Université de Montréal

Helix Explorer : Une nouvelle base de données de structures de protéines

par

Mohamed Tikah Marrakchi

Programme de bioinformatique

Faculté des études supérieures

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de M.Sc. en bioinformatique

Août, 2006

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée :

Helix Explorer : Une nouvelle base de données de structures de protéines

présentée par :

Mohamed Tikah Marrakchi

a été évaluée par un jury composé des personnes suivantes :

Dr. Joëlle N. Pelletier, président-rapporteur

Dr. Sylvie Hamel, directeur de recherche

Dr. Andreea-Ruxandra Schmitzer, co-directeur

Dr. François Major, membre du jury

Résumé

L'ingénierie des protéines est une discipline relativement jeune qui bénéficierait grandement de nouveaux outils informatiques pour l'analyse et l'exploitation de l'information contenue dans la base de données de structures macromoléculaires Protein Data Bank (PDB). Dans le présent document, nous introduisons Helix Explorer, une base de données de structures secondaires de protéines, qui offre à son utilisateur[1] de nouveaux moyens de rechercher et d'extraire de l'information sur les structures moléculaires de l'archive PDB. Helix Explorer définit et calcule, pour chaque entrée de la base PDB, des métriques telles que la distance entre deux structures secondaires à l'intérieur d'une même protéine, l'angle entre deux hélices α ou encore la distance d'une structure secondaire à la surface moléculaire de sa protéine. Les fonctionnalités d'Helix Explorer sont fournies à travers une interface Web dynamique qui est publique et dont le but est de faciliter la tâche de conception de nouvelles protéines ou plus largement de rendre plus accessible l'analyse des structures protéiques connues.

Mots-clés : Ingénierie des protéines, Protein Data Bank (PDB), structures secondaires de protéines, bases de données bioinformatiques.

---

[1] L'emploi du masculin ici, comme dans la suite du document, pour désigner des personnes n'a d'autres fins que celle d'alléger le texte.

Abstract

Despite being only at its infancy stage, protein design and engineering is a discipline that would certainly benefit from better exploitation of the fast growing Protein Data Bank (PDB) structures repository. In this document we introduce Helix Explorer, a searchable database of protein secondary structure elements that provides protein structure analysts with new means of querying the PDB. Helix Explorer defines some metrics such as the distance between two protein secondary structures, the angle between two α helices or the distance of protein secondary structures to their molecular surface which it computes for every single entry in the PDB. It also provides, through a publicly available and dynamic web-interface, new querying capabilities that make protein structure analysis more accessible.

Keywords: Protein engineering, Protein Data Bank (PDB), protein secondary structures, bioinformatics databases.

Table des matières

Liste des tableaux

Liste des figures

à mes parents

Remerciements

Avant tout, je tiens à remercier ma directrice de recherche Sylvie Hamel de m'avoir accepté comme étudiant, de m'avoir soutenu et guidé le long de ma maîtrise, de sa bonne humeur et de sa disponibilité indéfectible. Je remercie également ma co-directrice, Andreea Schmitzer de son suivi continu, de ses encouragements et d'avoir toujours trouvé le temps pour la multitude de réunions que nous avons tenues ensemble le long des deux dernières années.

Je remercie les responsables des cours de bioinformatique, Marie Pageau, Gertraud Burger, Sylvie et Hervé Philippe de m'avoir donné l'opportunité de les assister dans les cours et d'autres activités. Ce furent pour moi des expériences très enrichissantes.

Je remercie le comité des bourses biT, celui des bourses FES et mes deux directrices de leur support financier. Ce soutien a rendu possible la réalisation de ce travail. Je remercie aussi tous les membres du laboratoire Ibit de l'ambiance qu'ils ont su préserver. Enfin, je remercie toutes les personnes qui ont lues et relevées les erreurs dans le manuscrit de ce mémoire. Toutes les erreurs qui persistent dans ce document sont bien évidemment de mon entière responsabilité.

Abréviations

ADN	Acide DésoxyriboNucléique
AJAX	Asynchronous JavaScript and XML
ARN	Acide RiboNucléique
ASP	Active Server Pages
BLAST	Basic Local Alignment Search Tool
CGI	Common Gateway Interface
DSSP	Definition of Secondary Structure of Proteins
HTML	HyperText Markup Language
IUCr	International Union of Crystallography
JDBC	Java DataBase Connectivity
JSP	Java Server Pages
mmCIF	MacroMolecular Crystallographic Information File
MSD-EBI	Macromolecular Structure Database-European Bioinformatics Institute
MVC	Model-View-Controller
NCBI	National Center for Biotechnology Information
OpenMMS	Open MacroMolecular Structures
PDB	Protein Data Bank

PDBj	Protein Data Bank Japan
PDBML	Protein Data Bank Markup Language
PHP	PHP: Hypertext Preprocessor
RMN	Résonance Magnétique Nucléaire
SQL	Structured Query Language
TASP	Template-Assembled Synthetic Proteins
wwPDB	World Wide Protein Data Bank
XML	eXtended Markup Language
XSD	XML Schema Definition
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Transformations

Introduction

Comme nombreuses autres disciplines scientifiques, les sciences de la vie ont trouvé dans l'informatique un outil formidable pour explorer de nouvelles voies de recherche. Cette évolution a été possible grâce à deux facteurs. Le premier est l'accumulation rapide de données biologiques qui sont de natures très diverses: certaines sont des données génomiques obtenues du séquençage de génomes entiers dont celui de l'homme récemment complété [1], d'autres, sont des données structurales se rapportant à la structure de molécules trouvées dans les systèmes vivants [2,3], enfin d'autres, sont des données sur les processus biologiques tels que la régulation génétique [4], la cinétique des réactions enzymatiques [5] etc. Le deuxième des deux facteurs est le développement de nouvelles méthodes et techniques computationnelles pour traiter ces données. Cela comprend le développement de nouvelles bases de données et d'algorithmes de recherche dans les données génomiques [6], la conception de logiciels de visualisation et d'analyse de structures moléculaires [7], le développement de nouveaux standards permettant de représenter et d'échanger les données biologiques [8,9] etc.

Si historiquement les premiers outils bioinformatiques à entrer dans la vie quotidienne des chercheurs étaient les bases de données génomiques et les logiciels informatiques permettant la manipulation de ce type de données [10], aujourd'hui, plusieurs indicateurs pointent sur une évolution semblable dans le domaine de la *bioinformatique structurale* [11]. En effet, il suffit d'observer la croissance du nombre de structures de biomolécules nouvellement résolues [12], pour conclure qu'il existe incontestablement un intérêt croissant pour les données de structures moléculaires. Cet intérêt, qui trouve son origine dans plusieurs domaines de recherche, avec en tête celui de l'industrie pharmaceutique [13], est motivé par le lien important qui existe entre la structure et la fonction des protéines et plus généralement celles des biomolécules [14].

Cependant, les données structurales posent aux bio-informaticiens un défi encore plus grand que celui posé par les données génomiques. En effet, un génome n'est au fond rien d'autre qu'une simple séquence (même si cette dernière est parfois très longue) formée de

quatre types de bases. De ce fait, à ces débuts, la recherche dans les bases de données génomiques ou, plus généralement, la manipulation de séquences génomiques a pu largement profiter des développements significatifs qui existaient antérieurement dans le traitement de données textuelles. Malheureusement, la situation est bien différente pour ce qui est des données structurales. Les techniques d'indexation par exemple, qui sont souvent utilisés pour accélérer la recherche dans les bases de données textuelles et récemment, par extension, dans les bases de données génomiques, ne trouvent pas d'application directe dans le cas de bases de données de structures.

D'autre part, un des champs de recherche qui profiterait sûrement de nouveaux outils de recherche dans les bases de données structurales est celui de l'ingénierie des protéines. Ce domaine, qui est encore à ses débuts, a connu ces dernières années des développements importants [15]. En plus d'avoir contribué à la conception de nouvelles protéines, l'ingénierie des protéines a aussi permis, plus généralement, d'étendre la compréhension de la liaison entre la structure et la fonction des protéines [16].

Le travail décrit dans le présent mémoire, part de l'hypothèse qu'une méthode efficace de conception de nouvelles protéines passe par l'exemple, en l'occurrence des exemples de structures protéiques réelles qui ont été auparavant résolues par une des techniques expérimentales communément utilisées telles que la cristallographie à rayons X [17] ou la résonnance magnétique nucléaire (RMN) [18]. Cela signifie que le concepteur d'une protéine non naturelle, sans délaisser les techniques habituelles de simulation basées sur la mécanique et dynamique moléculaire [19] et/ou mécanique quantique [20], peut s'inspirer de structures protéiques connues [21]. Concrètement, cela exige de la part du concepteur de rechercher dans une base de structures protéiques les instances de structures locales[2] similaires à la structure qu'il souhaite développer. Le défi, soulevé précédemment, de l'indexation des données structurales pour rendre possible de telles recherches peut être

---

[2] Par *'locales'* nous entendons les structures situées dans une région bien délimitée de la protéine et qui, souvent, impliquent un petit nombre de structures locales élémentaires (i.e. de structures secondaires)

atténué par une représentation judicieuse des données en question. Dans cette perspective, notre approche, dont nous offrons une implémentation avec l'outil Helix Explorer introduit ici, consiste à représenter chaque structure protéique connue en tant que collection de structures secondaires (qui sont des structures élémentaires locales rencontrées dans les protéines) reliées entre elles par des métriques que l'on a définies. Certes, d'autres outils existants ont tenté d'adresser le même problème de recherche dans les bases de données de structures en s'intéressant aux structures secondaires qui s'y trouvent. En particulier, plusieurs tentatives de classifications (automatiques ou semi-automatiques) de protéines en se basant sur les motifs formées par les structures secondaires qu'elles renferment ont été entreprises avec succès. Les deux exemples classiques de classificateurs de domaines de protéines sont CATH [22] et SCOP [23]. Helix Explorer diffère de ces outils par le fait qu'il permet de faire une recherche plus spécifique à l'aide de séquences peptidiques et permet ensuite le filtrage des résultats obtenus à l'aide de paramètres géométriques bien définis. Nous croyons qu'une telle approche est particulièrement adaptée à la tâche de conception de nouvelles protéines où les moindres détails des structures examinées peuvent se révéler être cruciaux à la conception de la protéine recherchée et à la fonction de cette dernière.

Le présent document a pour but de présenter l'outil Helix Explorer disponible à l'adresse http://www-lbit.iro.umontreal.ca/mms/he.htm, ses fonctionnalités et son implémentation. Il est organisé comme suit : dans le premier chapitre nous introduisons les différents types de 'molécules de la vie' ou biomolécules. Ces dernières regroupent essentiellement les sucres, les acides nucléiques, les lipides et les protéines. Nous nous intéressons néanmoins dans ce chapitre plus particulièrement aux protéines et à leur structure. Le chapitre 2 traite de l'ingénierie des protéines, des avancées et techniques actuellement employées dans la conception de nouvelles protéines. Dans le chapitre 3 nous introduisons les plus importantes bases de données de structures protéiques ainsi que leur utilisation. Dans le chapitre 4 nous présentons notre contribution avec l'outil Helix Explorer dont nous décrivons tout d'abord les fonctionnalités et l'interface utilisateur avant de décrire en détails l'architecture logicielle et l'implémentation effectuée et enfin conclure en en

énumérant certaines limitations et en proposant des évolutions possibles. Le chapitre 4 est suivi par une brève conclusion.

Chapitre 1. Les protéines et leur structure

1.1 Les biomolécules

Les biomolécules sont des molécules organiques qui sont essentielles à la survie d'un organisme vivant. Elles sont constituées essentiellement des quatre éléments chimiques que sont le carbone, l'hydrogène, l'oxygène et l'azote.

Les biomolécules contiennent les quatre catégories suivantes: les sucres, les lipides, les protéines et les acides nucléiques. Cette classification est principalement basée sur la fonction de ces différents types de molécules dans le métabolisme définie par leur composition et propriétés physico-chimiques ainsi que leur localisation (les acides nucléiques par exemple sont majoritairement présents à l'intérieur de la cellule sous forme d'ADN dans le noyau de la cellule). Dans ce qui suit nous introduisons brièvement les protéines qui sont les biomolécules au centre du travail décrit dans les chapitres suivants.

1.1.1 Introduction aux protéines

Il est naturel de penser que toutes les molécules biologiques sont indispensables au fonctionnement d'un organisme vivant. La réalité est qu'elles le sont. Mais s'il était demandé de les classer selon leurs degrés d'importance, les protéines, que nous allons décrire en plus de détails dans la section 1.2 de ce chapitre, seraient certainement en haut de la liste. En effet, elles sont nécessaires à la structure, la fonction et la régulation des cellules, des tissus et des organes. Chaque protéine a sa propre fonction qu'elle soit une hormone, un anticorps ou une enzyme.

Sous sa forme chimique la plus simple[3], une protéine se présente sous forme d'une ou de plusieurs molécules (ou unités moléculaires), chacune n'étant rien d'autre qu'une séquence (ou chaîne) linéaire d'acides aminés. Dans un contexte plus large, le terme acide aminé

---

[3] Certaines protéines contiennent en effet des groupements ou 'sous-molécules' non-standards tel que le groupement hème dans l'hémoglobine par exemple.

désigne habituellement n'importe quelle molécule qui contient à la fois un groupement amine (NH_2) et un groupement acide carboxylique (*COOH*) attachés au même atome de carbone. Cependant, dans la suite de ce document, comme il est coutume chez les biochimistes, ce terme désignera une molécule parmi une collection bien étudiée de vingt molécules habituellement trouvées dans les organismes vivants. Tous les acides aminés, à l'exception de la proline et de la glycine, partagent la formule chimique NH_2-*CHR*-*COOH* où *R* dénote la chaîne latérale qui est spécifique au type d'acide aminé. Dans le cas de la *proline*, le groupement amine forme un cycle avec la chaîne latérale et dans celui de la *glycine*, la chaîne latérale *R* est réduite à un simple atome d'hydrogène (voir Figure 1.2). Chaque acide aminé peut être désigné soit par son nom complet (*proline* par exemple), une abréviation en trois caractères (en général les trois premiers caractères du nom complet) ou par une lettre unique de l'alphabet (voir Figure 1.2). Cette dernière représentation est naturellement adaptée à la transmission et au stockage informatique des données sur les séquences protéiques.

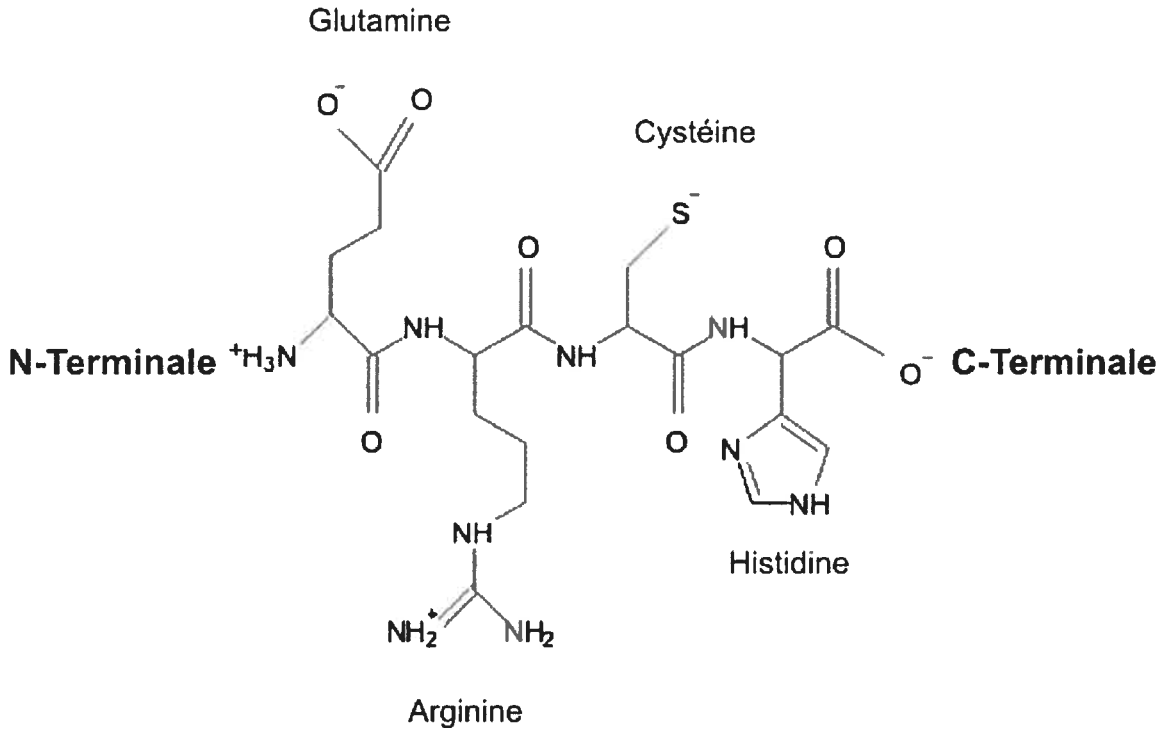

Fig. 1.1 Exemple de peptide. Représentation chimique de la chaine peptidique Glutamine-Arginine-Cystéine-Histidine.

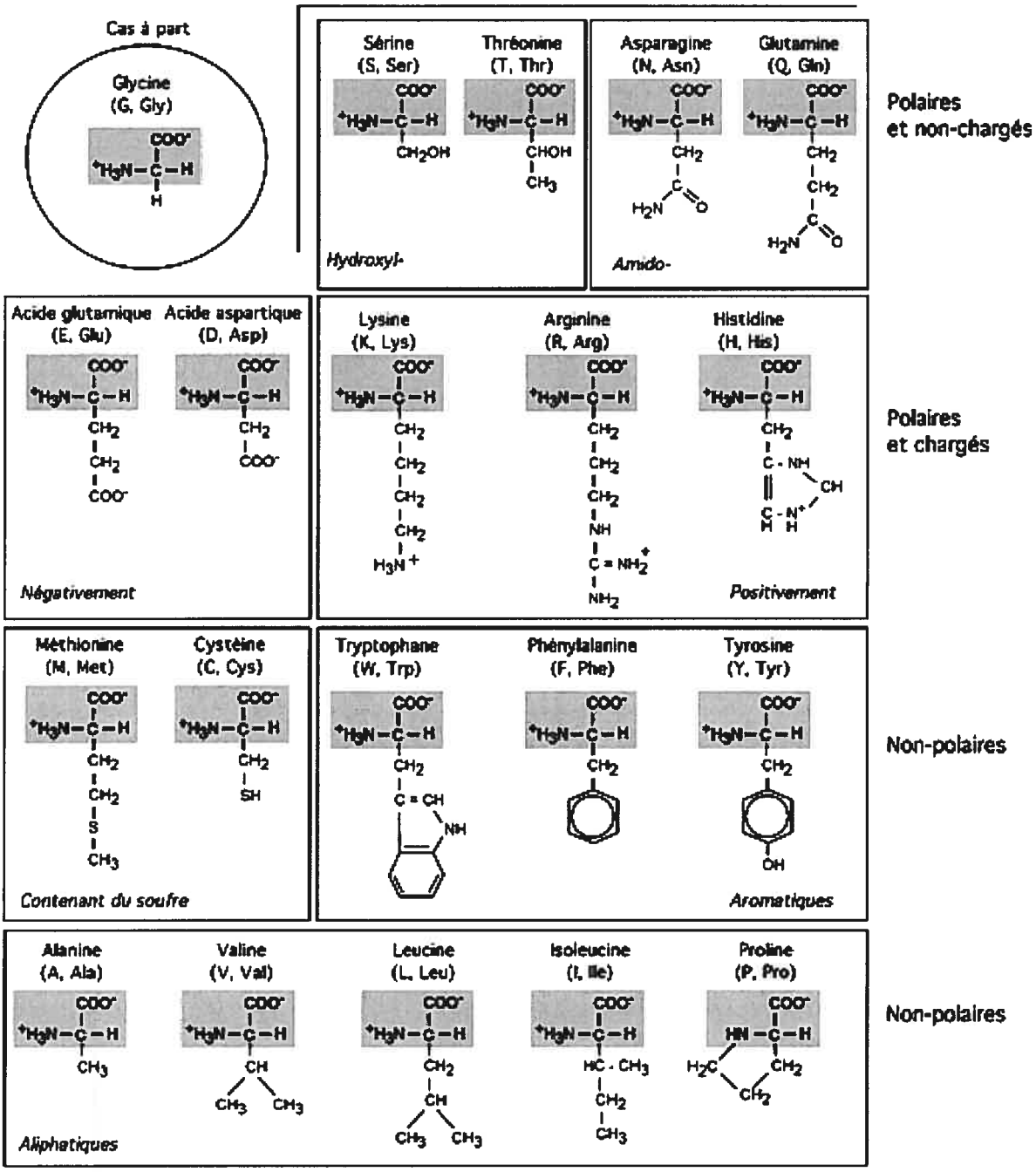

Fig. 1.2 Les 20 acides aminés naturels. Pour chaque acide aminé sont indiquées, la formule chimique de sa chaine latérale, son nom complet et ses identifiants à trois et à un seul caractère. Figure reproduite de [24].

La liaison chimique qui est établie entre deux acides aminés successifs dans une chaîne polypeptidique est appelée 'liaison peptidique'. Dans cette liaison, le groupement acide carboxylique d'un acide aminé est 'fusionné' avec le groupement amine de l'acide aminé qui le suit dans la chaîne polypeptidique. Ainsi, la chaîne polypeptidique, peu importe sa longueur, présente une extrémité amine dite N-terminale et une autre extrémité au côté opposé de la

chaîne dite C-terminale. Il en résulte qu'une chaîne polypeptidique peut être 'lue' selon une des deux directions N-terminale->C-terminale (qu'on notera N-C) ou C-terminale->N-terminale (C-N). Il est néanmoins d'usage de donner une séquence selon l'orientation N-C. Une représentation chimique de la séquence Glutamine-Arginine-Cystéine-Histidine par exemple est donnée dans la Figure 1.1.

1.2 Les protéines et leurs structures

1.2.1 Les protéines en tant que polymères d'acides aminés

Comme illustrée dans la section précédente, la formule chimique d'un acide aminé est très simple, cependant il existe une grande diversité dans les protéines et leurs fonctions. Celle-ci provient (1) du nombre relativement important de types d'acides aminés trouvés dans la nature qui est, comme mentionné précédemment, de vingt (comparativement à seulement cinq bases nucléiques trouvées dans l'ADN/ARN) et surtout (2) de la diversité des propriétés physiques et chimiques de ces derniers. Nous allons décrire dans la suite de cette section certaines propriétés des acides aminés et leurs implications sur la fonction et/ou comportement des protéines qui les renferment.

La Figure 1.2 liste les vingt acides aminés que l'on retrouve habituellement dans la nature. Selon leur composition, c'est-à-dire le type de leur chaîne latérale, les acides aminés vont avoir des propriétés physiques/chimiques différentes. Certains acides aminés seront plus hydrophiles (ou polaires) que d'autres et seront donc portés à se trouver en contact avec les molécules d'eau qui se trouvent en général dans le voisinage de la protéine, alors que

certains vont, au contraire, être hydrophobes et avoir tendance à fuir les molécules d'eau présentes dans le milieu et de s'associer entre elles et avec d'autres molécules qui sont aussi de nature hydrophobe (telles que les lipides). La nature hydrophile/hydrophobe de l'acide aminé se retrouve par ailleurs à l'échelle de la protéine. On parle ainsi de protéine hydrophile lorsque la protéine regroupe à sa surface majoritairement des acides aminés hydrophiles qui lui procurent une affinité pour les milieux aqueux alors qu'on parle de protéine hydrophobe si la surface est essentiellement hydrophobe. Certaines protéines, dont la surface contient à la fois des régions hydrophobes et des régions hydrophiles, sont dites amphiphiles. C'est le cas notamment des protéines transmembranaires[4].

