Hashing for Content Identification*. In Proceedings of the International Workshop on Content-Based Multimedia Indexing, 2001.

[Henikoff 92] S. Henikoff & J.G. Henikoff. *Amino Acid Substitution Matrices from Protein Blocks*. Proceedings of the National Academy of Sciences of the United States of America, vol. 89, no. 22, pages 10915–10919, 1992.

[Jégou 10] H. Jégou, M. Douze & C. Schmid. *Improving bag-of-features for large scale image search*. International Journal of Computer Vision, vol. 87, no. 3, pages 316–336, 2010.

[Jégou 11a] H. Jégou, M. Douze & C. Schmid. *Product quantization for nearest neighbor search.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 1, pages 117–128, 2011.

[Jégou 11b] H. Jégou, R. Tavenard, M. Douze & L. Amsaleg. *Searching in one billion vectors : re-rank with source coding.* In IEEE International Conference on Acoustics, Speech and Signal Processing, 2011.

[Keogh 01] E. Keogh, K. Chakrabarti, M. Pazzani & S. Mehrotra. *Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases.* Knowledge And Information Systems, vol. 3, no. 3, pages 263–286, 2001.

[Keogh 05] E. Keogh & C.A. Ratanamahatana. *Exact Indexing of Dynamic Time Warping.* Knowledge And Information Systems, vol. 7, no. 3, pages 358–386, 2005.

[Keogh 06] E. Keogh, X. Xi, L. Wei & C. A. Ratanamahatana. *The UCR Time Series Classification/Clustering Homepage : www.cs.ucr.edu/~eamonn/time_series_data/*, 2006.

[Lin 07] J. Lin, E. Keogh, L. Wei & S. Lonardi. *Experiencing SAX : a Novel Symbolic Representation of Time Series.* Data Mining and Knowledge Discovery, vol. 15, no. 2, pages 107–144, 2007.

[Manber 90] U. Manber & G. Myers. *Suffix Arrays : a New Method for On-line String Searches.* In Proceedings of the annual ACM-SIAM symposium on Discrete algorithms, pages 319–327, 1990.

[McCreight 76] E. McCreight. *A Space-Economical Suffix Tree Construction Algorithm.* Journal of the ACM, vol. 23, pages 262–272, 1976.

[Muja 09] M. Muja & D. G. Lowe. *Fast approximate nearest neighbors with automatic algorithm configuration.* In Proceedings of the International Conference on Computer Vision Theory and Applications, 2009.

[Muscariello 09] A. Muscariello, G. Gravier & F. Bimbot. *Variability Tolerant Audio Motif Discovery.* In Proceedings of the International Conference on Multimedia Modeling, january 2009.

[Naturel 05] X. Naturel & P. Gros. *A fast shot matching strategy for detecting duplicate sequences in a television stream.* In Proceedings of the International Workshop on Computer Vision Meets Databases, pages 21–27, 2005.

[Puglisi 06] S. Puglisi, W. Smyth & A. Turpin. *Inverted Files Versus Suffix Arrays for Locating Patterns in Primary Memory.* In Proceedings of the International Conference on String Processing and Information Retrieval, pages 122–133, 2006.

[Ratanamahatana 04] C.A. Ratanamahatana & E. Keogh. *Everything you know about dynamic time warping is wrong.* In Proceedings of the Workshop on Mining Temporal and Sequential Data, 2004.

[Sakoe 78] H. Sakoe & S. Chiba. *Dynamic Programming Algorithm Optimization for Spoken Word Recognition.* IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 26, pages 43–49, 1978.

[Salton 75] G. Salton, A. Wong & C. S. Yang. *A vector space model for automatic indexing.* Communications of the ACM, vol. 18, pages 613–620, 1975.

[Salvador 04] S. Salvador & P. Chan. *FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space.* In Proceedings of the KDD Workshop on Mining Temporal and Sequential Data, pages 70–80, 2004.

[Serrà 09] J. Serrà, X. Serra & R.G. Andrzejak. *Cross Recurrence Quantification for Cover Song Identification.* New Journal of Physics, vol. 11, 2009.

[Shieh 08] J. Shieh & E. Keogh. *iSAX : Indexing and Mining Terabyte Sized Time Series.* In Proceedings of the ACM International Conference on Knowledge Discovery and Data mining, pages 623–631, 2008.

[Sivic 03] J. Sivic & A. Zisserman. *Video Google : a Text Retrieval Approach to Object Matching in Videos.* In Proceedings of the International Conference on Computer Vision, volume 2, pages 1470–1477, 2003.

[Tavenard 09] R. Tavenard, L. Amsaleg & G. Gravier. *Model-Based Similarity Estimation of Multidimensional Temporal Sequences.* Annals of Telecommunications, vol. 64, pages 381–390, 2009.

[Tavenard 11] R. Tavenard, H. Jégou & L. Amsaleg. *Balancing clusters to reduce response time variability in large scale image search.* In IEEE Workshop on Content-Based Multimedia Indexing, 2011.

[Ukkonen 95] E. Ukkonen. *On-line Construction of Suffix Trees.* Algorithmica, vol. 14, pages 249–260, 1995.

[Wang 06] A. Wang. *The Shazam music recognition service.* Communications of the ACM, vol. 49, pages 44–48, 2006.

[Weiner 73] P. Weiner. *Linear Pattern Matching Algorithms.* In Proceedings of the Annual Symposium on Switching and Automata Theory, pages 1–11, 1973.

[Weiss 08] Y. Weiss, A. Torralba & R. Fergus. *Spectral Hashing.* In Proceedings of the Annual Conference on Neural Information Processing Systems, 2008.

[Yang 01] C. Yang & K. I. Lin. *An Index Structure for Efficient Reverse Nearest Neighbor Queries*. In Proceedings of the International Conference on Data Engineering, pages 485–492, 2001.

[Yi 98] B. Yi, H. V. Jagadish & C. Faloutsos. *Efficient Retrieval of Similar Time Sequences Under Time Warping*. In Proceedings of the International Conference on Data Engineering, pages 201–208, 1998.