Une autre propriété, importante à mentionner ici, est l'aspect acido-basique d'un acide aminé ou ce que l'on peut définir par la faculté d'un acide aminé à transférer, par l'intermédiaire de sa chaîne latérale, à d'autres molécules environnantes un proton (c'est-à-dire un hydrogène dépourvu d'un électron) ou au contraire celle de s'approprier, toujours par l'intermédiaire de sa chaîne latérale, un hydrogène des molécules de son voisinage. On comprend aisément pourquoi cette propriété est fondamentale pour prédire la nature des réactions chimiques dans lesquelles une protéine est susceptible d'intervenir. Certains acides aminés agissent comme donneurs de protons dans une solution au pH physiologique et ils sont dits acides, d'autres agissent comme receveurs de protons et sont dits basiques. De plus, parmi les acides aminés polaires/hydrophiles, il en existe certains qui ne portent jamais de charges électrostatiques et qui sont dits non-chargés et d'autres, qui ne sont rien d'autres que les acides aminés définis précédemment sous les catégories acides et basiques, dont les chaînes latérales se trouvent sous leurs formes chargées sous certaines conditions de pH.

Plusieurs autres critères de classification des acides aminés existent. Citons-en deux ici: (1) certains acides aminés sont dits aliphatiques (par opposition à non aliphatiques) et sont définis comme étant ceux qui ne contiennent pas d'atomes N, O ou S dans leurs chaînes

---

[4] Dans ce type de protéines, une région hydrophobe est généralement située à l'intérieur de la membrane cellulaire alors qu'une autre région hydrophile s'étend vers l'extérieur ou l'intérieur de la cellule.

latérales et (2) certains acides aminés sont dits aromatiques (par opposition à non aromatiques) et sont définis de leur part comme ceux contenant un anneau de benzène dans leurs chaînes latérales. Ces deux propriétés des acides aminés, comme les autres propriétés non mentionnées ici et qui servent d'ordinaire à définir des 'classes' d'acides aminés, ont en commun l'habilité d'accorder, sous certaines conditions[5], aux protéines qui les renferment leur réactivité et/ou comportements vis-à-vis des autres molécules dans leurs voisinages.

1.2.2 Enzymes et interactions moléculaires

Une grande proportion des protéines est constituée de catalyseurs. Ces protéines sont connues sous le nom générique d'enzymes. Comme tout catalyseur, l'apport principal d'une enzyme à une réaction biochimique donnée est celui d'abaisser la barrière d'énergie d'activation de cette dernière, ce qui permet d'accélérer la réaction qui autrement se serait produite à une vitesse suffisamment faible au point de la rendre 'inutile'. Il en découle que les enzymes, et par extension les protéines, sont les vrais moteurs de la vie.

À l'intérieur ou à la surface de toute enzyme se trouve un espace bien délimité où se déroule habituellement la réaction catalytique. Cet espace est désigné par le terme 'site catalytique' (ou site actif) et regroupe généralement un petit nombre d'acides aminés participant à la (aux) réaction(s) chimique(s) à l'origine de l'activité catalytique. Il est fréquent de trouver dans le site catalytique, en plus de ces acides aminés, d'autres molécules, dites cofacteurs, qui participent à la réaction enzymatique et qui sont en général spécifiques à cette dernière. D'autre part, il est important pour la suite de noter ici que les enzymes, ou plus généralement les protéines, interagissent avec les autres molécules par l'intermédiaire des résidus qui sont exposés à la surface moléculaire. Voir section 1.2.5.

---

[5] Par exemple d'accessibilité de l'acide aminé depuis le voisinage extérieur de la protéine.

1.2.3 Structures des protéines

Le fait qu'une protéine ne soit constituée que d'une simple chaîne séquentielle d'acides aminés, ne signifie aucunement qu'elle va adopter une structure linéaire. En effet, le repliement de la chaîne sur elle-même, lui confère souvent une apparence globulaire. Il est admis que la structure finale adoptée par la protéine dépend de la séquence en acides aminés qui la constituent, ou de sa «structure primaire», et éventuellement d'éléments externes intervenants lors du processus de repliement (l'intervention de protéines chaperonnes par exemple). La structure formée après le processus de repliement est dite «structure tertiaire» de la protéine. Si la protéine est un complexe où plusieurs 'structures tertiaires' se rejoignent pour former la structure finale, cette structure est dite «quaternaire». Enfin, pour compléter la description précédente des différents niveaux de structures distinguées au sein des protéines, il reste à introduire la «structure secondaire», ce que nous faisons dans la section qui suit.

1.2.4 Structures secondaires des protéines

Plusieurs structures *locales* peuvent se former à l'intérieur d'une chaîne protéique. De ces structures, trois ont été bien caractérisées : les hélices, les feuillets β et les tours.

1.2.4.1 Les hélices

Les hélices (Figure 1.3) peuvent se présenter sous plusieurs formes différentes : hélices α, hélices 3/10 et hélices pi. Ces trois types de structures ont tous la même forme hélicoïdale soutenue par des ponts hydrogènes le long de l'hélice (même si les ponts établis ne sont pas les mêmes pour les trois types d'hélices). Elles diffèrent uniquement dans les valeurs de certains paramètres géométriques. Le nombre d'acides aminés par tour d'hélice peut ainsi varier entre 3 et 5,6 selon le type de l'hélice. Il est de 3.6 dans les hélices α qui sont de loin les hélices les plus communes.

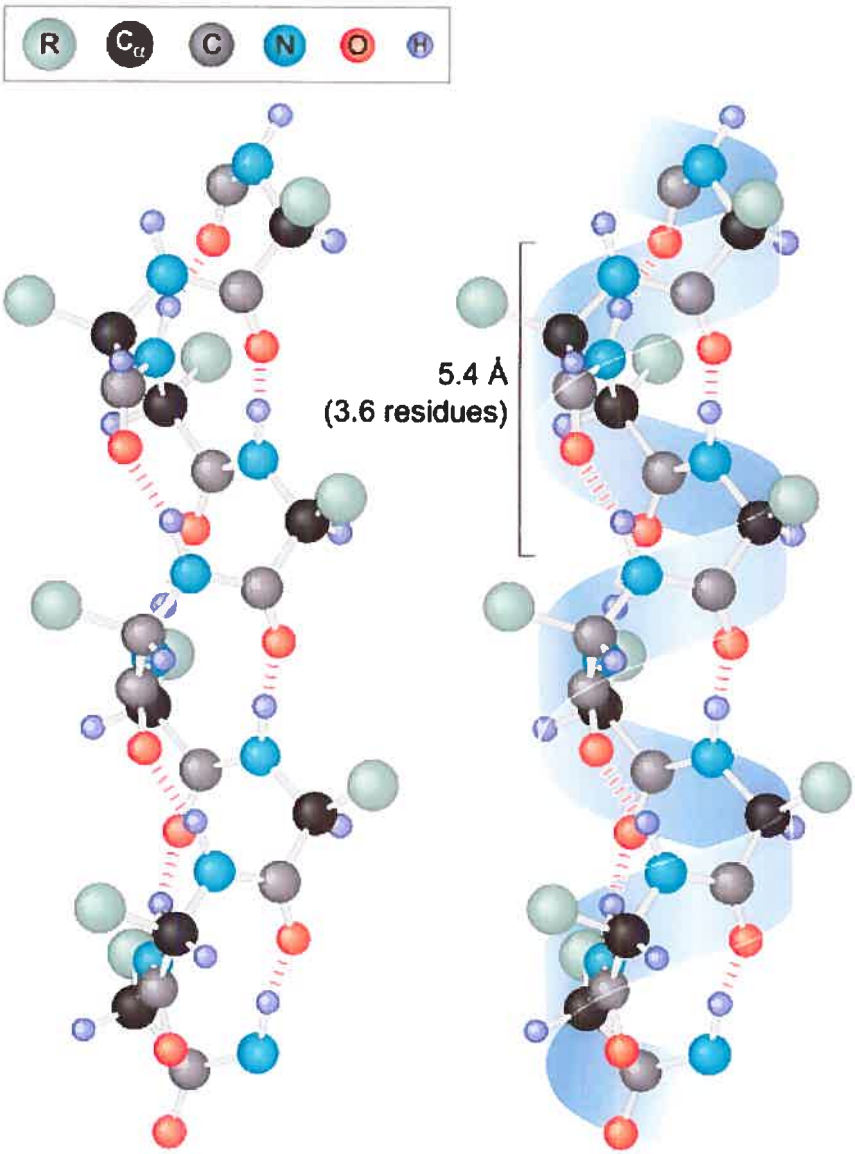

Fig. 1.3 Hélices α. Schéma représentant une hélice α. Les atomes de carbones C sont en noir, l'Azote N en bleu, l'oxygène O en rouge et l'hydrogène H en gris. Les chaînes latérales (R), qui sont toujours dirigées vers l'extérieur de l'hélice, sont représentées ici par de simples sphères. Les ponts hydrogènes sont représentés en pointillés. [25]

1.2.4.2 Les feuillets β

Comme dans le cas des hélices, il existe plus qu'un seul type de feuillets β. Ils sont regroupés généralement sous une des deux catégories suivantes: les feuillets parallèles et les feuillets antiparallèles (voir Figure 1.4). Comme illustré dans la Figure 1.4, deux brins du même feuillet parallèle sont dirigés dans le même sens (N-C, N-C) alors qu'ils sont orientés dans des directions opposées dans le cas d'un feuillet antiparallèle (N-C, C-N).

De la même façon que pour les hélices, ce sont des ponts hydrogènes, établis dans le cas des feuillets β entre deux brins adjacents, qui soutiennent la structure du feuillet.

1.2.4.3 Les tours

Le troisième type de structures secondaires regroupe les tours qui sont des structures irrégulières dans le sens où, contrairement aux hélices et feuillets β, elles n'ont pas de géométrie bien définie. En général, un tour relie deux structures secondaires qui se suivent dans la chaîne peptidique (deux hélices par exemple) ou relient entre eux deux brins du même feuillet β (les deux brins pouvant cependant ne pas être adjacents dans le feuillet). Les tours ont été étudiés en détails. Le nombre d'acides aminés (en général de 3 à 5) qui les forment et le type de ces derniers ont été bien décrits [26,27,28,30]. On sait par exemple que certains acides aminés sont souvent rencontrés en début, au milieu ou à la fin des tours et que selon son type, un acide aminé peut influencer la flexibilité d'un tour et par conséquent celle de la structure de la protéine tout entière.

1.2.5 Surfaces moléculaires

Dans ce document, nous avons utilisé précédemment l'expression 'surface d'une protéine' sans introduire formellement de quoi il s'agit. Intuitivement, la surface d'une protéine ou plus généralement la surface d'une molécule (aussi dite 'surface moléculaire') peut être vue comme la surface géométrique qui sépare la molécule de son extérieur. Cela sous entend que les atomes qui forment la molécule occupent un espace géométrique

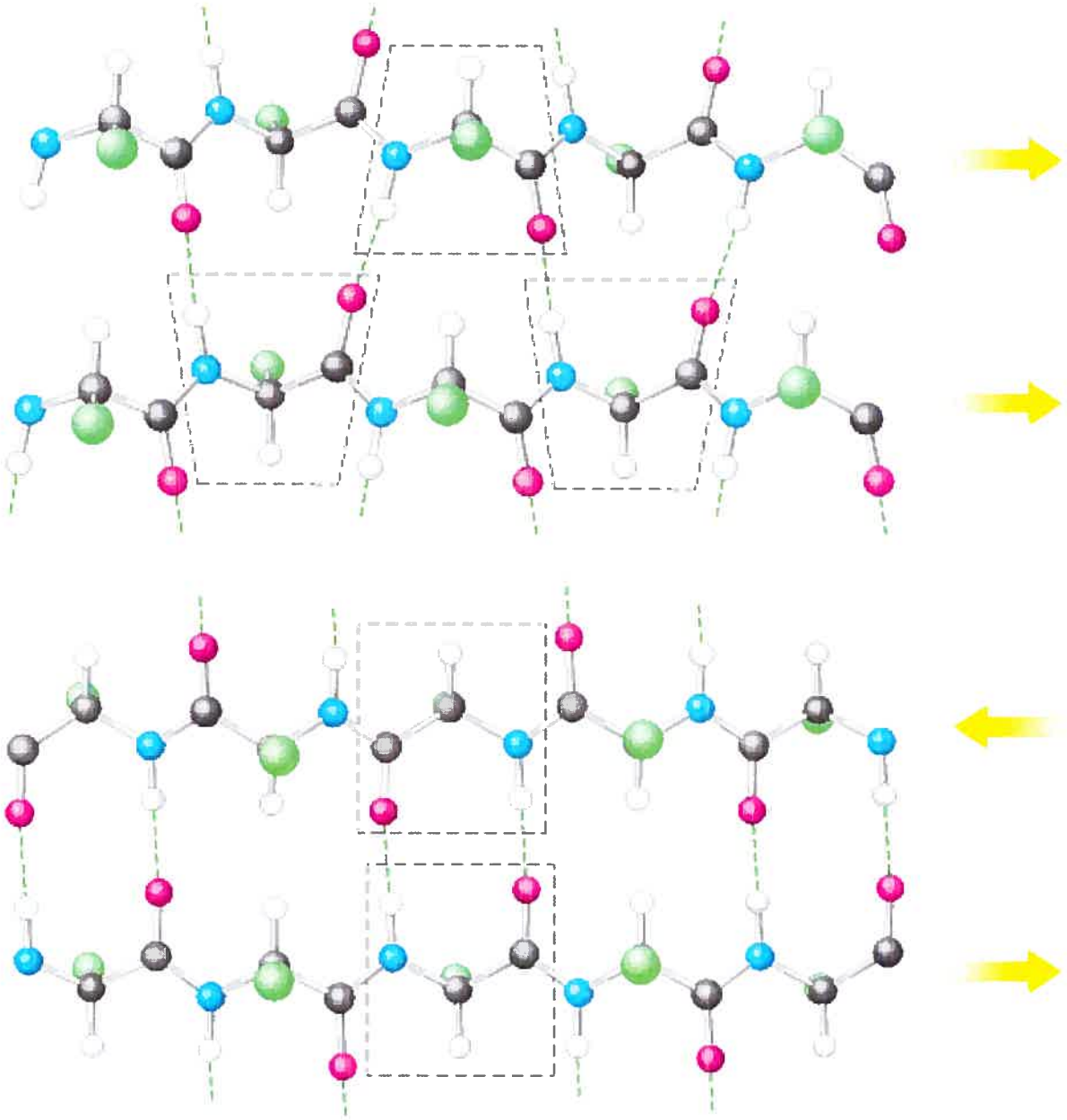

Fig. 1.4 Feuillets β. Schéma représentant un feuillet β parallèle (haut) et antiparallèle (bas) formé de deux brins. Les atomes de carbones C sont en noir, l'Azote N en bleu, l'oxygène O en rouge et l'hydrogène H en blanc. Les flèches indiquent la direction N-C de chaque brin. Les ponts hydrogènes sont représentés en pointillés. Figure reproduite de [30].

délimité. Si c'est le cas, comment cet espace est-il défini ? Comment construit-on à partir de cet espace ce que l'on appelle la surface moléculaire ? Cette section répond à ces questions en introduisant plusieurs définitions de types de surfaces et les techniques utilisées pour les calculer.

1.2.5.1 Surface de van der Waals

L'idée que l'espace occupé par un atome est restreint à un espace sphérique bien défini ou plus généralement à un espace fini est en contradiction avec la théorie de la mécanique quantique qui veut que la structure électronique d'un atome, qui est représentée par une fonction d'onde, ne soit pas bornée dans l'espace. Néanmoins, pour toute paire d'atomes qui sont 'proches' l'un de l'autre sans être pour autant liés (unbonded), on peut mesurer expérimentalement la distance entre eux. Cette distance correspond à la somme de ce qui est définit comme étant les rayons de van der Waals (van der Waals radii) respectifs des deux atomes. Lorsque, par exemple, les deux atomes sont identiques, on peut conclure, étant donnée la symétrie de la configuration formée, que le rayon de van der Waals de chacun des atomes est égal à la moitié de la distance mesurée entre les deux atomes. Il est important de noter ici que le rayon de van der Waals est différent du rayon 'métallique' ou 'covalent' qui correspond au rayon mesuré lorsque les deux atomes ne sont pas seulement en contact mais établissent en plus des interactions covalentes. Le volume défini par une sphère centrée sur un atome et dont le rayon est le rayon de van der Waals de cet atome est appelé volume de van der Waals de l'atome (voir Figure 1.5). La surface extérieure qui délimite l'ensemble des volumes de van der Waals d'une molécule, c'est-à-dire de tous ses atomes, est dite surface de van der Waals.

1.2.5.2 Surface accessible

Si l'on fait rouler une sonde sur la surface de van der Waals d'une molécule, c'est-à-dire sur la surface formée par l'union des surfaces de van der Waals de tous les atomes de

la molécule, le centre de la sphère décrit une autre surface désignée par «surface accessible». Cette surface peut être vue comme étant la surface de van der Waals étendue par un terme qui correspond au rayon de la sonde (voir Figure 1.5). La région définie à l'intérieur de la surface accessible est une région qui théoriquement n'est pas accessible aux molécules de l'eau/solvant (en supposant que la sonde représente une molécule d'eau/solvant) et elle est pour cette raison appelée volume d'exclusion.

1.2.5.3 Surface moléculaire

La partie de la surface de van der Waals qui peut être en contact avec une sonde de la taille de molécules d'eau (ou solvant) est dite 'surface de contact'. D'autre part, lorsque la même sonde est en contact avec plus qu'un atome, la surface définie par la face intérieure de la sonde est appelée 'surface réentrante'. Ce qui est communément connue par 'surface moléculaire' n'est alors rien d'autre que l'union de la surface réentrante et de la surface de contact (voir Figure 1.6). Un des avantages de la surface moléculaire sur la surface de van der Waals ou la surface accessible est qu'elle permet la visualisation de la complémentarité des formes entre deux molécules ou plus. Cela est très utile lorsqu'on veut étudier l'arrimage de deux molécules (ce qui est très fréquent dans le développement de

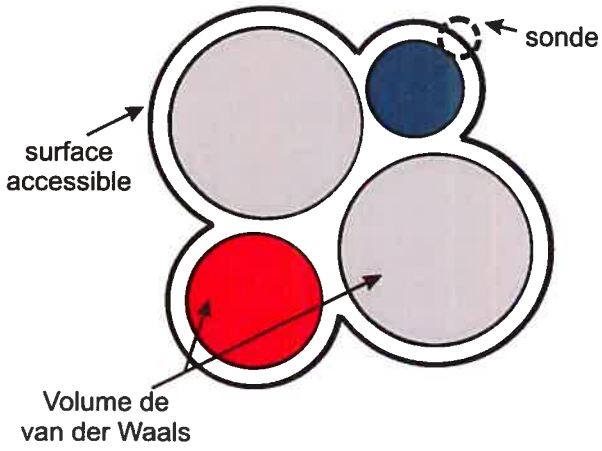

Fig. 1.5 Représentation simplifiée de la surface accessible et du volume de van der Waals. Ce schéma représente 4 atomes de trois types différents, i.e. avec trois rayons de van der Waals différents où chaque type est representé par une couleur différente. Lorsqu'une sonde représentant une molécule du solvant est roulée sur la surface de van der Waals, son centre décrit la surface accessible.

médicaments ou plus généralement dans le domaine de la recherche pharmaceutique).

À noter que, par abus de langage, certains termes introduits précédemment peuvent avoir de multiples définitions. Le terme 'volume d'exclusion' par exemple, désigne parfois le volume inclus à l'intérieur de la surface moléculaire.

1.2.5.4 Calcul de la surface moléculaire

Plusieurs algorithmes ont été conçus dans le but de calculer la surface moléculaire. L'approche initiale adoptée par Greer et Bush en 1978 [31] consistait à faire 'pleuvoir' selon une direction préalablement définie des sondes représentant les molécules du solvant (des molécules d'eau en général). L'intersection des sondes avec la molécule (i.e. l'ensemble des points définis par le contact de la sonde avec la surface van der Waals de la molécule) appartient à la surface moléculaire. En faisant usage d'une densité plus grande de sondes par unité de surface, il est possible d'affiner la surface moléculaire recherchée. Cette méthode fonctionne bien pour des surfaces 'aplaties' mais elle ne permet pas de décrire les régions irrégulières de la surface moléculaire qui comprennent des surplombs (*overhangs*).

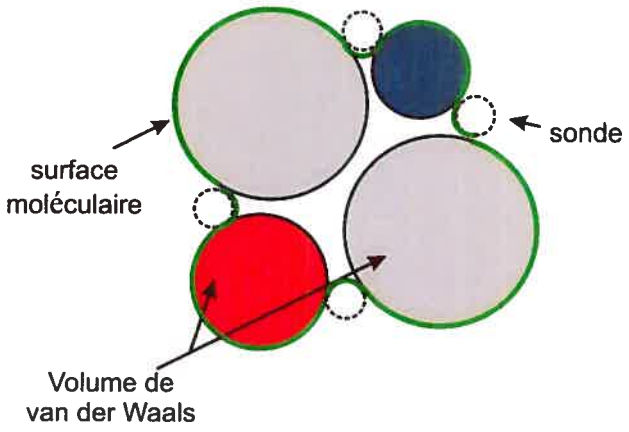

Fig. 1.6 Surface moléculaire. Schéma représentant une molécule avec 4 atomes de trois types différents, i.e. avec trois rayons de van der Waals différents. Lorsqu'une sonde représentant une molécule de solvant est roulée sur la surface de van der Waals, les points de contact de la sonde et la surface réentrante de la sonde dans certaines régions définissent ensemble la surface moléculaire.

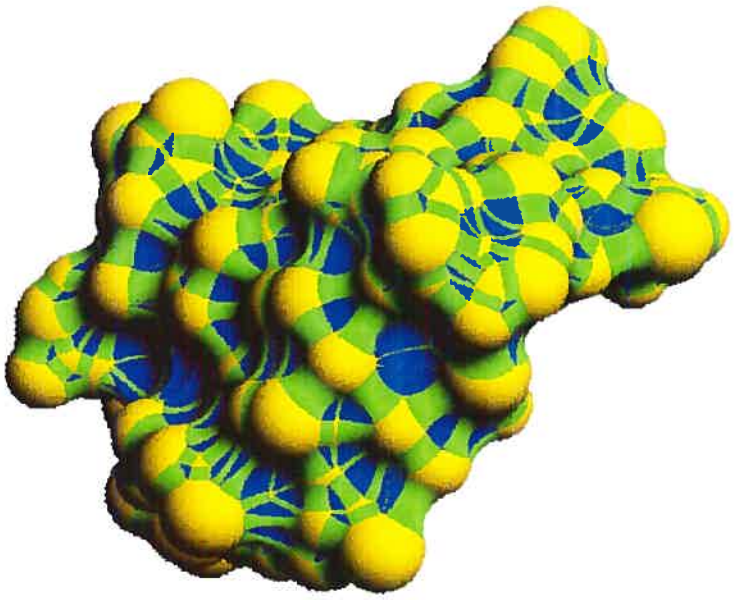

Fig. 1.7 Calcul de la surface moléculaire d'une protéine. La surface moléculaire est composée de deux sous-surfaces distinctes et complémentaires : (1) surface de contact (en jaune) définie par les points de contact de la sonde avec la surface de van der Waals lorsque ce contact se fait en un seul point et (2) surface réentrante, définie par la surface réentrante de la sonde lorsque le contact se fait en deux points (en vert) ou en trois point ou plus (en bleu). Figure reproduite de [32].

La méthode la plus communément utilisée aujourd'hui pour trouver la surface moléculaire consiste, sommairement, à placer une sonde tangente à la surface de van der Waals de chaque atome de la molécule et de la faire rouler sur cette dernière dans une direction donnée tout en s'assurant qu'elle n'entre pas en collision avec les surfaces van der Waals des atomes avoisinants. Lorsque la sonde est en contact avec un seul atome, le point de contact fait partie de la surface moléculaire. Au contact de deux atomes, la sonde décrit des régions en forme de selle de cheval (*saddle-points*) de tores qui sont incluses dans la surface moléculaire, alors qu'au contact de trois atomes ou plus la sonde définit une région

concave réentrante de la surface. La surface moléculaire est construite en combinant ces trois types de surfaces (voir Figure 1.7). À noter que plusieurs variantes de cette méthode existent. Elles appartiennent toutes néanmoins à une des deux catégories suivantes : (1) algorithmes numériques qui échantillonnent la surface en un ensemble discret de points ou de courbes planes (tels que des triangles) (2) algorithmes analytiques qui définissent la surface moléculaire comme une collection de sous-surfaces qui sont elles-mêmes ou bien incluses dans une sphère (dont le rayon et centre est connu) et délimitées par des arcs (aussi décrits analytiquement) ou alors sont des régions de tores incluses dans la surface réentrante.

Chapitre 2. Ingénierie des protéines

2.1 Introduction

En plus d'être au cœur des systèmes vivants (voir chapitre 1), les protéines, et en particulier les enzymes parmi elles, sont largement utilisées dans plusieurs industries telles que l'agriculture, l'alimentation, l'énergie ou l'industrie pharmaceutique. Certes, une grande proportion des enzymes utilisées par ces industries sont des protéines isolées et extraites de la nature ou sont des copies de ces dernières, qui ont été synthétisées, pour des raisons quelconques (souvent pour obtenir des quantités industrielles), dans des laboratoires. Cependant, depuis plusieurs années, de plus en plus de protéines qui sont utilisées dans l'industrie n'ont pas d'équivalents dans la nature et peuvent en conséquence être qualifiées d'artificielles. L'intérêt général qui existe aujourd'hui pour ces protéines n'est qu'un reflet de la multitude d'avantages qu'elles ont souvent sur les protéines naturelles.

Dans bien des cas, la fonction qui est recherchée dans une protéine n'a pas été préalablement observée dans la nature. L'unique voie possible pour réaliser la fonction en question est par conséquent de concevoir de nouvelles protéines (ou d'autres molécules non protéiques) pour l'accomplir. À première vue, concevoir de telles protéines, qui réaliseraient une nouvelle fonction donnée, représente un double défi. En effet, il faut non seulement pouvoir décider de la structure qui réalisera la fonction souhaitée mais il faut aussi pouvoir créer cette structure. De plus, si l'on garde à l'esprit que, rien que de comprendre en détails les mécanismes de certaines réactions enzymatiques, peut parfois nécessiter des années (voir des décennies) d'étude, de tels défis peuvent sembler insurmontables. Cependant, de nouvelles techniques, dont nous discutons plus loin dans ce chapitre, ont été développées. En particulier, une approche adoptée prend avantage de processus évolutifs aléatoires pour créer de nouvelles fonctions (voir la section 2.2). Ajoutons à cela le fait que le terme 'fonction' dans le présent contexte n'est pas synonyme de 'fonction enzymatique', mais peut par exemple, designer simplement la faculté de la protéine conçue de lier d'autres molécules (i.e. fonction de reconnaissance moléculaire)

sans pour autant agir comme catalyseur d'aucune réaction biochimique. La prise en compte de ces deux derniers points, rend théoriquement plus 'accessible' l'idée de créer des protéines artificielles avec de nouvelles fonctions.

D'autre part, dans beaucoup d'autres cas, la protéine artificielle n'est pas conçue pour réaliser une fonction non existante (ou du moins non encore découverte) dans la nature, mais dans le but de réaliser exactement la même fonction qu'une protéine connue. L'intérêt à utiliser une protéine artificielle à la place d'une protéine naturelle dans ces cas-ci est que souvent, la première peut offrir des avantages importants. Des exemples de ces avantages comprennent, entre autres, une plus grande stabilité thermique de la protéine artificielle par rapport à son homologue naturelle [33], la réalisation d'une meilleure efficacité de la réaction (que cette dernière soit enzymatique ou non) [34] ou, moins spécifiquement, une meilleure adaptabilité aux conditions industrielles [35]. Vues sous un autre angle, les nouvelles protéines peuvent être perçues comme des protéines naturelles améliorées. En pratique, cela signifie que la conception de telles protéines peut se faire en partant de leurs équivalents naturels et en y introduisant de façon progressive des modifications ponctuelles jusqu'à l'obtention des structures et fonctions recherchées. Cette conception peut être soit 'rationnelle' et utiliser plusieurs techniques que nous décrivons dans la section 2.2 ou alors peut dépendre de processus aléatoires (voir section 2.3). Avant de décrire ces deux approches, notons que l'ensemble de techniques et méthodes dont le but ultime est de développer de nouvelles protéines sont aujourd'hui au centre d'une discipline relativement jeune connue sous le nom d'ingénierie des protéines.

2.2 Conception rationnelle de protéines

Le plus grand obstacle à la création de nouvelles protéines est certainement la difficulté de prédire la structure d'une protéine à partir de sa séquence. Les protéines qui sont obtenues par 'conception rationnelle' (voir Figure 2.1), comme mentionné plus haut,

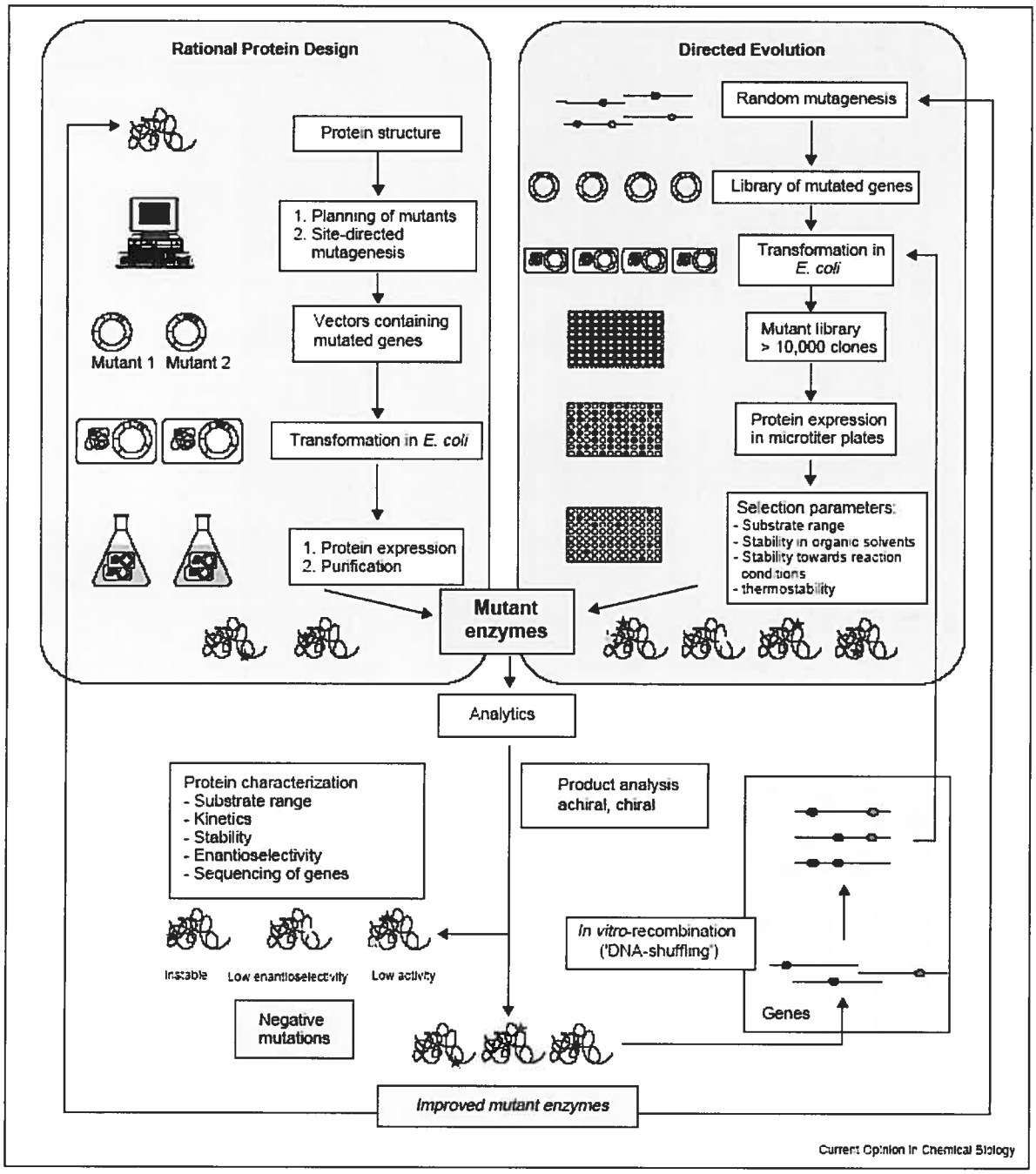

Fig. 2.1.Les deux principales approches dans la conception de nouvelles protéines. À droite, la conception par évolution dirigée. À gauche, la conception rationnelle. Les deux méthodes ne sont pas mutuellement exclusives. Figure reproduite de [36].

sont souvent conçues par introduction de modifications ponctuelles[6] dans des protéines qui ont été isolées dans la nature et dont la fonction et la structure ont été préalablement bien étudiées.

La plus couramment utilisée des techniques est la mutagenèse dirigée qui consiste à remplacer un acide aminé (ou des fois plus qu'un seul) de la protéine par un autre acide aminé. La procédure est dite rationnelle parce que ce changement n'est pas aléatoire et qu'il est justifié par deux suppositions : (1) que la structure de la protéine ne va être modifiée que légèrement dans la protéine mutante par rapport à celle de type sauvage (i.e. celle trouvée dans la nature), ce qui signifie qu'une part importante de la fonction telle que la reconnaissance moléculaire est généralement conservée et (2) que l'effet de la mutation introduite est prévisible ou du moins motivée par une hypothèse que l'on a faite au préalable sur la structure/fonction de la protéine synthétisée (ou mutante). Lorsque la protéine mutée est une enzyme, l'hypothèse, qui est ultérieurement vérifiée, est dans bien des cas que la nouvelle structure, tout en conservant la même fonction, modifie au moins un des paramètres de la réaction enzymatique. Ce paramètre peut être relatif à la cinétique de la réaction, comme il peut aussi être par exemple la température optimale, ou le pH optimal permettant une meilleur sélectivité de l'enzyme ou une meilleure efficacité de la réaction enzymatique.

L'analyse qui précède suggère une méthode très générale qui a été adoptée dans la conception 'rationnelle' de protéines. Cette méthode consiste en l'utilisation d'échafaudages. Un échafaudage est une construction moléculaire qui offre un 'cadre' dont la structure a été bien caractérisée et qui est relativement stable pour ne pas subir des changements importants de structure suite aux modifications qui y sont introduites. Un échafaudage peut être une simple protéine qui autorise des modifications selon les besoins

---

[6] Cela est réalisé, en pratique, en «reconstruisant une protéine identique partout à la protéine naturelle sauf à certaines positions de la chaîne protéique»

du concepteur comme il peut également être complètement ou partiellement non protéique. Dans la Figure 2.2 sont schématisés quelques exemples d'échafaudages utilisés.

Un des échafaudages présentés dans la Figure 2.2, (F), est formé de quatre hélices qui sont positionnées parallèlement l'une à l'autre. La structure alors formée est appelée 'tonneau à quatre hélices'. Un échafaudage similaire à cet échafaudage est souvent construit par une technique dite 'Template-Assembled Synthetic Protein' (TASP) dans laquelle quatre

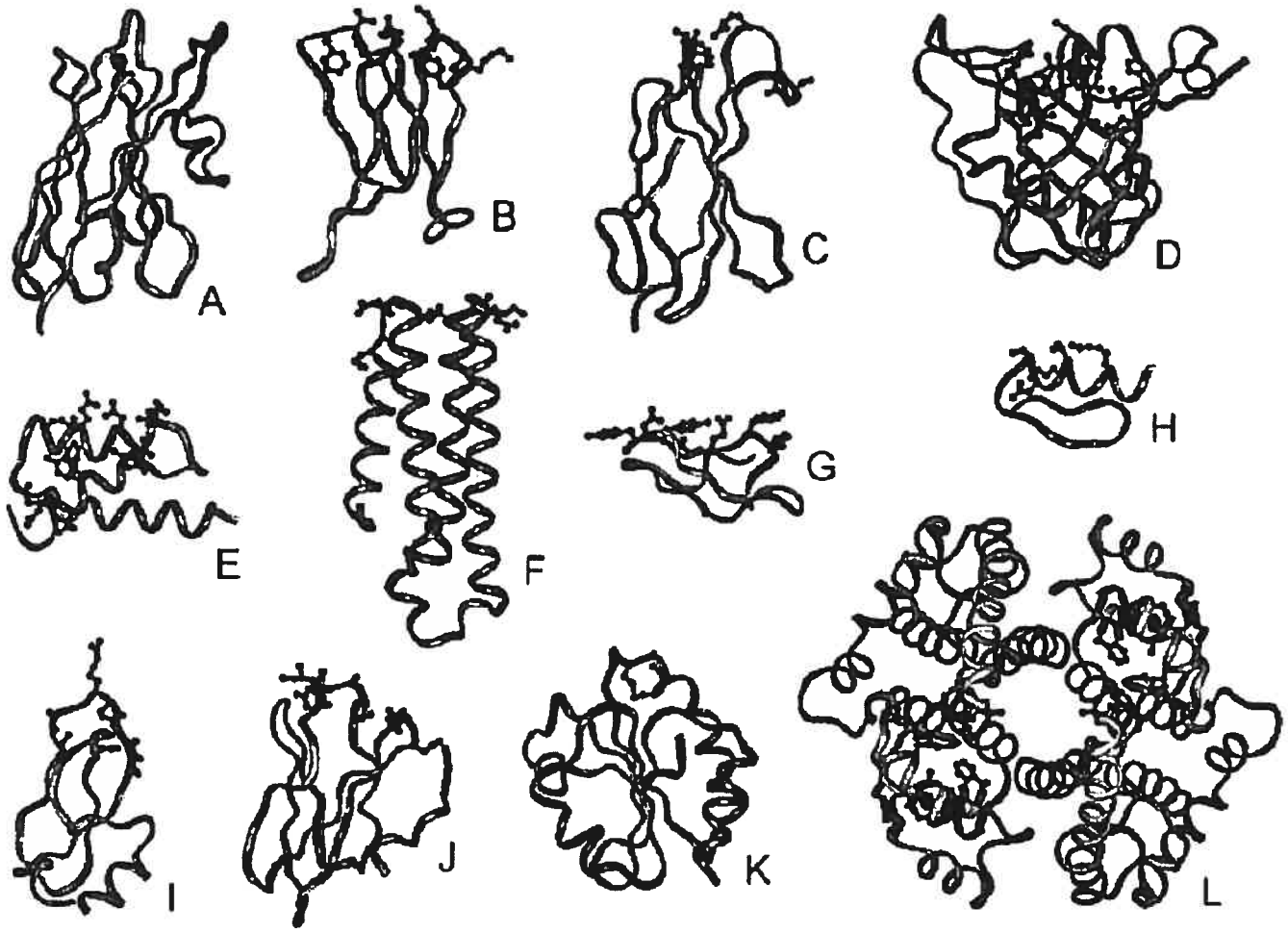

Fig. 2.2 Différents types d'échafaudages. Ceux-ci sont aussi utilisés, dans la conception par évolution dirigée, pour la construction de librairies pour la sélection de variantes possédant une fonction de reconnaissance moléculaire recherchée. (F) est un exemple d'un tonneau à 4 hélices (pdb : 256B). Dans cet exemple le tonneau est formé d'une seule chaîne peptidique. Les autres structures sont extraites des pdb suivants : (A) : 1MEL, (B) : 1MCP, (C) : 1FNA, (D) : 1BBP, (E) : 1BDD, (G) :1CBH, (H) :1MEY, (I) : 1AAP, (J) : 1HOE, (K) : 2TRX, (L) : 1GUH. Figure reproduite de [37].

hélices sont placées sur un support (ou modèle) dont la nature peut varier. La Figure 2.3 illustre certains de ces supports. Dans ce type de construction, la nature du support utilisé est importante. Elle peut en effet définir plusieurs caractéristiques importantes de l'échafaudage, telle que sa structure, sa rigidité, sa stabilité thermique, sa solubilité etc.

La création ou la sélection d'un échafaudage, qui, en fin de compte, peut être perçu comme le squelette moléculaire de la protéine recherchée, fait souvent appel à des techniques de modélisation informatique de molécules. Notons en particulier ici le travaux de David Baker et de Brian Kuhlman et de leurs équipes respectives à qui nous devons en partie l'outil de conception de structure RosettaDesign [38]. Le Laboratoire du Dr. Baker a par exemple reussi à concevoir, en utilisant ces techniques, une protéine formée de 93 acides aminés dont la déviation de la structure théoriquement prédite ne dépasse pas 1,2Angströms. Pour cela, ils ont réalisées plusieurs itérations entre deux phases, une de conception de séquences et une autre de prédiction de structures [39]. En outre, il existe plusieurs publications qui font une revue de modélisations informatiques [40]. Nous invitons le lecteur à s'y référer pour plus de détails. Nous noterons ici uniquement que, malgré les progrès accomplis pendant les dernières années et les succès réalisés dans ce domaine (citons en particulier le travail pionnier de Jane Richardson dans la conception 'de novo' de protéines [41]), il n'est pas encore possible aujourd'hui de construire une large protéine ou une large molécule 'de novo', qui possède la structure et les propriétés recherchées. Même les exemples les plus cités d'échafaudages formés de tonneau à 4 hélices ont pu initialement être construits, non pas grâce à des modélisations informatiques, mais en se servant de certaines séquences peptidiques connues expérimentalement pour former des hélices sous certaines conditions. À noter néanmoins que cette dernière approche a elle aussi ses limites : une séquence peptidique qui forme une structure donnée dans le contexte d'une molécule particulière, peut prendre en effet une structure très différente lorsqu'insérée à l'intérieur d'une autre protéine ou molécule. Un exemple apparent de cela est celui de la séquence Ala-Trp-Thr-Val-Glu-Lys-Ala-Phe-Lys-Thr-Phe rapportée par Minor et Kim [42]. Cette séquence de 11 acides aminés apparaît en effet, à la fois à l'intérieur d'une hélice dans une structure protéique et à l'intérieur d'un feuillet β

dans une autre structure. En conclusion de ce qui précède, il apparaît prudent d'affirmer, sans se montrer particulièrement pessimiste, que le seul moyen fiable de s'assurer qu'une structure est bien celle que l'on désire obtenir de façon rationnelle est de le vérifier expérimentalement en résolvant la structure entière de la molécule.

2.3 Conception de protéines par évolution dirigée

Le terme 'évolution' dans la désignation de cette méthode (*directed evolution*) provient du fait que cette technique s'inspire des mécanises d'évolution des protéines (ou pour être plus précis, de l'information génétique). La technique, telle qu'illustrée dans la Figure 2.1, procède en introduisant des mutations aléatoires dans la séquence ADN[7] à l'origine de la protéine. La synthèse des protéines depuis ces séquences, qui est réalisée à travers un processus similaire à celui de la nature, résulte en une librairie de mutants. Ces mutants sont testés ensuite pour les propriétés désirées. Si l'on cherche par exemple une enzyme mutante qui offre la meilleure stabilité à un intervalle de températures donnée, il suffit de mesurer, individuellement mais souvent en parallèle, l'activité de chaque échantillon de la librairie sous les conditions de température recherchées.

Pour conclure ce chapitre, notons que les techniques de conception présentées précédemment sont essentiellement expérimentales. Elles pourraient néanmoins, selon l'approche que nous présentons dans le chapitre 4, bénéficier de nouvelles techniques computationnelles. En particulier, de l'exploration des dizaines de milliers de structures de protéines qui ont été résolues dans les vingt dernières années. Mais avant de discuter de cette approche, nous présentons, dans le chapitre suivant, Protein Data Bank (PDB), la plus grande des bases de données de structures macromoléculaires.

---

[7] Une autre approche qui a été considérée consiste à effectuer la recombinaison entre fragments aléatoires de gènes obtenus de la dégradation par une DNAase du gène codant pour plusieurs variantes de la protéine naturelle [71].

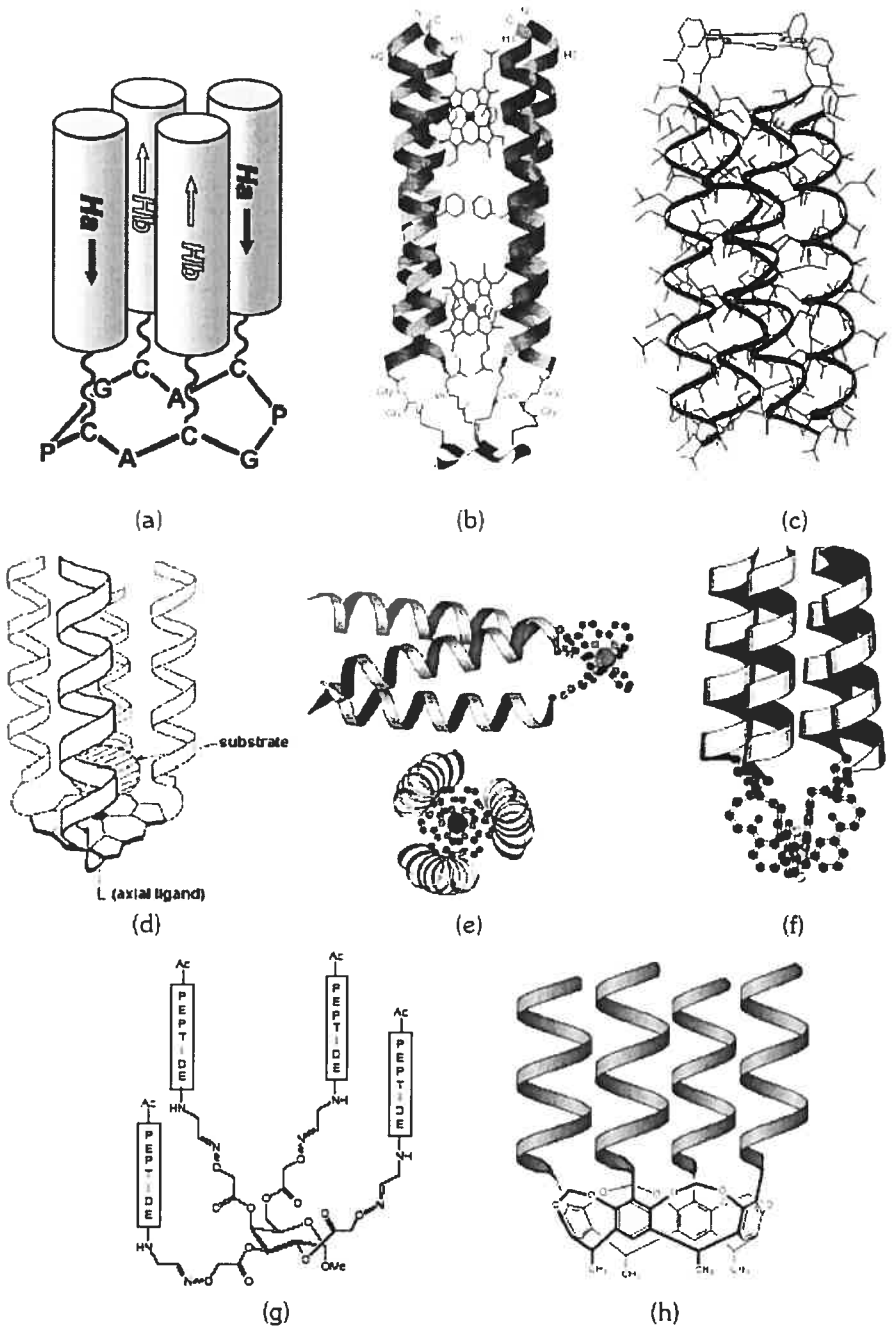

Fig. 2.3 Modèles pour les tonneaux à 4 hélices. Schéma représentant certains modèles (ou supports) utilisées pour construire des échafaudages en forme de tonneau à 4 hélices. Le support choisi dans la conception d'une telle structure définit les propriétés physiques et chimiques de l'échafaudage. Dans (a) le support est un cycle peptidique (P :proline , C : Cysteine, A : Alanine, G : Glycine). Comme (a), le support dans (b) est aussi un cycle peptidique. Cependant deux groupes hèmes additionnels on été ajoutées à l'intérieur de la structure (b). Figure reproduite de [43].

Chapitre 3. Bases de données de protéines

3.1 La base de données PDB

La base de données Protein Data Bank (PDB) a été créée en 1974 et elle est, depuis cette date, la base de données de référence pour ce qui est des structures de molécules biologiques et en particulier de protéines. Elle contient à ce jour 36344 structures. De ces structures, certaines sont néanmoins des structures de molécules identiques mais qui ont été obtenues avec différentes techniques et/ou différentes résolutions. En 2003, un organisme a été créé sous le nom de 'Worldwide Protein Data Bank' (wwPDB) pour veiller à ce que les trois sites PDB qui existent : RCSB PDB (États-Unis), MSD-EBI (Europe) et PDBj (Japon), qui acceptent tous les trois les nouvelles structures provenant en général de leurs régions respectives, offrent une archive de structures unique et gratuite pour la communauté scientifique. Ainsi, une nouvelle structure, qui est déposée par exemple à RCSB PDB, est ultérieurement accessible à partir de n'importe lequel des trois sites. À noter également que chaque structure dans la base PDB possède un identifiant unique qui lui est attribué à son dépôt par les administrateurs de la base PDB. Il s'agit d'un identifiant à 4 caractères alphanumériques (le premier caractère étant un chiffre de 1 à 9) dit identifiant pdb (ou 'pdb id').

Les données dans la base PDB sont structurées à l'aide d'une arborescence (comme dans un système de fichiers classique) où une structure correspond à un fichier sous l'un des trois formats suivants : '.pdb', mmCIF ou XML/PDBML. Dans la prochaine section nous décrivons chacun de ces trois formats de données utilisés par la base PDB.

3.1.1 Formats des données PDB

3.1.1.1 Le format '.pdb'

Ce format de données est historiquement le premier format défini pour décrire les structures de macromolécules biologiques. Même si d'autres formats ont été définis ultérieurement, les 'fichiers .pdb' restent encore le moyen le plus commun pour

communiquer les données sur les structures moléculaires de PDB ainsi que le format le plus communément supporté par les logiciels de visualisation ou de traitement de structures. Un des avantages du format '.pdb' est qu'il est très facile à lire et à interpréter par un utilisateur humain même si ce dernier n'est pas familier avec les données de structures moléculaires. Ce format présente néanmoins le grand désavantage de ne pas être assez 'descriptif' pour permettre une exploitation facile et automatisée (i.e. par l'intermédiaire de programmes) des données de structures. Ceci est particulièrement vrai quand il s'agit d'exploiter les données expérimentales associées aux structures, c'est-à-dire toute l'information associée à une structure PDB et qui est relative à l'expérience ayant conduit à la résolution de la structure (description du processus de cristallisation, références bibliographiques en rapport avec les détails expérimentaux, etc.). Il en résulte que l'automatisation du processus d'analyse, tout en conservant un certain niveau de cohérence, précision et reproductibilité [44], n'est pas aussi aisée qu'elle ne pourrait théoriquement l'être.

À titre d'exemple l'annexe 1 présente un extrait d'un fichier '.pdb'. Chaque entrée *atom* dans cet exemple représente les coordonnées cartésiennes de l'atome dans une base donnée.

3.1.1.2 Le format mmCIF

Le format 'macromolecular Crystallography Information File' (mmCIF) a été introduit pour corriger certaines lacunes dans le format '.pdb'. Ce format s'inspire du format CIF développé par la 'International Union of Crystallographers' (IUCr) et utilisé par les chimistes pour décrire les molécules organiques de petite taille. L'extension à des macromolécules biologiques a été introduite par PDB en 1997. [45]

Lorsque ce nouveau format a été défini, un projet de migration de données depuis le format '.pdb' au format mmCIF a été entrepris par le RCSB PDB. Environ 2000 structures qui étaient alors décrites uniquement en format '.pdb' ont été 'traduites' (à posteriori) en données mmCIF. Les résultats et conclusions de cette migration ont été publiés en 2001 [46]. Cette étude, en plus d'avoir abouti à la publication en format mmCIF (en plus du

format '.pdb') de toutes les structures qui existaient alors, a permis de voir les limites de la description des données structurelles des macromolécules au format '.pdb'.
Le format mmCIF est associé à un 'dictionnaire' qui décrit les éléments qui doivent y être inclus. Ce dictionnaire peut être (et a été par le passé) étendu pour inclure de nouvelles informations. En pratique, les dictionnaires définissent des 'catégories', 'sous-catégories' et autres éléments qui apparaissent dans un fichier mmCIF. Un exemple de catégorie est ATOM_SITE qui donne des informations sur les sites des atomes de la molécule. Ces informations comprennent, entre autres, la position de chaque atome de la molécule définie dans une base cartésienne donnée. Cette catégorie est en outre équivalente aux entrées *atom* du format .pdb (voir la section précédente). L'annexe 2 présente un extrait d'un fichier au format mmCIF.

3.1.1.3 Le format PDBML

Le format Protein Data Bank Markup Language (PDBML) [47] est plus récent que les deux autres formats supportés par PDB décrits précédemment. Il utilise le langage XML en se basant sur un schéma XML/XSD obtenu par traduction du dictionnaire mmCIF. Ainsi une correspondance PDBML a été définie pour chaque élément qui existe dans le format mmCIF avec comme contrainte de conserver la sémantique des données. Une conséquence du renforcement de la correspondance entre les deux formats est que les données PDBML sont moins hiérarchiques que les données habituellement décrites au format XML. En plus du format standard (ou complet), les fichiers PDBML sont aussi fournis (1) sans les entrées de types *atom* et (2) avec les entrées de types *atom* optimisées pour minimiser l'espace utilisé pour les représenter/stocker. L'annexe 3 présente un extrait d'un fichier au format PDBML.

3.1.2 Manipulation des données PDB

Il est à noter que les deux derniers formats décrits dans la section 3.1.1 sont plus verbeux que le format '.pdb' et ce malgré l'optimisation, évoquée antérieurement, qui est introduite dans la 'catégorie' *atom* des fichiers PDBML. Cependant ces deux formats

offrent l'avantage d'être plus précis dans le sens où ils suppriment, dans la plupart des cas, les ambiguïtés qui peuvent exister dans tout langage descriptif et surtout rendent possible (ou facilitent) le traitement des données de structures.

Comme exemple de types de limitations qui existent dans le format '.pdb' mais qui ont été corrigées avec les nouveaux formats, notons la fameuse section 'REMARK' qui, dans les fichiers '.pdb' peut contenir des informations aussi diverses que la résolution obtenue pour une structure ou les spécificités de certains résidus appartenant à la protéine. Même si une 'remarque' qui dit par exemple «this structure is isomorphous to wild type» est facilement compréhensible par un utilisateur humain, la sémantique d'une telle phrase n'est pas aisément accessible à un programme (puisque la formulation d'une telle remarque peut différer selon l'auteur de la structure). Bien évidemment, il est possible de demander au programme de rechercher des mots clés tels que 'isomorphous' et/ou 'wild', mais l'existence de telles termes ne garantit pas que la remarque est bien celle à laquelle on s'attend. En effet, le programme ne saura pas s'il est confronté à de 'nouvelles' remarques. La solution la plus fiable au problème posé ainsi serait de définir une catégorie qui décrirait la nature de la protéine : 'sauvage' ou 'non sauvage' et dans ce dernier cas, éventuellement rajouter une sous-catégorie pour énumérer les mutations que la nouvelle protéine introduit par rapport au type 'sauvage'.

Dans le but de maintenir la qualité de ses données, PDB offre aussi à partir de son site Web (celui de RCSB) plusieurs outils logiciels pour la validation de données de structures (celles qui vont être publiées). Parmi ces outils citons Molprobity [48], PROCHECK [49] et SFCheck [50]. À noter que ces outils ne remplacent pas la curation manuelle effectuée par PDB avant la publication de toute nouvelle structure.

3.2 PDBSum

PDBSum est un site Web qui donne un résumé des caractéristiques d'intérêt incluses dans une structure PDB. Il permet par exemple de donner à son utilisateur un aperçu rapide du contenu d'une protéine donnée en structures secondaires. De plus, une représentation

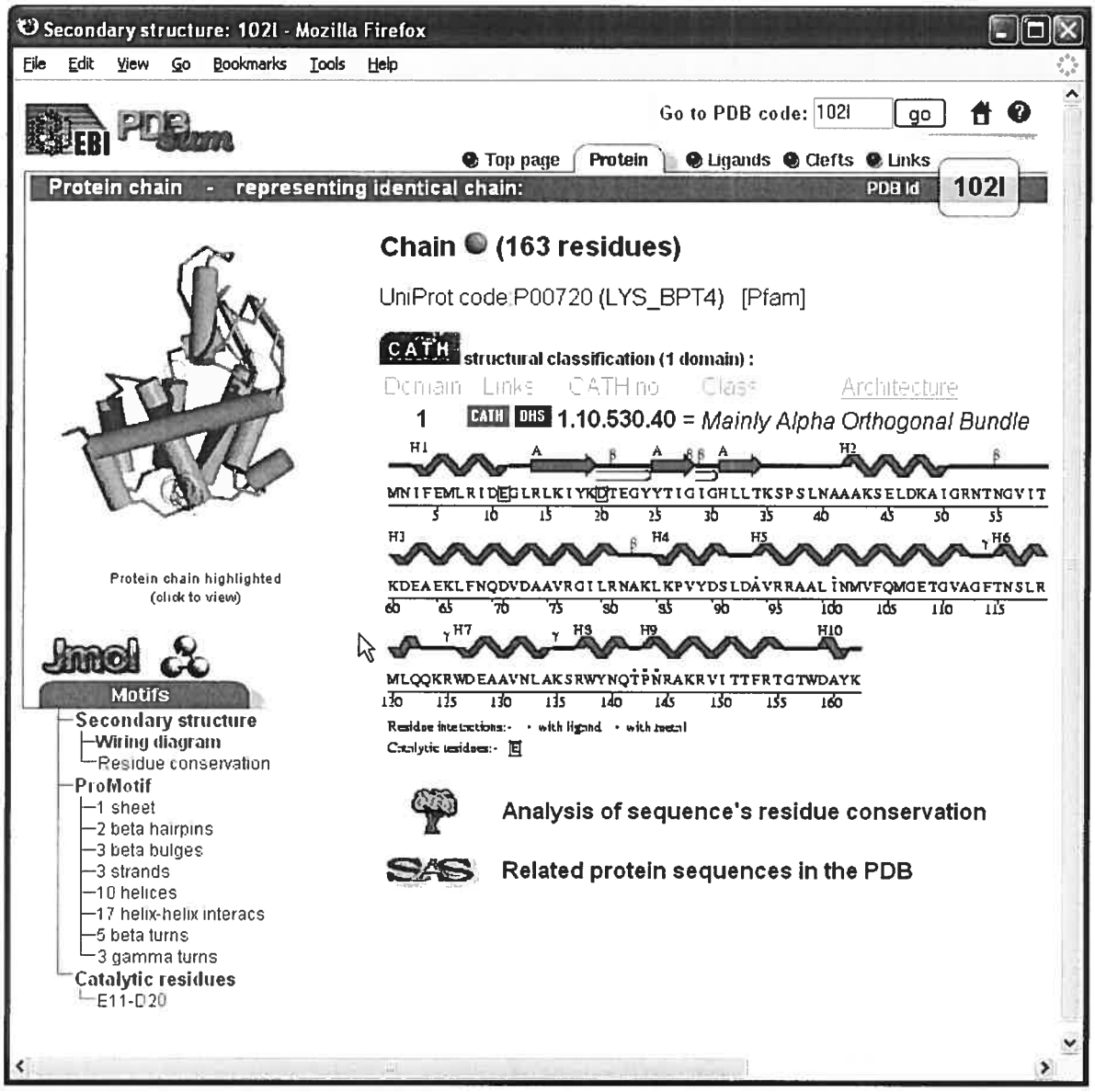

Fig. 3.1 Site Internet PDBSum. Ce site fournit un résumé des caractéristiques d'une structure PDB donnée incluant les structures secondaires qui s'y trouvent.

graphique permet d'identifier les chaînes présentes dans la protéine, et le long de chacune de ces chaînes, les différents éléments de structures secondaires (voir Figure 3.1). PDBSum donne aussi à son utilisateur le moyen de faire une recherche qui couvre toute la base de données PDB mais cette recherche est une simple recherche textuelle et elle est limitée aux seules informations/catégories suivantes trouvées dans un fichier de structure '.pdb' : TITLE, HEADER, COMPND et AUTHOR. Dans Helix Explorer, que nous décrirons en détails au prochain chapitre, nous entendons offrir à l'utilisateur la possibilité de rechercher dans la structure elle-même et non seulement dans les métadonnées qui lui sont associées.

3.3 OpenMMS et PDBase

Récemment, un projet a été lancé par Protein Data Bank pour développer des outils logiciels à sources libres (*open source software*) permettant d'interroger la base de données PDB en utilisant le standard *Corba* [51]. Cette 'boîte à outils logiciels', OpenMMS [52], comprend, entre autres, une base de données relationnelle, PDBase, que nous utilisons dans l'implémentation de Helix Explorer (voir section 4.3). Cette base de données est fournie sous forme d'un schéma relationnel compatible avec les bases de données, relationnelles les plus communes. Elle est aussi accompagnée d'un outil logiciel permettant son chargement à partir de la base de données PDB. Ce dernier outil utilise, pour les raisons décrites précédemment, le format mmCIF au lieu du format '.pdb' pour les structures présentes dans PDB.

La base PDBase définit un tableau par catégorie mmCIF et dans chacun de ces tableaux une colonne représente une sous-catégorie. Au total, 177 tableaux sont créés dans la base de données PDBase. Une fois la base de données PDB chargée, le plus grand des tableaux de PDBase (en espace occupée sur le disque et aussi en nombre de registres ou lignes) est celui associé à la catégorie ATOM_SITE. Ce tableau n'est autre que le tableau contenant les coordonnées géométriques de chaque atome dans chacune des structures de PDB (équivalent de l'entrée *atom* du format '.pdb'). En effet, que ce soit à l'échelle d'une seule

structure ou à l'échelle de toute la base PDB, les informations expérimentales (i.e. liées à la conduite de l'expérience de résolution de la structure) ainsi que les informations dérivées (telles que l'annotation des structures secondaire présentes dans la protéine) restent 'minoritaires' comparées à la description géométrique de la structure macromoléculaire en trois dimensions.

Outre les données PDB, PDBase contient aussi les méta-informations qui décrivent chacun des tableaux qui y sont contenu. Il est ainsi possible d'interroger PDBase pour savoir, par exemple, quelle tableau/colonne de PDBase représente une catégorie/sous-catégorie mmCIF donnée. De la même manière, il suffit de faire une simple requête SQL à la base PDBase elle-même, pour obtenir une description textuelle (en anglais) permettant de comprendre à quoi correspond chaque tableau/colonne de la base de données. Certaines catégories, malgré qu'elles soient définies dans le dictionnaire mmCIF, ne sont que très peu utilisées (parfois jamais) par les auteurs des structures. C'est le cas par exemple de la catégorie STRUCT_BIOL_VIEW qui, d'après le standard mmCIF, peut contenir les détails servant à dessiner et à annoter une vue didactique de la structure biologique. Aucune structure de celles que nous avons chargées dans PDBase ne déclare cette catégorie. D'où la grande disparité relevée précédemment dans les dimensions des tableaux/catégories de PDBase.

Un des avantages de l'utilisation de la base de données PDBase (au lieu d'accéder aux données brutes telles que publiées par PDB) est qu'elle facilite grandement l'obtention d'informations recherchées si l'on s'intéresse non pas seulement à une structure PDB particulière mais que l'on souhaite interroger l'ensemble de la base PDB. Ceci est illustré dans l'exemple des réponses aux questions 2 et 3 qui suivent et qui sont adressées à PDBase sous forme de requêtes SQL.

1. Quel est le centre d'une molécule ?

```
select MMS_ENTRY.id as PDB_ID, count(*) as atoms,
  avg(cartn_x), avg(cartn_y), avg(cartn_z)
from MMS_ENTRY, ATOM_SITE
where MMS_ENTRY.entry_key=ATOM_SITE.entry_key
  and MMS_ENTRY.id = '1hwt'
group by MMS_ENTRY.id ;
```

2. ***Comment trouver tous les registres (entrées) de la base PBD où un auteur particulier est cité(e)?***

```
select MMS_ENTRY.id as PDB_ID, name
from MMS_ENTRY, CITATION_AUTHOR
where MMS_ENTRY.entry_key=CITATION_AUTHOR.entry_key
    and CITATION_AUTHOR.name like '%Bancroft%'
group by MMS_ENTRY.id, name ;
```

3. ***Quelles sont les entrées PDB qui contiennent une catégorie mmCIF donnée ?***

```
select id, len from MMS_ENTRY, MMS_ENTRY_CATEGORIES
where MMS_ENTRY.entry_key=MMS_ENTRY_CATEGORIES.entry_key
and category_key =(select category_key from MMS_CATEGORY where
id='struct_ncs_oper' );
```

L'extraction d'informations telles que celles-ci est aussi possible sans passer par la base de données relationnelle PDBase mais elle est sûrement beaucoup plus longue et fastidieuse.

Chapitre 4. Helix Explorer

4.1 Motivations et approche générale

Lors de la conception rationnelle d'une nouvelle protéine, il est important de comprendre les interactions locales qui sont établies entre les différentes régions de cette dernière et les règles qui gouvernent son repliement. Ces interactions sont attribuées à des forces internes telles que les forces hydrophobes, les forces électrostatiques ou les ponts hydrogènes établis entre les chaînes latérales de deux acides aminées voisins etc. D'autre part, tant que le problème de la prédiction de la structure tertiaire des protéines ne sera pas définitivement résolu, les deux approches principales suivantes - qui ne sont pas mutuellement exclusives - seront utilisées pour comprendre et prédire ce type d'interactions :

(1) études utilisant des simulations basées sur des méthodes de mécanique et dynamique moléculaire [19] et/ou mécanique quantique [20]. Sans mentionner les différences importantes qui peuvent exister entre ces méthodes, il est possible d'affirmer qu'en général elles nécessitent toutes des ressources computationnelles relativement importantes et ne fournissent des résultats acceptables que dans le cas où les structures étudiées sont de petite taille. En effet, la qualité des prédictions obtenues en utilisant ces méthodes se dégrade très rapidement avec l'augmentation de la taille des structures.

(2) étude comparatives : l'idée derrière ce type d'études est qu'il est possible d'utiliser l'information qui existe sur les structures de protéines qui ont été résolues pour prédire les structures de nouvelles protéines qui leurs sont 'proches' [21]. Ceci sous entend que de petites modifications dans la séquence d'un peptide sous certaines conditions – comme par exemple le remplacement

d'un acide aminé par un autre qui lui est similaire[8] - ne résulte pas en de grands changements dans la structure finale adoptée par la protéine. C'est d'ailleurs cette même idée qui est exploitée dans l'utilisation d'échafaudages dans la conception de nouvelles protéines (voir section 2.3). D'autre part, certaines méthodes de prédiction basées sur les techniques d'apprentissage machine, par programmation par logique inductive [53,54] ou d'autres approches (réseaux de neurones par exemple) peuvent être regroupées dans cette catégorie puisqu'elles reposent sur la connaissance de structures existantes.

Helix Explorer, qui est présenté dans ce chapitre, trouve son application dans le domaine de l'ingénierie des protéines en rendant plus accessible la conception de nouvelles protéines selon l'approche comparative (point (2) qui précède). D'autre part, il est également important de pouvoir prédire la structure secondaire d'une séquence d'acides aminés, sans faire appel à des algorithmes de prédiction de structure. Beaucoup de groupes de recherches dont les activités de recherches sont dédiées à la création d'assemblages peptidiques, sont intéressés à pouvoir utiliser des motifs de structure secondaire, tels que retrouvés dans les protéines cristallisées, pour les associer entre elles afin d'obtenir des 'protéines synthétiques simplifiées'. Dans notre approche, l'outil que nous avons développé permet de rechercher dans la base PDB tous les motifs de structure secondaire correspondant à une séquence primaire donnée. Dans la suite, nous examinons tout d'abord les idées et principes derrière la conception d'Helix Explorer avant de décrire en détails ses fonctionnalités et son implémentation.

---

[8] Deux acides aminés sont dits similaires s'ils possèdent des propriétés physicochimiques semblables et sont de ce fait interchangeables dans certaines situations.

4.1.1 Les structures secondaires comme structures élémentaires des protéines

Il est très commun de manipuler les protéines en les polymères (ou les séquences) d'acides aminés (voir section 1.2.1) qu'elles sont. Ceci est particulièrement vrai lorsqu'on est intéressé uniquement par la structure primaire d'une protéine. En génomique par exemple, les scientifiques sont régulièrement portés à rechercher dans les bases de données de séquences protéiques (que ces dernières comprennent uniquement des protéines effectivement séquencées ou aussi des protéines hypothétiques dont les séquences ont été obtenues par 'traduction' de séquences génomiques) les séquences similaires à une séquence protéique donnée. Cependant, du fait qu'il existe une relation importante entre la structure (sous entendu structure tertiaire/quaternaire) et la fonction d'une protéine, il est aussi important de pouvoir effectuer des recherches dans une base de données de structures. On pourrait, par exemple, vouloir rechercher dans la base des structures PDB une 'configuration structurelle' bien particulière où trois acides aminés (définis par leurs types) sont positionnés d'une manière bien précise les uns par rapport aux autres. Une telle recherche serait envisageable si l'on souhaitait par exemple identifier une 'triade catalytique'[9] spécifique dans l'ensemble de la base PDB. Pour parvenir à cette fin, il est certes possible d'indexer l'ensemble de configurations qui incluent les trois acides aminés en question, mais une telle indexation, en plus d'être coûteuse, aurait peu d'utilité en dehors du contexte pour lequel elle aurait été initialement réalisée : à titre d'exemple, il ne serait pas possible de réutiliser le même index si l'on désirait rechercher des configurations identiques à celles décrites précédemment mais qui engagent trois acides aminées de types différents. Dans cet exemple, indexer toutes les combinaisons de trois acides aminés de la base PDB serait certainement plus utile, mais serait aussi prohibitif du point de vue computationnel.

---

[9] Cette expression désigne en général un triplet de résidus (acides aminés) qui participent au mécanisme catalytique d'une enzyme. Elle est souvent associée à une protéase telle que la chymotrypsine par exemple.

L'approche que nous suggérons consiste à représenter une protéine en tant que collection de structures secondaires reliées entre elles par certaines métriques que nous définissons. Cela a pour conséquence immédiate de réduire considérablement le nombre de 'briques élémentaires' formant la protéine qui seraient alors, non plus les acides aminés, mais les structures secondaires, puisqu'il existe manifestement dans une structure PDB beaucoup moins de structures secondaires que d'acides aminés. Les métriques définies par Helix Explorer comprennent, entre autres, la 'distance' entre deux structures secondaires qui rend compte de la proximité entre deux structures secondaires à l'intérieur de la même protéine, l'angle entre les deux axes respectifs de deux hélices etc. En plus de ces informations, qui relient entre elles deux structures secondaires, il est aussi important d'avoir l'information sur la proximité d'une structure secondaire à la surface moléculaire de la protéine qui la contient. En effet, comme expliqué dans la section 1.2.5, des interactions entre une structure secondaire et l'extérieur de la protéine sont possibles uniquement si la structure secondaire se trouve à proximité de la surface moléculaire.

Remarquons cependant que la description d'une structure protéique en tant que collection de structures secondaires et de relations entre ces dernières, telle que suggérée ici, résulte irrémédiablement en la perte d'information sur les détails structuraux à l'échelle atomique qui sont associés à toute structure de la base PDB. Cette perte d'information n'est néanmoins pas aussi dramatique qu'elle pourrait le sembler à première vue. En effet, les structures secondaires dans les protéines, du moins les hélices et les feuillets β, ont des géométries régulières, ce qui signifie qu'il n'est pas nécessaire de connaître la position de chaque acide aminé/atome pour 'visualiser' l'espace occupé par la structure secondaire toute entière. En d'autres termes, la seule connaissance du type d'une structure secondaire et de certains de ses paramètres (e.g. sa distance à d'autres structures secondaires, les acides aminés qui la forment ou, s'il s'agit d'une hélice, l'angle qu'elle établit avec d'autres hélices etc.) est dans bien des cas suffisante pour confirmer ou exclure l'éventualité d'interactions de la structure secondaire avec son voisinage.

Dans la section qui suit, nous décrivons les métriques introduites par Helix Explorer. Une description détaillée de l'implémentation de ces métriques est fournie dans la section 4.3.4.

4.1.2 Métriques utilisées par Helix Explorer

4.1.2.1 Distances entre structures secondaires

Deux structures secondaires présentes à l'intérieur d'une même protéine peuvent se trouver à proximité immédiate l'une de l'autre et établir, éventuellement, des interactions entre elles par le moyen de ponts hydrogènes par exemple, ou, au contraire, être localisées dans des régions 'éloignées' (à l'échelle des distances de la protéine) rendant impossible toute interaction directe entre elles. Pour rendre compte de la notion de 'proximité' entre deux structures secondaires, Helix Explorer introduit trois définitions de distances, chacune offrant une vue partielle du positionnement relatif d'une structure secondaire par rapport aux autres structures secondaires présentes dans la même protéine. Combinées, ces trois distances offrent un aperçu global de l'agencement de deux structures secondaires quelconques à l'intérieur de la protéine. Ces trois distances sont : (1) la distance minimale $D_{\min}$, (2) la distance des centres D_{ctr} et (3) la distance moyenne D_{moy} :

(1) La distance minimale $D_{\min}$ mesure à quel point deux structures secondaires se retrouvent proches l'une de l'autre à leurs points respectives les plus proches. En d'autres termes, c'est la plus petite distance (euclidienne) qui sépare deux carbones α chacun faisant partie d'une des deux structures secondaires.

(2) La distance des centres D_{ctr} est définie comme étant la distance entre les deux centres géométriques des deux structures secondaires. Par centre géométrique d'une structure secondaire, nous entendons le centre de l'ensemble de ses carbones α. Cette distance mesure la distance 'moyenne' entre deux structures secondaires lorsque ces dernières sont vues comme deux 'nuages' distants de points (où un point représente un carbone α), mais

ne fournit pas d'information sur l'éventualité d'interactions entre les deux structures. Les deux centres peuvent être proches sans pour autant que les carbones α de chacune des structures secondaires soient 'en moyenne' proches de l'autre structure secondaire.

(3) La distance moyenne D_{moy} se veut être une mesure *moyenne* de la distance entre une structure secondaire et une autre structure secondaire. En particulier, dans le cas où les deux structures sont des hélices, cette distance est définie de façon à être optimale lorsque les axes des deux hélices sont parallèles et se trouvent le plus proches possibles l'une de l'autre. Concrètement, le calcul de cette distance est effectué comme suit : la plus petite distance de chaque carbone α de la structure secondaire la plus courte (c'est-à-dire celle contenant le plus petit nombre d'acides aminés) à l'autre structure secondaire est mesurée. La distance D_{moy} est alors la moyenne de ces distances sur l'ensemble de tous les carbones α de la 'petite' structure secondaire. Lorsque les deux structures secondaires ont la même taille (c'est-à-dire qu'elles ont exactement le même nombre d'acides aminés), une des deux structures, au hasard, est considérée pour des fins de calcul comme étant la 'plus petite'. À noter que, du fait de l'existence de cette dernière incertitude sur le choix de la structure secondaire employée pour le calcul de D_{moy}, la distance moyenne, telle que définie précédemment, n'est pas une relation symétrique et par conséquent n'est pas une 'distance' dans le sens mathématique.

Formellement les distances $D_{\min}$, D_{ctr} et D_{moy} entre deux structures secondaires $S=\{(\alpha_{S_i}), i \le m\}$ et $P=\{(\alpha_{P_j}), j \le n\}$, où α_{S_j} désigne le carbone α a la position i de S, sont définis par les équations suivantes :

$$D_{\min}(S,P) = \min\{d(\alpha_{S_i}, \alpha_{P_j}) \mid i \le m, j \le n\}$$

$$D_{ctr}(S,P) = d(Centre(S), Centre(P))$$

$$D_{moy}(S,P) = \left(\sum_{i=1}^{m} \min\{d(\alpha_{S_i}, \alpha_{P_j}) \mid j \le n\} \right) / m$$

où d est la distance cartésienne et *Centre(S)* désigne le centre géométrique de la structure secondaire S :

$$d(\alpha_{S_i}, \alpha_{P_j}) = \sqrt{(x_{S_i} - x_{P_j})^2 + (y_{S_i} - y_{P_j})^2 + (z_{S_i} - z_{P_j})^2}$$

$$Centre(S) = \left(\sum_{i=1}^{m} \alpha_{S_i} \right) / m$$

La Figure 4.1 donne une représentation visuelle des distances entre deux structures secondaires, en l'occurrence deux hélices dans cet exemple.

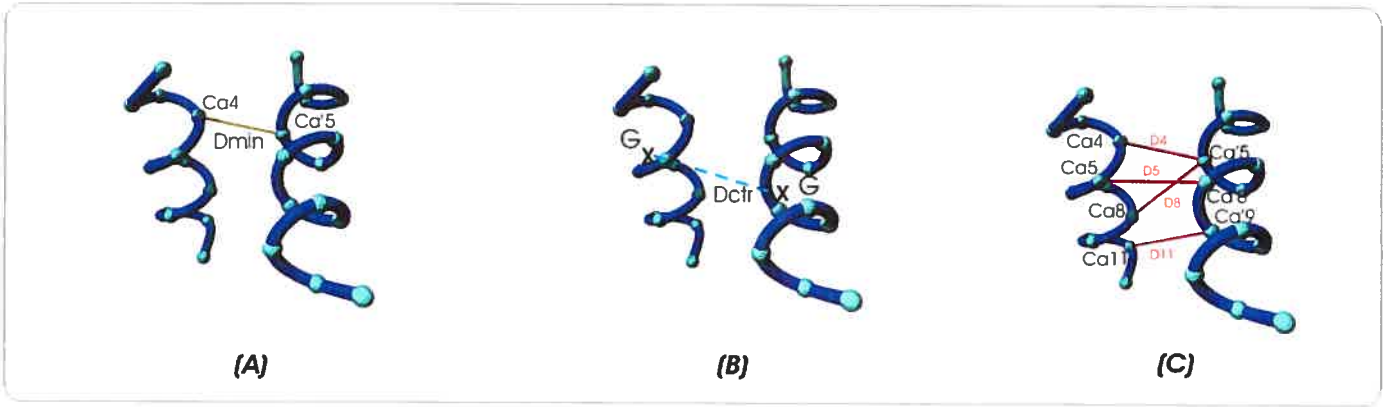

Fig.4.1 Distances dans Helix Explorer. Schéma représentant les trois distances entre deux hélices dans Helix Explorer : (A) $D_{\min}$ *est la distance minimale (B)* D_{ctr} *est la distance entre les centres et (C)* D_{moy} *est calculée en utilisant la formule suivante (D1+...+D12)/12, ou Di est la distance minimale entre Cai* (C_{α_i}) *et l'autre structure secondaire.*

4.1.2.2 Angles définis entre deux structures secondaires

L'angle entre deux hélices se trouvant dans la même protéine est une autre métrique considérée par Helix Explorer. Cet angle est défini comme étant l'angle entre les axes

respectifs des deux hélices (voir Figure 4.2). Les axes des hélices sont calculés en utilisant la méthode *parametric least-square* décrite en détails dans la section 4.3.3.3.

À noter qu'une autre métrique que l'on aurait pu considérer dans Helix Explorer est celle définissant l'angle entre deux feuillets β ou entre un feuillet β et une hélice. La définition d'une telle métrique aurait été possible (et pertinente) si les feuillets β qui se trouvent dans la base PDB étaient majoritairement proches de la structure idéale d'un feuillet β telle que décrite dans la section 1.4.2.4. Plus précisément, si les feuillets se trouvaient être planes. Cependant, en examinant de près un échantillon relativement restreint et aléatoire d'entrées de la base PDB, il apparaît clairement que les feuillets β sont rarement proche des structures idéales et ne peuvent, dans la grande majorité des cas, être considérés planaires.

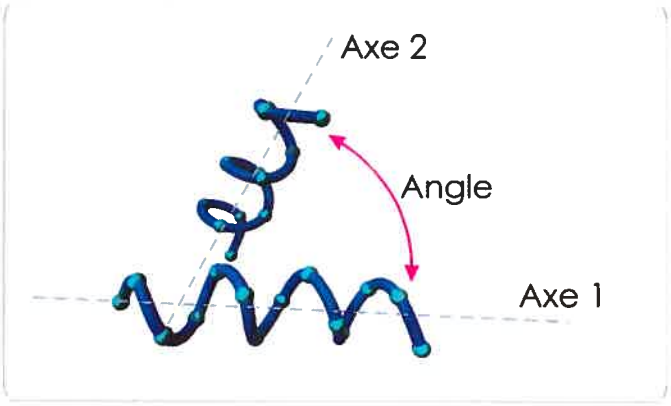

Fig.4.2 Angles entre hélices. Schéma représentant l'angle entre deux hélices dans Helix Explorer. L'axe d'une «hélice» donnée est celui de la structure hélicoïdale idéale qui s'en rapproche le plus. L'angle entre les deux hélices est l'angle non orienté (entre 0° et 180°) qui existe entre les projections des axes des deux hélices sur un plan qui est parallèle aux deux axes. C'est aussi (comme calculé dans l'implémentation de Helix Explorer) l'arccosinus du produit scalaire de deux vecteurs unitaires chacun définissant l'axe d'une des hélices.

4.1.2.3 Distance d'une structure secondaire à la surface moléculaire

À l'instar des distances définies précédemment, qui rendent compte de la 'proximité' entre deux structures secondaires, Helix Explorer introduit de la même façon la distance entre une structure secondaire et la surface moléculaire de la protéine qui la

renferme. Pour que cette dernière distance conserve une 'sémantique' identique à celles des distances entre structures secondaires, Helix Explorer procède dans le calcul de cette distance comme suit : (1) d'abord en réduisant la surface moléculaire à une simple structure secondaire d'un 'autre type' que les hélices, les feuillets β ou les tours (voir la section 4.3.3.2 décrivant l'implémentation) et (2) ensuite en utilisant la définition de distance D_{moy} pour calculer la distance entre les deux structures secondaires que sont, d'une part la structure secondaire qui nous intéresse, et d'autre part la pseudo-structure secondaire qui remplace la surface moléculaire. À noter que, comme pour les distances entre structures secondaires, chacune des trois distances définies précédemment fournirait une vue partielle du positionnement d'une structure secondaire par rapport à la surface moléculaire. À défaut d'avoir un parametre numérique qui à lui seul décrirait la «vue globale», nous optons ici pour l'utilisation d'une des trois distances (D_{moy}). Le choix de D_{moy} plutôt que $D_{\min}$ ou D_{ctr} est justifié par le fait que c'est la seule des trois distances dont la mesure rapportée exprime 'le degré d'exposition' de l'ensemble de la structure secondaire vis-à-vis de l'extérieur de la protéine. En effet, d'une part, la distance $D_{\min}$ en dit seulement sur le rapprochement optimal entre la structure secondaire et la surface moléculaire et serait ainsi trompeuse dans le cas où un seul acide aminé de la structure secondaire se trouve 'très proche' de la surface moléculaire alors que le reste de la structure secondaire se trouve enfouie à l'intérieur de la protéine. D'autre part, la valeur de D_{ctr} n'est pas informative du fait que la surface moléculaire n'est pas une entité géométriquement séparée comme c'est le cas pour les structures secondaires mais elle enveloppe l'ensemble de la protéine. Ainsi, calculer la distance d'une structure secondaire au 'centre de la surface' n'apporte en soit aucune information supplémentaire, à moins de connaître aussi les dimensions de la protéine et la 'forme' de la surface moléculaire elle-même.

4.2 Interface utilisateur d'Helix Explorer

Dans cette section nous allons présenter en détails l'interface utilisateur d'Helix Explorer. Mais tout d'abord, énumérons les fonctionnalités offertes par cette interface :

(1) Permet la recherche d'une séquence d'acides aminés dans toute la base de données des structures secondaires de PDB. Cette recherche est 'exacte' dans le sens où Helix Explorer ne retourne que les séquences qui sont identiques à la séquence entrée par l'utilisateur. En d'autres termes, contrairement à d'autres outils de recherche dans les bases de données de séquences, tels que BLAST[10], la recherche implémentée par Helix Explorer ne s'étend pas à une recherche par similarité. Cela peut sembler très restrictif mais d'autres fonctionnalités (voir le point 2 qui suit) permettent de contourner les limitations imposées par ce choix de conception. Voir Figure 4.3.

(2) Permet la recherche de séquences dans la base de structure secondaires en utilisant des 'expressions régulières'. Si l'utilisateur souhaite par exemple trouver toutes les structures secondaires de longueur au moins 9 acides aminées dans lesquelles au moins deux acides aminés de types hydrophiles se trouvent aux positions *n* et *n+4* et où la première des deux occurrences est suivie, dans cet ordre, par une proline (P) et deux acides aminés indéfinis, alors que la deuxième occurrence est respectivement suivie par une arginine (R), un glutamate (E) et deux acides aminés indéfinis, il lui suffit d'entrer dans Helix Explorer comme requête l'expression suivante : *(N)P($)($)(N)RE($)($)*. Dans cette expression, *(N)* indique qu'à la position correspondante se trouve un acide aminé hydrophobe alors que *($)* remplace n'importe lequel des vingt acides aminés. Comme mentionné dans le point (1) ceci permet d'ajouter un 'semblant' de recherche par similarité même si la recherche reste déterministe et ne retourne en fin de compte que les séquences qui contiennent une sous-séquence dont toutes les positions sont décrites par l'expression régulière. Cette fonctionnalité, nous pensons, rend la recherche par similarité redondante étant donné le type d'usage que nous prévoyons pour Helix Explorer. À savoir, où la plupart des séquences recherchées sont supposées être relativement courtes et ne laissent donc pas de place à de

[10] Basic Local Alignment Search Tool [72]

longues insertions. De plus, même s'il s'avérait nécessaire de représenter ces insertions, elles pourraient être 'simulées' par l'introduction de caractères *($)* dans l'expression régulière.

(3) Lister pour chaque structure PDB, identifiée par son 'pdb id', les éléments de structures secondaires qui s'y trouvent. Cette fonctionnalité est similaire à celle retrouvée dans PDBSum (voir la section 3.2).

(4) Indique pour chaque structure secondaire retournée par Helix Explorer, certaines informations qui lui sont associées, tels que la liste des entrées PDB qui renferment exactement la même séquence, l'identifiant de l'unité moléculaire qui contient la séquence, si la protéine est formée de plusieurs unités (cet identifiant est généralement une lettre majuscule de l'alphabet), etc. Helix Explorer indique aussi si la structure secondaire se trouve à la surface de la protéine qui la renferme (voir section 4.4) ou dans le cas d'une hélice, si cette dernière se trouve en tonneau[11] avec une autre hélice.

(5) Affiche pour chaque séquence de structure secondaire tous ses 'voisins'. Par 'voisins', nous entendons toutes les structures secondaires qui partagent avec la séquence initiale au moins une structure PDB ou en d'autres mots, toutes les structures secondaires qui apparaissent dans au moins une des protéines où la séquence en question est présente. De plus, pour chacun des voisins, Helix Explorer affiche certains paramètres qui relient la paire de voisins.

---

[11] Deux hélices forment un tonneau dans Helix Explorer si elles sont distantes d'au plus 10Angströms et qu'elles forment entre elles un angle d'au plus 10 degrés.

L'interface utilisateur d'Helix Explorer est accessible par l'intermédiaire de n'importe quel fureteur sous condition que ce dernier supporte les scripts clients JavaScript [55] (voir section 4.3). L'interface est fournie sous forme de 3 vues principales :

(1) *Vue des séquences* : cette vue est représentée par les pages web retournées suite à une recherche dans Helix Explorer. Cette recherche peut être explicite lorsque l'utilisateur rentre manuellement la séquence à rechercher ou le 'pdb id' de la structure PDB à explorer. Elle peut aussi être implicite lorsque l'utilisateur clique sur un lien de type 'pdb id'. Voir Figure 4.4.

(2) *Vue des voisins* : cette vue affiche pour chaque séquence résultante d'une recherche avec Helix Explorer, tous ses voisins dans chacun des voisinages (i.e. entrée PDB où une séquence qui lui est identique est présente). En plus de fournir une liste des voisins, il est possible de filtrer les résultats selon plusieurs critères. La vue est mise à jour automatiquement en fonction des critères sélectionnés par l'utilisateur. Voir Figure 4.5.

(3) *Vue des structures secondaires* : renferme tous les détails sur une structure secondaire donnée. Cette vue est intégrée aux deux vues précédentes. Pour y accéder, il suffit à l'utilisateur de cliquer sur une séquence apparaissant dans une des deux vues (1) ou (2). Une section supplémentaire s'affiche alors dans la même page. Cette stratégie permet à l'utilisateur de requérir les détails de plusieurs structures secondaires sans avoir à quitter la vue initiale. Voir Figure 4.6

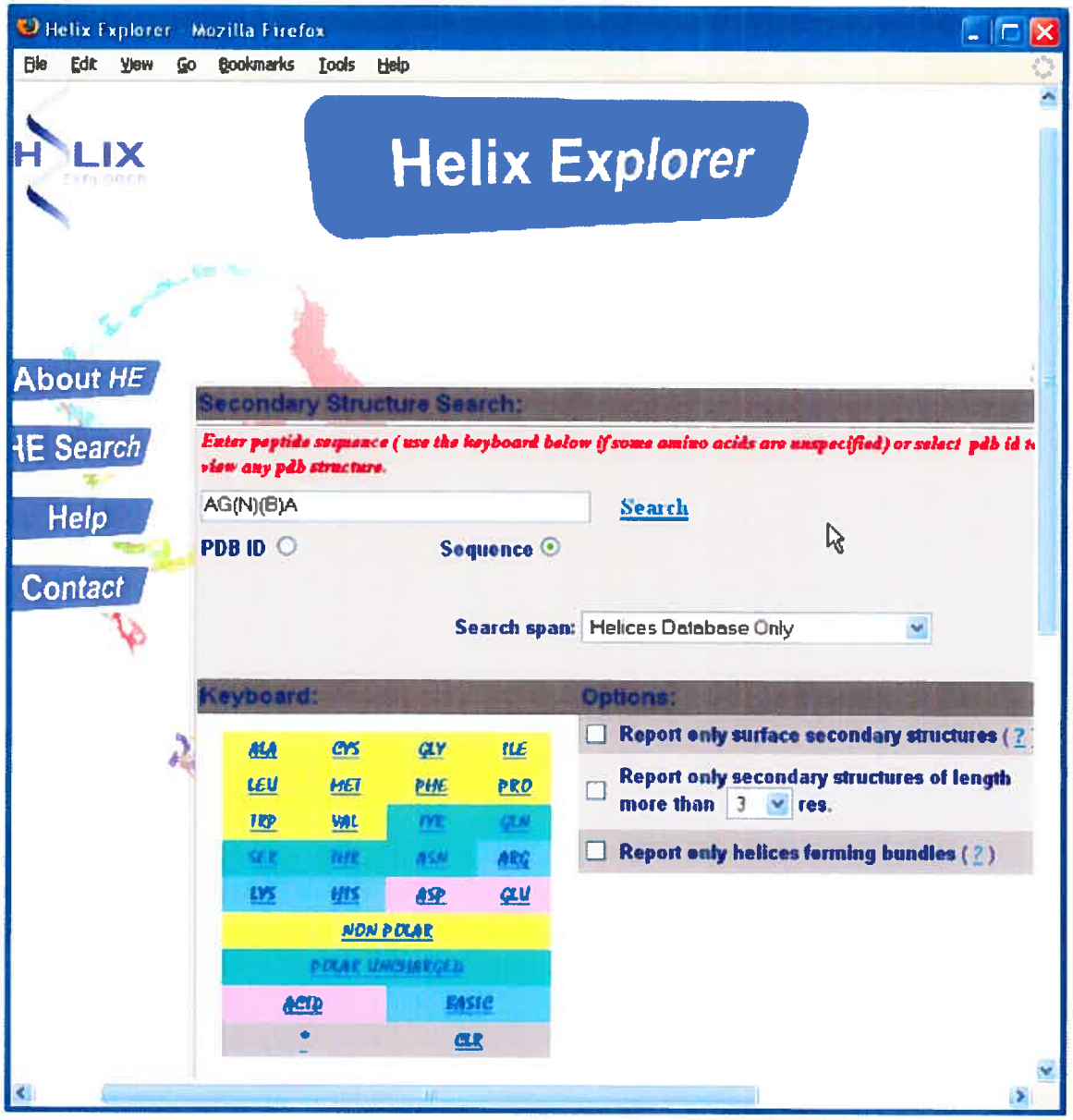

Fig.4.3 Page de recherche de l'interface utilisateur d'Helix Explorer.

En plus de fournir les trois vues décrites plus haut, Helix Explorer contient des liens vers des pages du site de PDB. Chacun de ces liens externes qui pointe sur la page décrivant l'entrée correspondante dans le site RCSB PDB est indiqué sous la forme *'PDB : pdb id'*. Le lien *'PDB : 102l'* par exemple pointe sur l'entrée dans le site RCSB PDB qui décrit la

Fig.4.4 Vue des séquences dans Helix Explorer.

structure *102l*. À noter que lorsqu'un lien similaire n'est pas précédé par *'PDB :'*, c'est-à-dire qu'il indique uniquement l'identifiant pdb, un clic sur ce lien est interprété par Helix Explorer comme une requête implicite qui doit retourner la liste des structures secondaires de l'entrée PDB en question.

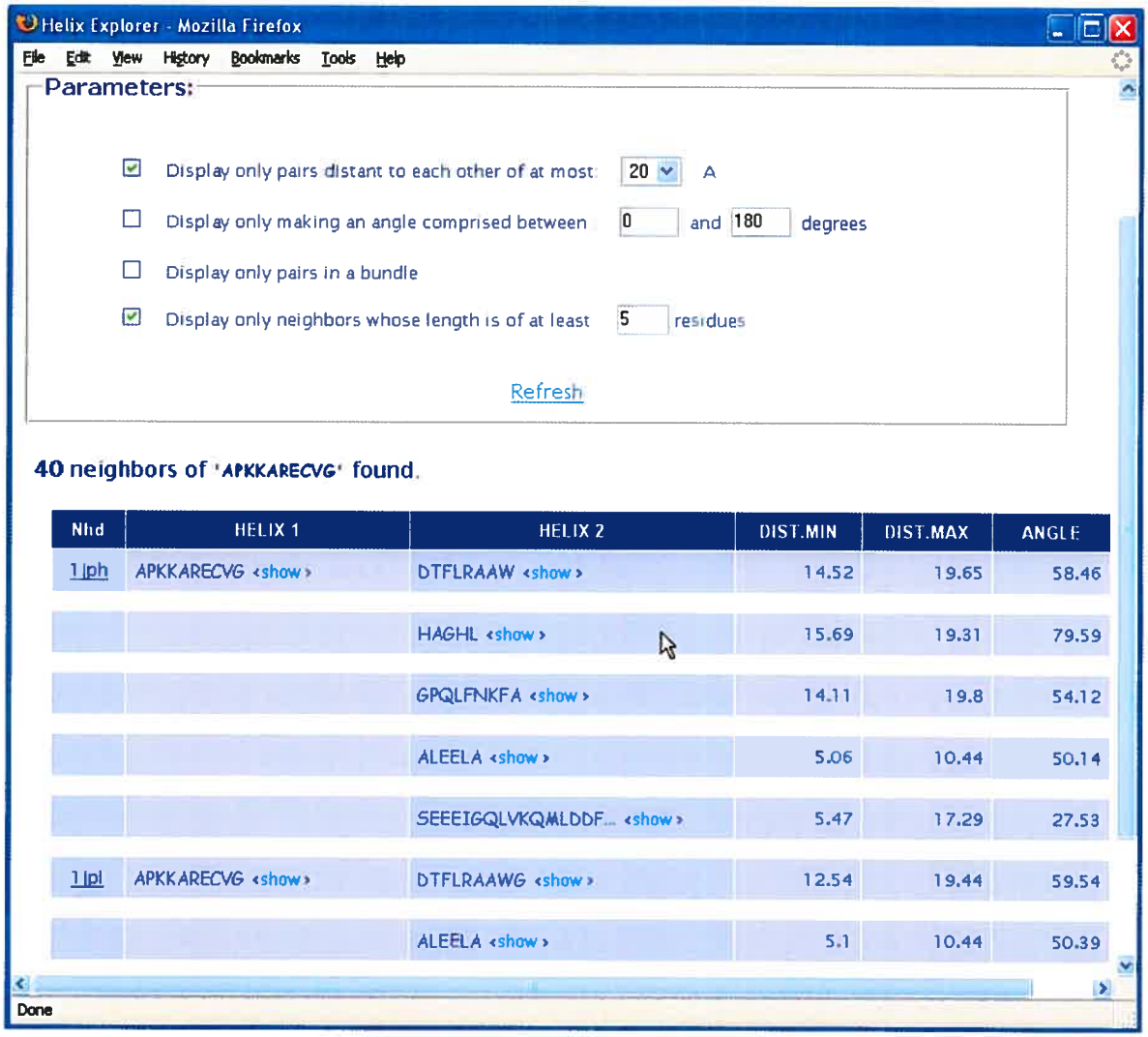

Fig. 4.5 Vue des voisins dans Helix Explorer

Plus la séquence recherchée est courte et fréquemment rencontrée dans les protéines et plus elle contient de positions indéfinies (si la séquence est exprimée comme expression régulière), plus le nombre de résultats retournés peut être important. Helix Explorer affiche les 20 premiers résultats trouvés et offre des liens 'suivant' et 'précédent' pour requérir les 20 résultats qui suivent ou précédent les résultats apparaissant sur la page courante. À noter

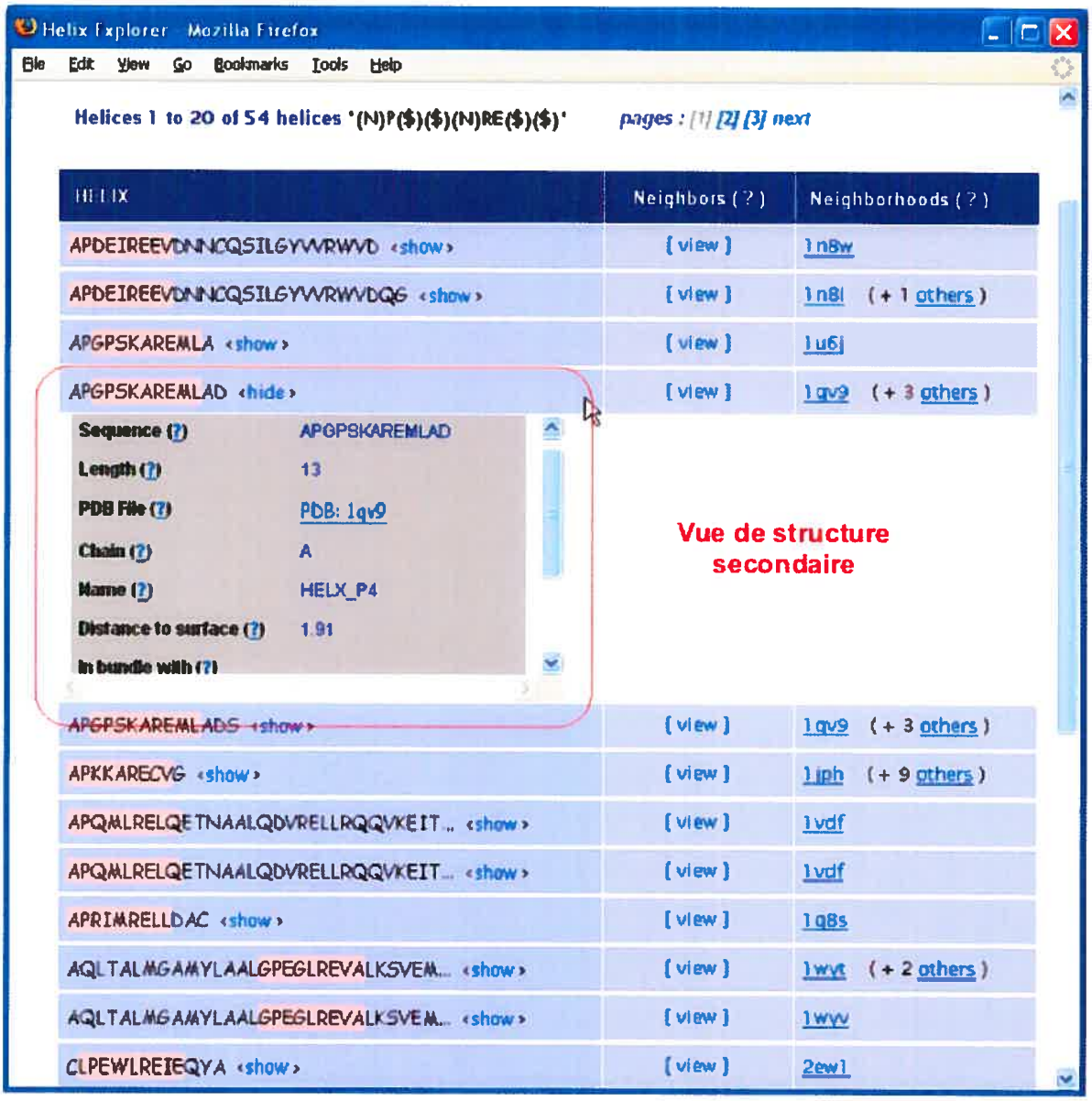

Fig.4.6 Vue des structures secondaires dans Helix Explorer. Cette vue est intégrée aux vues de séquences et des voisins.

que le nombre total de résultats est indiquée en haut de chaque page permettant ainsi à l'utilisateur d'affiner sa recherche si le nombre de résultats obtenus est trop élevé ou, au contraire, relaxer les contraintes sur la séquence recherchée si le nombre de résultats obtenus est insuffisant.

4.3 Architecture et implantation

Helix Explorer est constitué de trois composants que sont (1) la base de données d'Helix Explorer qui renferme les données utilisées par Helix Explorer, (2) l'interface Web d'Helix Explorer permettant l'interrogation de la base de données Helix Explorer en utilisant n'importe quel fureteur et (3) le composant 'Constructeur de la base de données Helix Explorer' (Helix Explorer Database Builder) qui permet de construire et de mettre à jour la base de données Helix Explorer avec les données PDB. Dans ce qui suit nous détaillons l'implémentation de chacun de ces trois composants (voir Figure 4.7).

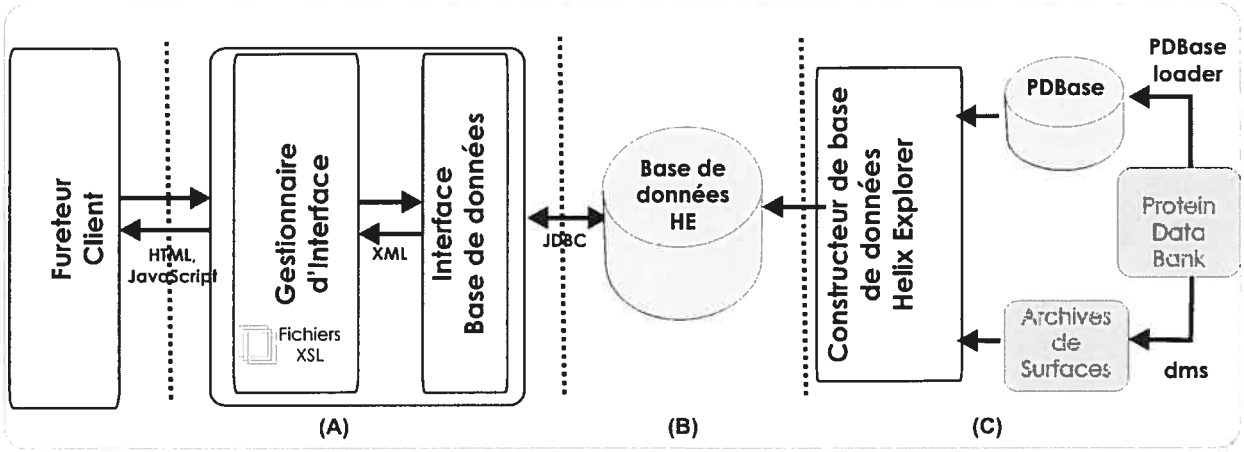

Fig.4.7 Architecture d'Helix Explorer. Helix Explorer est formé de trois composants : L'interface Web d'Helix Explorer (A), la base de données d'Helix Explorer (B) et son constructeur (C).

4.3.1 Base de données Helix Explorer

Cette base de données est construite à partir de la base PDBase. Les données sont principalement produites dans deux tableaux :

Le premier contient un registre (ou ligne) par structure secondaire de la base PDB. À cette structure sont associées plusieurs colonnes d'information : un identifiant unique de la structure (qui est aussi la clé du tableau), la séquence en acides aminés de la structure, et d'autres informations telles que la distance de la structure secondaire à la surface de la molécule et, s'il s'agit d'une hélice, la direction (en utilisant la même base de coordonnées cartésiennes que la base PDB) de l'axe de l'hélice.

Le deuxième tableau, des 'voisins', contient l'ensemble de toutes les paires de structures secondaires qui font partie de la même entrée PDB et pour chacune de ces paires les informations qui sont propres à la relation entre les deux structures de la paire. Si les deux structures sont des hélices par exemple, certaines colonnes sont utilisées pour conserver des distances mesurées entre les deux hélices ainsi que l'angle que leurs deux axes font ensemble. Nous reviendrons dans la section 4.3.4 sur le calcul de tels distances et angles.

En plus de ces deux tableaux, un troisième tableau contient des informations relatives à la base de données d'Helix Explorer. Il contient notamment des informations statistiques, tels que le nombre de structures secondaires dans la base, le nombre d'entrées PDB chargées etc. Dans l'annexe 4 sont présentées les structures des tableaux des structures secondaires et celui des voisins ainsi que le code SQL utilisé dans leur construction.

4.3.2 Interface Web d'Helix Explorer

Helix Explorer utilise l'interface 'Common Gateway Interface' (CGI) [56] pour traiter les requêtes dynamiques reçues par le serveur Web. CGI est la première des interfaces standards qui ont été implémentées par les serveurs Web pour transmettre les données provenant d'un client Web (i.e. le fureteur) vers une application sur le serveur et utiliser par la suite la même connexion pour diriger la réponse de l'application au client. En

comparaison avec d'autres approches qui ont été développées depuis pour assumer la même fonction dans les serveurs Web/serveurs d'applications (i.e. utilisant les Servlets [57], PHP [58], pages JSP [59] ou ASP [60]), l'interface CGI offre l'avantage d'être simple d'utilisation et universellement supportée par les serveurs Web. Néanmoins, cette interface a le désavantage de ne pas s'ajuster (*scale*) facilement à une augmentation du nombre de requêtes, puisqu'en pratique, un processus[12] supplémentaire est démarré à chaque nouvelle requête du client. Si le processus démarré, comme dans le cas de l'implémentation réalisée dans Helix Explorer, nécessite en plus le démarrage de la machine virtuelle Java (puisque l'application démarrée a été écrite en langage Java), cela peut avoir comme conséquence une consommation excessive de ressources (en mémoire vive et en temps de calculs) de la machine serveur. L'utilisation des serveurs d'applications permet souvent d'éviter ce type de problèmes puisque l'on peut en général avec un serveur d'applications spécifier le nombre d'instances de l'application/objets que l'on veut démarrer (ou définir la politique de démarrage des instances et/ou chargement d'objets dans le serveur). Chacune de ces instances/objets, sous conditions qu'elles soient conçues dans ce but, peut servir plusieurs clients à la fois. Nous avons néanmoins choisi dans l'implémentation d'Helix Explorer d'utiliser l'interface CGI pour sa simplicité et ce après avoir constaté que la responsivité de l'interface Web d'Helix Explorer est malgré tout acceptable.

D'autre part, l'interface Web d'Helix Explorer utilise elle-même l'interface JDBC de Java [61] pour communiquer avec la base de données d'Helix Explorer. L'interface JDBC est l'interface standard utilisée par les programmes Java pour accéder à une base de données relationnelle. Elle permet, entre autres, d'envoyer des requêtes SQL au serveur de la base de données, de récupérer les résultats de la requête et offre aussi les fonctions nécessaires pour parcourir (i.e. itérer dans) les données récupérées.

---

[12] Le terme processus doit être compris ici dans son sens informatique : une instance du programme qui est en cours d'exécution.

4.3.2.1 Génération des résultats des requêtes et utilisation de XSLT

Lorsque l'interface Web d'Helix Explorer reçoit une requête suite à une recherche explicite effectuée par l'utilisateur ou simplement parce que l'utilisateur a cliqué sur un lien dans une des pages de l'interface utilisateur, cette dernière construit la requête SQL correspondante et l'envoie au serveur de base de données. Le résultat retourné par la base de données est ensuite formaté pour être transformé en page HTML qui elle peut être visualisée par le client à travers le fureteur. L'ensemble de ce processus de formatage est schématisé dans la Figure 4.8 :

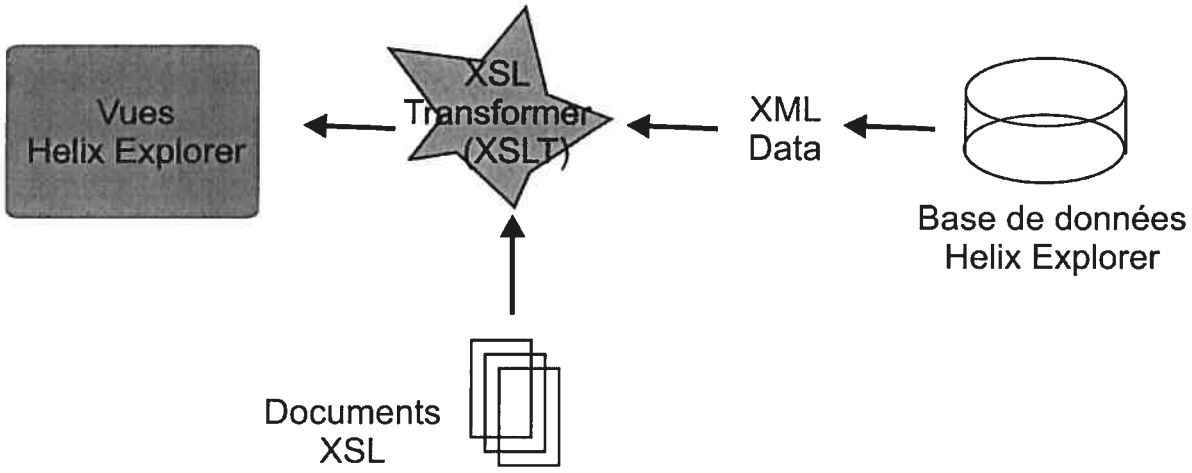

Fig.4.8 Générations des vues d'Helix Explorer. À la réception d'une nouvelle requête de l'utilisateur, Helix Explorer interroge la base de données, dont il formate les résultats en XML avant de faire appel à un transformateur XSLT. Ce dernier utilise les documents XSL pour construire la vue d'Helix Explorer correspondant à la requête.

En se basant sur le processus décrit dans cette figure, il est possible de distinguer dans l'interface Web d'Helix Explorer deux entités bien distinctes : la première est responsable de la génération de l'interface utilisateur (c'est-à-dire la génération de toutes les pages HTML). La deuxième entité implémente toute la 'logique applicative', c'est elle qui prend en charge la requête de l'utilisateur à son arrivée et construit la requête SQL qu'elle envoie

ensuite à la base de données et qui transmet les résultats de cette dernière requête sous le format XML à la première entité. Du point de vue des composants/objets présents dans l'interface Web d'Helix Explorer, l'implémentation suit un modèle similaire au modèle 'Model-View-Controller' (MVC).[62] Le renforcement de la séparation entre la 'présentation' et la 'logique' de l'application qui en résulte a principalement comme avantage de faciliter la maintenance de l'ensemble de l'application et aussi de rendre plus direct le rajout de nouvelles fonctionnalités à Helix Explorer et la mise à jour de l'interface utilisateur sans introduction de modifications aux composants qui n'ont pas réellement besoin d'être modifiés. L'interface utilisateur, qui définit la structure des pages HTML retournées à l'utilisateur, c'est-à-dire l'emplacement des éléments tels que les tableaux, titres, liens etc. dans ces pages, est entièrement décrite par l'intermédiaire de documents de type XSL (*eXtensible Stylesheet Language*) [63]. Ces documents sont 'transformés' (par XSLT) dynamiquement (ou 'sur demande') dans un processus où des données XML représentant les informations de la base de données d'Helix Explorer leurs sont intégrées. En d'autres termes, les documents XSL définissent le format de l'interface utilisateur alors que des données XML construites par l'interface Web d'Helix Explorer décrivent elles les informations extraites de la base de données d'Helix Explorer. En pratique, à chaque vue d'Helix Explorer correspond un fichier XSL. Ainsi, pour modifier l'interface utilisateur il suffit d'éditer un ou plusieurs fichier(s) XSL (qui sont de simples fichiers 'texte'). Aucune recompilation n'est nécessaire puisque les pages HTML sont générées dynamiquement en partant directement des fichiers XSL.

4.3.2.2 Interface Web dynamique et utilisation des scripts clients

Certaines pages Web de l'interface utilisateur d'Helix Explorer (voir la section 4.2) ne sont pas complètement statiques[13]. Elles fournissent à l'utilisateur un certain degré d'interactivité même si, certes, cette interactivité reste limitée. Cette dernière est

[13] 'Statiques' signifie qu'elles incluent peu ou pas d'éléments dynamiques et ce indépendamment de la nature (dynamique ou statique) de la génération de ces pages.

implémentée à l'aide de code exécutable sur le fureteur même du client : des procédures écrites en langage JavaScript ont été intégrées aux pages Web au côté des données de la base de données d'Helix Explorer qui sont elles générées dynamiquement. Concrètement, ce code qui s'exécute sur la machine du client implémente la fonction de filtrage décrite dans la section 4.2. Il aurait été bien évidemment possible d'implémenter cette fonction sur le serveur. Cependant, cela aurait eu comme conséquence de limiter la responsivité de l'interface utilisateur d'Helix Explorer et de surcharger le serveur. En effet, chaque mise à jour de la liste de paramètres définis dans la vue des voisins (voir section 4.2) aurait exigé alors l'envoi d'une nouvelle requête au serveur, ce qui aurait signifié des délais d'attente plus grands pour l'utilisateur et une augmentation du nombre de requêtes reçues par le serveur.

4.3.2.3 Implémentation des expressions régulières

Helix Explorer définit certains symboles pour décrire un groupe particulier d'acides aminés ayant une ou plusieurs caractéristiques en commun (voir sections 1.2 et 4.2). Le tableau suivant donne pour chacune des expressions définies par Helix Explorer (colonne 1) la liste d'acides aminés auxquels le symbole peut se substituer (colonne 2). Lorsque l'interface Web d'Helix Explorer reçoit une requête comprenant un des symboles de la première colonne du tableau 1, elle émet une requête au serveur de base de données où le symbole en question a été substitué par l'expression régulière donnée dans la colonne 3 du tableau. Ainsi, Helix Explorer ne fournit pas sa propre 'implémentation' des expressions régulières mais dépend pour cela de l'implémentation intégrée au serveur de base de données MySQL. [64]

Symbole	Acides aminés	Expression régulière
($)	Tous les acides aminés	[A-Z][14]
(N)	Non polaires : Ala, Cys, Gly, Ile, Leu, Met, Phe, Pro, Trp, Val	[ACGILMFPWV]
(P)	Polaires non chargés: Tyr, Gln, Ser, Thr, Asn	[YQSTN]
(A)	Acides : Asp, Glu	[DE]
(B)	Basiques : Arg, Lys, His	[RKH]

Tableau 1. Symboles utilisés dans Helix Explorer et leurs significations. Expressions régulières dans Helix Explorer. Pour chaque symbole défini par Helix Explorer (colonne 1) sont donnés, la liste des acides aminés auxquels le symbole se substitue (colonne 2) ainsi que l'expression régulière construite en réponse émise par Helix Explorer (colonne 3).

4.3.3 Constructeur de la base de données d'Helix Explorer

Le constructeur de la base de données d'Helix Explorer a pour fonction de créer et de mettre à jour les tableaux décrits dans la section 4.3.1. Cette tâche est effectuée principalement en deux étapes :

(1) Création des tableaux des structures secondaires et des 'voisins' et leur peuplement par les données de structures de la base de données PDBase. Cette étape a besoin uniquement des données qui sont soit des informations déjà présentes dans PDBase, soit des données qui peuvent en être déduites avec relativement peu de calculs[15]. Dans le cas du tableau des voisins, toute paire de structures secondaires appartenant à la même entrée PDB est représentée par une et une seule ligne dans le tableau des voisins. Dans le cas hypothétique d'une entrée PDB contenant uniquement 3 hélices, H1, H2 et H3, par

[14] Bien que les caractères de l'alphabet ne représentent pas tous des acides aminés, cette expression régulière est produite néanmoins pour simplification.

[15] En pratique, de telles données ne sont pas 'dérivées' mais sont les résultats de simples opérations sur les données PDB. Nous concédons néanmoins que la distinction entre ces deux types de données est subjective.

exemple, trois paires seront définies dans le tableau des voisins : (H1, H2), (H1, H3) et (H2, H3). Plus généralement, une structure PDB contenant *n* structures secondaires va contribuer de C_n^2, c'est-à-dire n(n-1)/2, lignes dans le tableau de voisins, chacune des lignes correspondant à une combinaison de deux structures secondaires distinctes de la structure initiale. Il est important de noter ici que, même si le nombre de paires croît quadratiquement avec le nombre de structures secondaires à l'intérieur d'une même structure PDB, la taille du tableau des voisins croît elle linéairement avec le nombre de PDB couverts par la base de données d'Helix Explorer. Cette propriété est importante pour garantir l'extensibilité de la base de données d'Helix Explorer lors de la mise à jour avec les structures PDB nouvellement résolues.

(2) Utilisation des données de PDB, ou en l'occurrence de PDBase, pour obtenir des données dérivées et les stocker dans la base de données d'Helix Explorer. Par 'données dérivées', nous entendons les informations qui ne se trouvent pas dans la base de données PDBase mais qui peuvent être obtenues en exploitant les informations brutes qui s'y trouvent. Des exemples de données dérivées incluent, par exemple, les coordonnées qui définissent l'axe d'une hélice ou la distance d'une structure secondaire à la surface d'une molécule. Pour calculer néanmoins cette dernière valeur, il est nécessaire de disposer d'une représentation de la surface moléculaire de la protéine à laquelle la structure secondaire appartient (voir section 3.4). Helix Explorer obtient les surfaces moléculaires de toutes les entrées de la base PDB en utilisant le programme '*dms*' [65] et les conserve par la suite dans une archives où chaque surface moléculaire est associée à un fichier unique contenant les coordonnées géométriques d'un nombre suffisant (dans bien des cas, des millions) de points qui appartiennent à la surface.

4.3.4 Calcul des distances, angles et autres données dérivées

Dans Helix Explorer, une structure secondaire d'une protéine est représentée par une liste ordonnée d'éléments décrivant chacun un acide aminé de la structure secondaire par (1) son type, c'est-à-dire de quel acide aminé il est question et (2) les coordonnées

géométriques de son carbone α[16]. Dans cette section nous décrivons comment à partir de ces informations, nous avons dérivé certaines données qui ont été par la suite intégrées à la base de données d'Helix Explorer.

4.3.4.1 Implémentation des distances dans Helix Explorer

Distances entre structures secondaires :

L'implémentation des trois types de distances ($D_{\min}$, D_{ctr} et D_{moy}) entre deux structures secondaires dans Helix Explorer est directement accomplie en se basant sur les formules de la section 4.1.2.1.

Distance à la surface moléculaire :

Le calcul des surfaces moléculaires en utilisant *dms* permet la construction d'une archives avec toutes les surfaces moléculaires des entrées PDB (voir section 4.3.3). Chacune de ces surfaces moléculaires est décrite par un ensemble de points répartis sur la surface elle-même selon une densité qu'on a préalablement spécifié à *dms* (5 points par Angström carré en l'occurrence). La surface moléculaire est, dès lors, perçue par Helix Explorer comme une structure secondaire particulière dont les carbones α coïncident avec l'échantillon de points calculé par *dms*. Cette représentation de la surface moléculaire comme structure secondaire rend possible l'utilisation par Helix Explorer de l'implémentation existante de D_{moy} pour calculer les distances de toutes les structures secondaires d'une protéine à sa surface moléculaire.

---

[16] C'est la seule information sur la structure de la protéine qui est utilisée par Helix Explorer. Les coordonnées des atomes de la chaîne latérale ne sont jamais considérées. Voir les limitations dans la section 4.5.

4.3.4.2 Recherche de l'axe d'une hélice

Plusieurs algorithmes ont été proposés pour trouver l'axe d'une hélice [66] lorsque l'on connaît les coordonnées d'un ensemble de points se trouvant sur la courbe de l'hélice (même si cette dernière n'est pas parfaite). Ces algorithmes sont intrinsèquement différents de ceux dont le but est d'identifier les hélices, ou plus généralement toutes les structures secondaires dans une protéine, à partir d'un ensemble de points représentants les carbones α. En effet, cette dernière catégorie d'algorithmes, dont une implémentation de référence est DSSP [67], contrairement aux algorithmes de [66] qui nous intéressent ici, n'ont aucune connaissance *à priori* du type de structures secondaires recherchées.

Tous les algorithmes couverts dans [66] acceptent comme entrée des points se trouvant sur l'hélice dont l'axe est recherché. Parmi ces algorithmes on peut néanmoins distinguer deux catégories : (a) les algorithmes qui acceptent seulement des points qui se trouvent être régulièrement répartis le long de l'hélice et (b) les autres algorithmes qui n'imposent pas une telle contrainte. Il est aussi à noter que ces algorithmes ne sont pas spécifiques au problème des structures hélicoïdales dans les protéines mais peuvent également être appliqués à n'importe quels types de problèmes où l'on recherche l'axe d'une hélice. Cependant, dans le cas de structures secondaires hélicoïdales des protéines, le fait que les carbones α sont équidistants (i.e. régulièrement espacés le long de l'hélice) signifie que l'on peut utiliser autant les algorithmes de la catégorie (a) que ceux de la catégorie (b). Le tableau 2 donne la liste des algorithmes décrits dans [66] et indique, pour chacun de ces algorithmes, la catégorie à laquelle ce dernier appartient. Dans Helix Explorer, nous avons choisi d'implémenter l'algorithme du *parametric least-square* (voir l'implémentation en Java donnée en annexe 5) plutôt que les autres algorithmes pour sa simplicité et en raison du fait que cet algorithme, lorsqu'il est utilisé pour trouver l'axe d'une hélice α (c'est le cas de la plupart des hélices de PDB), performe relativement bien comparé aux autres algorithmes (voir l'étude comparative réalisée dans [66]). Nous décrivons dans la suite de cette section cet algorithme tel qu'implémenté par Helix Explorer.

Algorithme	Les points considérés ont besoin d'être régulièrement distribués - catégorie (a)
parametric least squares	Oui
Moment matrix (eigenfit)	Non
Cylindrical fit (́qvist)	Non
Cross product of triad bisectors (Kahn)	Oui
Rotational least square (rotfit)	Oui

Tableau 2. Algorithmes pour le calcul de l'axe d'une hélice.

Parametric least square :

Dans cette méthode, l'axe de l'hélice est représenté par une équation paramétrée du type :

$$r = p + t.d \qquad (\mathrm{I})$$

où d est un vecteur le long de l'axe, p est un vecteur de position et t est un paramètre. Cette équation, projetée sur chacun des trois axes de coordonnées cartésiens, donne trois équations de droites auxquels on peut faire correspondre les points en utilisant la technique classique du moindre-carré. La résolution de ces trois équations donne les formules suivantes pour $d=(dx,dy,dz)$ et $p=(px,py,pz)$:

$$dx = \begin{vmatrix} \sum_i t_i r_{x,i} & \sum_i t_i \\ \sum_i r_{x,i} & n \end{vmatrix} \div D \qquad (\mathrm{II})$$

$$px = \begin{vmatrix} \sum_i t_i^2 & \sum_i t_i r_{x,i} \\ \sum_i t_i & \sum_i r_{x,i} \end{vmatrix} \div D \qquad (\mathrm{III})$$

$$\text{où} \qquad D = \begin{vmatrix} \sum_i t_i^2 & \sum_i t_i \\ \sum_i t_i & n \end{vmatrix} \qquad \text{(IV)}$$

dy et dz sont obtenus à partir de formules similaires.

4.4 Quelques statistiques obtenues avec Helix Explorer

Comme mentionné dans la section 4.3.1, la base de données d'Helix Explorer est principalement structurée autour de deux tableaux que sont le tableau des structures secondaires et le tableau des voisins (ou des paires). À partir de ces deux tableaux, il est possible d'extraire des informations très générales à la base de données PDB ou plus spécifiques aux structures secondaires qui s'y trouvent. Le tableau 3 rapporte certains paramètres généraux relatifs à la base de données PDB et des structures secondaires qui s'y trouvent.

Paramètre d'Helix Explorer	***Valeur***
Nombre d'entrées PDB	31266
Nombre de structures secondaires	651561
Nombre de paires de structure secondaire	4865347
Moyenne des longueurs des hélices (en acides aminés)	10,69

Tableau 3. Quelques paramètres de la base de données d'Helix Explorer. À cette date, seules les hélices sont incluses dans la base de données d'Helix Explorer.

D'autres informations extraites de la base de données d'Helix Explorer peuvent être considérées d'ordre 'plus statistique'. C'est le cas par exemple du nombre de paires de structures secondaires qui sont distantes l'une de l'autre d'au plus une valeur seuil donnée ou encore de la distribution sur toutes les structures de la base PDB des distances des

hélices à la surface moléculaire de leurs protéines (voir Figure 4.9). Pour obtenir de telles statistiques, il suffit, en général, d'émettre une requête SQL à la base de données d'Helix Explorer.

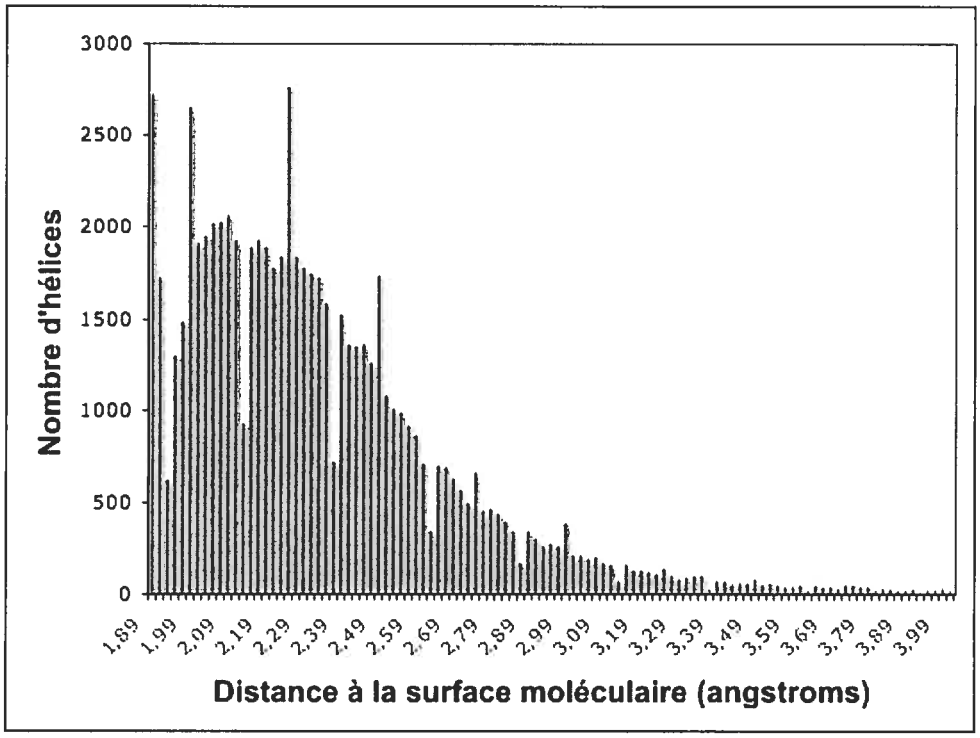

Fig.4.9 Distribution des distances des hélices à la surface moléculaire. Résultat obtenu pour un échantillon de 71069 hélices dans 4667 entrées PDB aléatoirement sélectionnées.

Dans la Figure 4.9, il est possible de distinguer un premier pic qui représente l'ensemble des hélices qui se trouvent à 'la surface' de leurs molécules. En effet, les hélices qui en font partie se trouvent à une distance de 1.9 angströms de la surface moléculaire de leurs protéines respectives. Cette distance, qui est la distance moyenne des carbones α à la surface moléculaire, coïncide avec la valeur du rayon Van der Waal d'un atome de carbone tel que considéré par *dms*. Cela signifie que tous les carbones α de ces hélices se trouvent en contact avec la surface moléculaire et donc sont soit totalement ou partiellement exposés au milieu extérieur de la molécule. À droite du premier pic de cette figure se trouve un 'creux' qui peut être pour sa part, interprété par le fait que dès que l'un des carbone α d'une

hélice se trouve à l'intérieur du volume décrit par la surface moléculaire, la moyenne des distances des carbones α à la surface moléculaire est considérablement augmentée. En effet, la présence d'autres atomes (voir d'acides aminés ou de structures secondaires entières) entre l'hélice et la surface moléculaire a pour effet d'augmenter cette distance d'au moins une valeur minimale - qui peut être vue en quelque sorte comme une valeur '*bottleneck*' – correspondant aux dimensions de ces atomes. Ceci est d'autant plus vrai que, dans la plupart des cas, la nature rigide d'une hélice fait que lorsqu'un carbone α d'une hélice donnée est enfoui à l'intérieur de la protéine, certains des carbones α qui lui sont voisins, et qui sont sur la même hélice, se trouvent eux aussi enfouis et contribuent par conséquent à l'augmentation de la distance de l'hélice à la surface de la molécule. Cette explication peut même être étendue à l'ensemble de la distribution rapportée de la Figure 4.9. En effet, la présence de la majorité des hélices sur des 'plages discrètes' centrées autour des pics dans cette figure peut être grossièrement expliquée par la nature discrète du nombre 'd'obstacles' séparant une hélice de la surface moléculaire. Le deuxième pic regrouperait ainsi les hélices séparées de la surface moléculaire d'un seul obstacle alors que le troisième regrouperait celles séparées par deux obstacles. D'autre part, nous considérons dans Helix Explorer qu'une hélice est 'de surface' si sa distance à la surface moléculaire est inférieure à 1.95Angströms. Cette valeur est choisie aléatoirement au pied du premier pic de la distribution (voir section 4.2).

Un autre exemple de données statistiques pour lesquelles une interprétation directe existe est celui donné par la Figure 4.10 où a été représenté la 'moyenne des distances moyennes' des hélices de la base PDB - d'un échantillon de cette dernière pour être plus précis - à la surface moléculaire de leurs protéines respectives comme fonction de la longueur de ces hélices. Il apparaît clairement de cette figure que plus une hélice est longue, plus la distance de cette dernière à la surface moléculaire est grande. En effet, plus une hélice est longue, plus elle est susceptible de contenir des acides aminés hydrophobes qui eux, comme mentionné dans le chapitre 1, ont naturellement tendance à se trouver enfouis à l'intérieur de la protéine. Cela a comme conséquence d'éloigner l'hélice de la surface moléculaire.

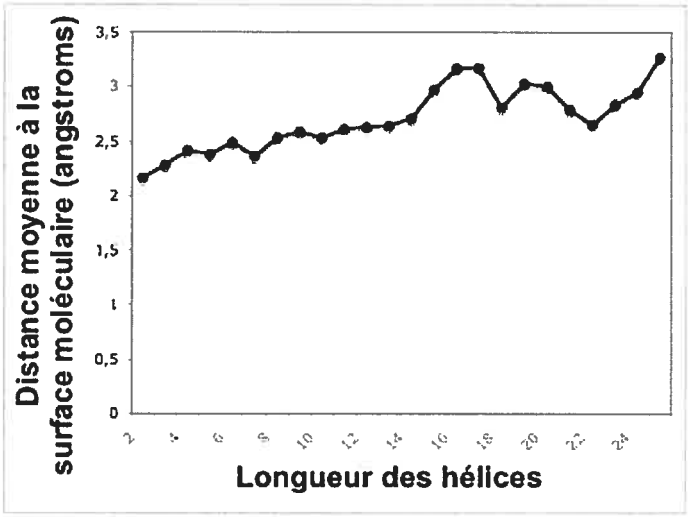

Fig.4.10 Distance d'hélices à la surface moléculaire comme fonction de leurs longueurs. Résultat obtenu pour un échantillon de 71069 hélices dans 4667 entrées PDB aléatoirement sélectionnées.

4.5 Limitations

Nous avons présenté le long de ce document un outil logiciel, Helix Explorer, qui est conçu principalement (mais pas exclusivement) pour être utilisé dans la conception de nouvelles protéines. Comme tout outil, Helix Explorer a ses limitations que nous évoquons dans cette section.

Une des hypothèses derrière l'implémentation d'Helix Explorer est que la structure de n'importe quelle protéine peut être perçue comme une collection des structures secondaires qui la composent. Cela est justifiable lorsque la protéine en question est riche en hélices et feuillets β. Mais si, au contraire, la structure est majoritairement constituée d'acides aminés qui ne se trouvent pas dans une structure secondaire 'régulière', la structure protéique en question risque de contribuer de façon très limitée à la base d'Helix Explorer. En d'autres termes, toutes les sous-séquences de la protéine qui ne font pas partie d'une hélice, d'un feuillet β ou d'un tour ne seront pas visibles à Helix Explorer et par conséquent à son utilisateur.

D'autre part, Helix Explorer ne tient compte à aucun moment du type des acides aminés présents dans les structures protéiques pour le calcul des métriques décrites dans la section 4.1.2. Ainsi, la distance entre deux structures secondaires - ceci est aussi vrai pour la distance d'une structure secondaire à la surface moléculaire - est mesurée 'seulement' entre les deux chaînes principales respectives des deux structures secondaires. Il s'en suit que la distance mesurée n'est qu'une représentation partielle de la distance réelle qui existe entre les deux structures secondaires puisque les chaînes latérales peuvent s'étendre bien au delà de la chaîne principale de la protéine. Certes, dans Helix Explorer, la connaissance des séquences primaires des deux structures secondaires peut donner des indications sur l'étendu de l'espace qu'elles occupent. On peut ainsi s'attendre à ce que les acides aminés avec les chaînes latérales les plus longues s'étendent plus loin de la chaîne principale que ceux avec une chaîne latérale courte. Cependant, il est évident que de telles déductions ne peuvent remplacer les conclusions plus directes que l'utilisateur pourrait faire s'il disposait de mesures de distances réelles entre structures secondaires.

Par ailleurs, si l'on souhaitait évaluer l'implémentation réalisée d'Helix Explorer, il est naturel de s'interroger tout d'abord sur les fonctionnalités que ce nouvel outil apporte à son utilisateur. Ce point a été couvert dans la section 4.2. Une autre question que l'on pourrait se poser dans le cas d'Helix Explorer, et qui n'est pas moins importante, est de savoir si une recherche effectuée par l'intermédiaire de l'interface utilisateur d'Helix Explorer retourne des résultats 'pertinents'. Cette question serait elle même sans doute pertinente dans le cas où la recherche en question se fait par 'similarité' de séquences et utilise des heuristiques, c'est-à-dire selon le même principe utilisé par BLAST pour la recherche dans la base de données des séquences génomiques de NCBI [6]. En effet, lorsqu'un algorithme heuristique est utilisé pour rechercher dans une base de données, quelle que soit cette dernière, le problème de faux positifs et faux négatifs est toujours présent. Dans Helix Explorer, la recherche peut être qualifiée de 'déterministe', dans le sens où la réponse retournée à la suite d'une requête couvre strictement les séquences des structures secondaires qui répondent à la totalité des critères de la recherche. Cependant, même si les problèmes de faux positifs/négatifs ne se posent pas dans le cas d'une recherche avec Helix Explorer,

d'autres problèmes liés à l'implémentation de la recherche se posent. Nous en citons deux ici :

(1) Lorsque la base de données d'Helix Explorer augmente de taille, suite à l'ajout de nouvelles structures dans la base PDB, la performance des requêtes à la base en pâtit malgré l'utilisation d'index dans la base MySQL.

(2) La possibilité de rechercher les séquences dans la base PDB en utilisant des expressions régulières, introduit un coût additionnel à chaque requête à la base de données d'Helix Explorer. En effet, la base de données MySQL ne peut utiliser les index pour accélérer ce type de recherches dans la base de données d'Helix Explorer. Elle utilise plutôt comme moyen d'optimisation, sous certaines conditions[17], l'algorithme *Turbo Boyer-Moore*. [68]

D'autre part, il est sans doute possible d'amener de grandes améliorations à l'interface utilisateur d'Helix Explorer en utilisant certaines des nouvelles technologies Web. En particulier, l'interface utilisateur peut être enrichie pour ressembler à une application de bureau (*desktop application*) en utilisant les techniques AJAX [69]. Cela aurait pour conséquence d'améliorer 'l'expérience' de l'utilisateur.

Il existe néanmoins dans Helix Explorer d'autres limitations auxquelles il est difficile de remédier. En particulier, une qui mérite d'être mentionnée, concerne l'intégrité des données dans la base de données d'Helix Explorer. Comme c'est le cas pour la grande majorité des bases de données en bioinformatique, il est important et même nécessaire d'avoir une curation manuelle. En effet, sans cette curation, certains registres de la base de données d'Helix Explorer peuvent être corrompus du fait de la corruption dans la base de données PDB ou pour une autre raison quelconque. De notre expérience, même si de tels

---

[17] Dans mySQL, les index sont utilisés lorsque l'expression recherchée ne débute pas par un caractère 'wild card'. Cependant dans l'implémentation d'Helix Explorer, les index ne sont pas utilisés puisque la séquence recherchée peut apparaître à l'intérieur d'une séquence de structure secondaire.

occurrences sont rares et leur détection peut souvent être automatisée, l'automatisation ne peut en fin de compte complètement se substituer à toute intervention humaine.

4.6 Développements présents et futurs

En plus des évolutions permettant de remédier aux limitations décrites dans la section 4.5, plusieurs développements ont été envisagés et entrepris dans Helix Explorer.

Une extension possible des fonctionnalités de l'interface utilisateur d'Helix Explorer serait le rajout d'un module permettant de visualiser deux structures secondaires voisines en faisant abstraction du reste de la structure PDB qui les renferme. En effet, la présence d'autres éléments de la protéine peut rendre l'analyse des interactions entre les deux structures plus difficile. Cela peut être réalisé ou bien par l'inclusion d'une simple image des deux structures adjacentes sans l'interférence créée par le reste de la structure protéique ou alors par l'intermédiaire d'une application qui s'exécute sur le fureteur de l'utilisateur, lui permettant ainsi de naviguer entre les deux éléments, de changer de perspective, de 'zoomer' ou d'effectuer d'autres opérations communes dans les logiciels de visualisation en 3D. Pour l'instant, l'outil qui devrait prochainement être accessible depuis la vue des voisins décrite dans la section 4.2, permet l'affichage et la navigation uniquement entre deux voisins qui sont tous deux des hélices. Ce module, implémenté par Alexandre Guerrini dans le but d'être intégré à l'interface utilisateur d'Helix Explorer, utilise la librairie Java3D [70] et, de ce fait, exige de la part de l'utilisateur d'Helix Explorer d'installer au préalable sur son fureteur une extension (*plugin*) Java3D qui se trouve rarement intégrée par défaut aux fureteurs actuels.

Dans un autre projet indépendant de celui décrit dans ce mémoire, mais qui utilise néanmoins la base de données d'Helix Explorer, nous nous intéressons aux motifs structuraux les plus courants dans la base PDB, c'est-à-dire les différentes configurations géométriques locales récurrentes qui sont formées par des structures secondaires. Étant donné le nombre important de structures secondaires qui existent dans la base PDB, et pour

éviter l'explosion combinatoire qui se poserait si l'on souhaitait identifier tous les motifs existants, ce problème peut être abordé de façon progressive en s'intéressant en premier temps uniquement à un type spécifique de structures secondaires, les hélices en l'occurrence, et en se limitant pour commencer aux motifs constitués de trois structures secondaires seulement. En d'autres termes, nous commençons par cataloguer les triplets d'hélices qui se trouvent dans un voisinage immédiat[18] les unes des autres. Chaque entrée du catalogue construit contient, en plus des séquences des trois hélices, toutes les métriques qui les relient. C'est-à-dire, en tout, pour chaque triplet, six paramètres sous forme de trois paires (distance, angle), chacune définissant l'angle et la distance entre les deux hélices de chaque paire du triplet. L'extraction d'information depuis le catalogue peut être effectuée par des techniques d'excavation de données (*datamining*) en commençant par l'application d'un algorithme de 'clustering' qui regroupe les triplets possédants des paramètres similaires dans le même groupe (*cluster*). Les groupes identifiés peuvent par la suite être affinés en des groupes encore plus petits par le regroupement des triplets qui partagent d'autres propriétés tel que leur caractère hydrophile/hydrophobe ou ceux qui se trouvent à des distances similaires de la surface moléculaire de leurs protéines respectives. Les groupes obtenus après cette étape peuvent alors être étudiés plus en détails pour identifier éventuellement, si elle existe, une fonction commune attribuée au motif. L'objectif ultime étant de pouvoir inférer, en utilisant un algorithme de classification et en partant uniquement de la structure d'une protéine donnée, la fonction ou une des fonctions de la protéine en identifiant les motifs qui s'y trouvent. Ce dernier projet, qui pourrait aussi conduire à la réalisation d'une classification des protéines selon les domaines qu'elles renferment, c'est-à-dire de manière similaire à la classification faite par CATH [22] ou SCOP [23], a été récemment entrepris par Vincent Normandeau-Babin et nous regardons en avant pour les résultats qui vont en découler.

---

[18] C'est-à-dire se trouvant à une distance qui n'excède pas une valeur seuil l'une de l'autre.

Conclusion

Dans ce mémoire, nous avons introduit Helix Explorer qui est un nouvel outil bioinformatique permettant la recherche dans l'ensemble des structures se trouvant dans la base de structures de protéines PDB. En plus de rendre possible la recherche de structures secondaires dans la base de données PDB par séquences peptidiques ou par extension en utilisant des expressions régulières, Helix Explorer permet 'l'exploration' des structures protéiques, non pas individuellement comme le permettent plusieurs autres outils informatiques, mais en s'appuyant sur l'ensemble des données de la base de données PDB. Cela rend possible les études comparatives de structures préalablement résolues. Certes, nous avons présenté dans ce qui précède Helix Explorer principalement comme un outil destiné aux concepteurs de nouvelles protéines, nous le percevons toutefois, plus généralement, comme un outil pour l'analyse comparative des structures protéiques. En particulier, pour l'analyse des sous-structures protéiques occupant un espace relativement limité et qui sont formées essentiellement de structures secondaires régulières.

Nous avons également présenté en détails dans ce mémoire l'architecture logicielle et l'implémentation ainsi que certaines limitations d'Helix Explorer et avons proposé des voies pour le faire évoluer.

Pour conclure enfin, nous invitons le lecteur à visiter l'adresse Web à laquelle Helix Explorer se trouve http://www-lbit.iro.umontreal.ca/mms/he.htm en utilisant n'importe quel fureteur et de consulter sur ce site les exemples et la documentation supplémentaire qui y sont publiées afin d'évaluer soi même les fonctionnalités offerte par ce nouvel outil.

Bibliographie

[1] Lander,E.S., Linton,L.M., Birren,B., Nusbaum,C., Zody,M.C., Baldwin,J., Devon,K., Dewar,K., Doyle,M. and FitzHugh,W. et al., Initial sequencing and analysis of the human genome. (2001) Nature, 409, 860-921.

[2] Murthy,V.L. and Rose,G.D., RNABase: An annotated database of RNA structures. (2003) Nucleic Acids Res., 31, 502–504.

[3] Bernstein,F.C., Koetzle,T.F., Williams,G.J., Meyer,E.E. Jr., Brice,M.D. and Rodgers,J.R. et al., The Protein Data Bank: a computer-based archival for macromolecular structures. (1977) J. Mol.Biol., 112, 535-542.

[4] Heinemeyer,T., Wingender,E., Reuter,I., Hermjakob,H., Kel,A.E., Kel,O.V., Ananko,E.A., Podkolodnaya,O.A., Kolpakov,F.A., Podkolodny,N.L. and Kolchanov,N.A., Databases on Transcriptional Regulation: TRANSFAC, TRRD, and COMPEL. (1998) Nucleic Acids Res., 26, 362-367.

[5] Schomburg,I., Chang,A., and Schomburg,D., BRENDA, enzyme data and metabolic information. (2002) Nucleic Acids Res., 30, 47-49.

[6] NCBI, http://www.ncbi.nlm.nih.gov/

[7] Kaplan,W. and Littlejohn,T., Swiss-PDB Viewer (Deep View). (2001) Brief Bioinform., 2, 195-197.

[8] Achard,F., Vaysseix,G. and Barillot,E., XML, bioinformatics and data integration. (2001) Bioinformatics, 17, 115–125.

[9] Hucka,M., Finney,A., Sauro,H.M., Bolouri.,H., Doyle,J.C., Kitano,H., Arkin,A.P., Bornstein,B.J., Bray,D. and Cornish-Bowden,A. et al., The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. (2003) Bioinformatics, 19, 524-531.

[10] Bioinformatics Web (BIW), Bioinformatics - Origin and History, http://www.geocities.com/bioinformaticsweb/his.html

[11] Bourne,P.E. and Weissig,H., Structural Bioinformatics. Wiley, New York, 2003.

[12] Sussman,J.L., Lin,D., Jiang,J., Manning,N.O., Prilusky,J., Ritter,O. and Abola,E.E., Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules. (1998) Acta Crystallogr., 54, 1078-1084.

[13] Drews,J., Drug Discovery: A Historical Perspective. (2000) Science, 287, 1960-1964.

[14] Whisstock,J. and Lesk,A., Prediction of protein function from protein sequence and structure. (2003) Q. Rev. Biophys., 36, 307-340.

[15] Pokala,N. and Handel,T.M., Review: Protein design – where we were, where we are, where we're going. (2001) J. Struct. Biol., 134, 369-281.

[16] Orengo,C.A., Todd,A.E. and Thornton,J.M., From protein structure to function. (1999) Curr. Opin. Struct. Biol., 9, 374-82.

[17] Drenth,J., Principles of Protein X-Ray Crystallography. Springer-Verlag Inc, 1999.

[18] Wuthrich,K., NMR of Proteins and Nucleic Acids. Wiley, New York, 1986.

[19] Karplus,M. and Mc Cammon,J.A., Molecular dynamics simulations of biomolecules. (2002) Nat Struct Biol, 9, 646-652.

[20] Murphy,R.B., Philipp,D.M. and Friesner,R.A., A mixed quantum mechanics/molecular mechanics (QM/MM) method for large-scale modeling of chemistry in protein environments. (2000) J. Comput. Chem., 21, 1442-1457.

[21] Desjarlais,J.R. and Handel,T.M., New strategies in protein design. (1995) Curr. Opin. in Biotech., 6 (4) , 460-466.

[22] Pearl,F.M.G., Bennett,C.F., Bray,J.E., Harrison,A.P., Martin,N., Shepherd,A., Sillitoe,I., Thornton,J. and Orengo,C.A., The CATH database: an extended protein family resource for structural and functional genomics. (2003) Nucleic Acids Res., 31(1), 452–455.

[23] Murzin,A.G., Brenner,S.E., Hubbard,T. and Chothia,C., SCOP: a structural classification of proteins database for the investigation of sequences and structures. (1995) J. Mol. Biol., 247, 536-540.

[24] Biochimie des protéines, Université de Sherbrooke, http://www.callisto.si.usherb.ca/~bcm514/1a.html

[25] Pearson Education, Inc., publishing as Benjamin Cummings, 2004.

[26] Milner-White,E.J., Situations of Gamma-turns in Proteins: Their Relation to Alpha-helices, Beta-sheets and Ligand Binding Sites, (1990) J. Mol. Biol., 216, 385-397.

[27] Pavone,V., Gaeta,G., Lombardi,A., Nastri,F., Maglio,O., Isernia,C., and Saviano,M., Discovering Protein Secondary Structures: Classification and Description of Isolated α-Turns. (1996) Biopolymers, 38, 705-721.

[28] Rajashankar,K.R. and Ramakumar,S., π-Turns in proteins and peptides: Classification, conformation, occurrence, hydration and sequence. (1996) Protein Sci., 5, 932-946.

[29] Protein Structure, http://oregonstate.edu/instruction/bb450/lecturenoteskevin/proteinstructureIoutline.html

[30] Wilmot,C.M. and Thornton,J.M., Analysis and Prediction of the Different Types of beta-Turn in Proteins. (1988) J. Mol. Biol., 203, 221-232.

[31] Greer,J. and Bush.,B., Macromolecular shape and surface maps by solvent exclusion. (1978) Proc. Natl. Acad. Sci. USA, 75, 303-307.

[32] Connolly,M.L., Molecular Surfaces: A review, http://www.netsci.org/Science/Compchem/feature14.html

[33] Perl,D., Mueller,U., Heinemann,U. and Schmid,F.X., Two exposed amino acid residues confer thermostability on a cold shock protein. (2000) Nature Structural Biology, 7, 380-383.

[34] Chen,K. and Arnold,F.H., Enzyme Engineering for Nonaqueous Solvents: Random Mutagenesis to Enhance Activity of Subtilisin E in Polar Organic Media. (1991) Bio/Technology, 9, 1073-1077.

[35] Chirumamilla,R.R., Muralidhar,R., Marchant,R. and Nigamarnold,P., Improving the quality of industrially important enzymes by directed evolution. (2001)

Mol. Cell. Biochem., 224, 159–168.

[36] Bornscheuer,U.T. and Pohl,M., Improved biocatalysts by directed evolution and rational protein design. (2001) Curr. Opin. Chem. Biol., 5, 137-143

[37] Skerra,A. Engineered protein scaffolds for molecular recognition: Review. (2000) J. Mol. Recognit., 13, 167-187.

[38] Liu,Y. and Kuhlman,B., RosettaDesign server for protein design. (2006) Nucleic Acids Res., Jul 1, 34.

[39] Kuhlman,B., Dantas,G., Ireton,G.C., Varani,G., Stoddard,B.L. and Baker,D., Design of a Novel Globular Protein Fold with Atomic-Level Accuracy. (2003) Science, Vol. 302. 5649, 1364-1368.

[40] Leach,A.R., Molecular Modeling: Principles and Applications. Second edition. Pearson Education EMA, 2001.

[41] Richardson Laboratory, http://kinemage.biochem.duke.edu/lab/Richardson/richardson.php

[42] Minor,D.L. Jr. and Kim,P.S., Context-dependent secondary structure formation of a designed protein sequence. (1996) Nature, 380, 730-734.

[43] Van den Heuvel,M., An introduction to artificial proteins. (2002) Chapter 1. Ph. D. thesis, University of Groningen Library. http://www.ub.rug.nl/

[44] Westbrook,J. and Bourne,P.E., STAR/mmCIF: An extensive ontology for macromolecular structure and beyond. (2000) Bioinformatics, 16, 159–168.

[45] Bourne,P., Berman,H.M., Watenpaugh,K., Westbrook,J. and Fitzgerald,P.M.D., The macromolecular Crystallographic Information File (mmCIF). (1997) Meth. Enzymol., 277, 571-590.

[46] Westbrook,J., Feng,Z., Jain,S., Bhat,T.N., Thanki,N., Ravichandran,V., Gilliland,G.L., Bluhm,W., Weissig,H., Greer,D.S., Bourne,P.E. and Berman,H.M., The Protein Data Bank: unifying the archive. (2002) Nucleic Acids Res., 30, 245–248.

[47] Westbrook,J., Ito,N., Nakamura,H., Henrick,K., and Berman,H.M., PDBML: The representation of archival macromolecular structure data in XML. (2005)

Bioinformatics, 21, 988–992.

[48] Lovell et al., Structure validation by C-alpha geometry: phi, psi, and C-beta deviation. (2003) Proteins, 50, 437-450.

[49] Laskowski,R.A., MacArthur,M.W., Moss,D.S. and Thornton,J.M., PROCHECK: a program to check the stereochemical quality of protein structures. (1993) J. Appl. Cryst., 26, 283-291.

[50] Vaguine,A.A., Richelle,J. and Wodak.,S.J., SFCHECK: a unified set of procedures for evaluating the quality of macromolecular structure-factor data and their agreement with the atomic model. (1999) Acta Cryst., D55, 191-205.

[51] Vinoski,S., CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. (1997) IEEE Communications Magazine, Vol. 14, No. 2.

[52] Greer,D.S., Westbrook,J.D. and Bourne,P.E., An ontology driven architecture for derived representations of macromolecular structure. (2002) Bioinformatics, 18, 1280-1281.

[53] Rawlings,C.J., Taylor,W.R., Nyakairu,J., Fox,J. and Sternberg,M.J.E., Using Prolog to Represent and Reason about Protein Structure. (1986) ICLP, 536-543.

[54] Turcotte,M., Muggleton,S. and Sternberg,M.J.E., Use of Inductive Logic Programming to Learn Principles of Protein Structure. (2000) Electron. Trans. Artif. Intell., 4(B), 119-124.

[55] ECMA-262, http://www.ecma-international.org/news/index.html

[56] Gundavaram,S., CGI Programming on the World Wide Web. O'Reilly Associates, Sebastopol, CA, 1996.

[57] Java Servlet, Sun Microsystems, http://java.sun.com/products/servlet/

[58] PHP, http://www.php.net/

[59] Java Server Pages, Sun Microsystems, http://java.sun.com/products/jsp/

[60] ASP.NET, Microsoft http://www.asp.net/

[61] Java Database Connectivity, http://java.sun.com/javase/technologies/database.jsp

[62] Buschmann,F., Meunier,R., Rohnert,H., Sommerlad,P. and Stal,M., Pattern-Oriented Software Architecture. John Wiley and Sons, 1996.

[63] XML Stylesheet Language, http://www.w3.org/Style/XSL/

[64] MySQL AB, http://www.mysql.com/products/database/

[65] Ferrin,T.E., Huang,C.C., Jarvis,L.E. and Langridge,R., The MIDAS display system. (1988) Journal of Molecular Graphics archive, Volume 6, Issue 1, 13-17.

[66] Christopher,J.A., Swanson,R. and Baldwin,T.O., Algorithms for finding the axis of a helix: fast rotational and parametric least-squares methods. (1996) Comput. Chem., 20(3), 339-45.

[67] Kabsch,W. and Sander,C., Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. (1983) Biopolymers, 22(12), 2577-637.

[68] Baker,B.S., Parameterized pattern matching by Boyer-Moore-type algorithms. (1995) Symposium on Discrete Algorithms archive. Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms, 541-550.

[69] Garrett,J.J., Ajax: A New Approach to Web Applications, http://www.adaptivepath.com/publications/essays/archives/000385.php (February 18, 2005).

[70] Java 3D, http://java.sun.com/products/java-media/3D/

[71] Moore,J.C., Jin,H.M., Kuchner,O. and Arnold,F.H., Strategies for the in vitro evolution of protein function: enzyme evolution by random recombination of improved sequences. (1997) J. Mol. Biol., 272, 336–347.

[72] Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J., Basic Local Alignement Search Tool. (1990) J. Mol. Biol., 215, 403-410.

Annexe 1 : Exemple de Fichier '.pdb'

```
HEADER    HYDROLASE(O-GLYCOSYL)                    29-SEP-92   102L      102L   2
COMPND    LYSOZYME INSERTION MUTANT WITH ALA INSERTED AFTER ASN 40,     102L   3
COMPND   2 CYS 54 REPLACED BY THR, CYS 97 REPLACED BY ALA (INS(N40-A),  102L   4
COMPND   3 C54T,C97A)                                                   102L   5
SOURCE    BACTERIOPHAGE T4 (MUTANT GENE DERIVED FROM THE M13            102L   6
SOURCE   2 PLASMID BY CLONING THE T4 LYSOZYME GENE)                     102L   7
AUTHOR    D.W.HEINZ,B.W.MATTHEWS                                        102L   8
REVDAT   2   15-JAN-95 102LA   1       HET                             102LA  1
REVDAT   1   31-OCT-93 102L    0                                       102L   9
JRNL        AUTH   D.W.HEINZ,W.A.BAASE,F.W.DAHLQUIST,B.W.MATTHEWS       102L  10
JRNL        TITL   HOW AMINO-ACID INSERTIONS ARE ALLOWED IN AN          102L  11
JRNL        TITL 2 ALPHA-HELIX OF T4 LYSOZYME                           102L  12
JRNL        REF    NATURE                        V. 361   561 1993     102L  13
JRNL        REFN   ASTM NATUAS  UK ISSN 0028-0836                 006  102L  14
REMARK   1                                                              102L  15
REMARK   2                                                              102L  16
REMARK   2 RESOLUTION. 1.74 ANGSTROMS.                                  102L  17
REMARK   3                                                              102L  18
REMARK   3 REFINEMENT.                                                  102L  19
REMARK   3   PROGRAM                    TNT                             102L  20
REMARK   3   AUTHORS                    TRONRUD,TEN EYCK,MATTHEWS       102L  21
REMARK   3   R VALUE                    0.174                           102L  22
REMARK   3   RMSD BOND DISTANCES        0.014  ANGSTROMS                102L  23
REMARK   3   RMSD BOND ANGLES           2.0    DEGREES                  102L  24
...
REMARK   5                                                              102L  42
REMARK   5 THE ORTHOGONAL X,Y,Z AXES OF THIS COORDINATE SET ARE         102L  43
REMARK   5 ALIGNED IN THE A*,B,C CRYSTALLOGRAPHIC DIRECTIONS.           102L  44
REMARK   6                                                              102L  45
REMARK   6 RESIDUES 162 - 164 IN WILD-TYPE AND ALL MUTANT LYSOZYMES     102L  46
REMARK   6 ARE EXTREMELY MOBILE.  THUS THE COORDINATES FOR THESE        102L  47
REMARK   6 RESIDUES ARE VERY UNRELIABLE.  THIS ENTRY DOES NOT INCLUDE   102L  48
REMARK   6 RESIDUES 163 AND 164.                                        102L  49
REMARK   7                                                              102L  50
REMARK   7 THERE ARE SEVERAL SUBTLE ASPECTS OF THE SECONDARY STRUCTURE  102L  51
REMARK   7 OF THIS MOLECULE WHICH CANNOT CONVENIENTLY BE REPRESENTED    102L  52
REMARK   7 IN THE HELIX AND SHEET RECORDS BELOW.  THESE ASPECTS         102L  53
REMARK   7 INFLUENCE THE REPRESENTATION OF HELIX 6 AND STRAND 3 OF      102L  54
REMARK   7 SHEET *S1*.  THE PAPER J.MOL.BIOL., V. 118, P. 81, 1978     102L  55
REMARK   7 SHOULD BE CONSULTED FOR THESE SUBTLETIES.                    102L  56
REMARK   8                                                              102L  57
REMARK   8 SEO 901 FORMS AN S-S LINKAGE TO SEO 902.  THE DEPOSITORS     102L  58
REMARK   8 ASSIGNED THE RESIDUE NAME BME TO THE BETA-MERCAPTOETHANOL.   102L  59
REMARK   8 IN ORDER TO FOLLOW PROTEIN DATA BANK NOMENCLATURE, THE       102L  60
REMARK   8 THE MERCAPTOETHANOLS ARE PRESENTED AS *HET* GROUPS *SEO*.    102L  61
REMARK   9                                                              102LA  2
REMARK   9 CORRECTION. INSERT MISSING HET RECORD.  15-JAN-95.           102LA  3
...
CRYST1   60.900   60.900   96.100  90.00  90.00 120.00 P 32 2 1      6  102L  88
```

```
ORIGX1      1.000000  0.000000  0.000000        0.00000                      102L  89
ORIGX2      0.000000  1.000000  0.000000        0.00000                      102L  90
ORIGX3      0.000000  0.000000  1.000000        0.00000                      102L  91
SCALE1      0.016420  0.009480  0.000000        0.00000                      102L  92
SCALE2      0.000000  0.018961  0.000000        0.00000                      102L  93
SCALE3      0.000000  0.000000  0.010406        0.00000                      102L  94
ATOM      1  N   MET     1      44.061  -3.277   8.755  1.00 22.03           102L  95
ATOM      2  CA  MET     1      43.619  -1.924   8.869  1.00 14.21           102L  96
ATOM      3  C   MET     1      42.195  -1.991   9.394  1.00 18.00           102L  97
ATOM      4  O   MET     1      41.523  -2.983   9.193  1.00 16.68           102L  98
ATOM      5  CB  MET     1      43.727  -1.353   7.458  1.00 17.98           102L  99
ATOM      6  CG  MET     1      42.994  -0.069   7.291  1.00 31.76           102L 100
ATOM      7  SD  MET     1      44.005   1.333   7.794  1.00 35.36           102L 101
ATOM      8  CE  MET     1      45.515   0.985   6.912  1.00 39.74           102L 102
ATOM      9  N   ASN     2      41.782  -0.980  10.131  1.00 11.26           102L 103

...

HETATM 1299  OB1 SEO   901      33.365  -0.851  15.514  1.00 38.25    2      102L1393
HETATM 1300  CB1 SEO   901      31.965  -0.933  15.030  1.00 33.04    2      102L1394
HETATM 1319  O   HOH   183      34.192  11.289  28.922  1.00 39.30           102L1413
HETATM 1320  O   HOH   184      48.531   9.908  28.532  1.00 69.93           102L1414
HETATM 1321  O   HOH   185      31.118   5.925  23.394  1.00 30.98           102L1415
HETATM 1322  O   HOH   186      40.670   5.447  20.531  1.00 26.46           102L1416
HETATM 1323  O   HOH   187      44.851  -3.689  11.364  1.00 44.47           102L1417
HETATM 1324  O   HOH   188      39.211  22.653  -6.614  1.00 28.22           102L1418
HETATM 1325  O   HOH   189      50.725  17.496  19.609  1.00 39.08           102L1419
HETATM 1326  O   HOH   190      36.286  27.659  16.011  1.00 43.54           102L1420
HETATM 1327  O   HOH   191      37.325  29.870  14.316  1.00 54.86           102L1421
HETATM 1328  O   HOH   192      44.145  25.354   9.871  1.00 29.12           102L1422

...

CONECT 1302 1301 1306                                                       102L1538
CONECT 1303 1304                                                            102L1539
CONECT 1304 1303 1305                                                       102L1540
CONECT 1305 1304 1306                                                       102L1541
CONECT 1306 1302 1305                                                       102L1542
MASTER       49    7    4    0    0    0    0    6 1439    1    8   13      102LA  5
END                                                                         102L1544
```

Annexe 2 : Exemple de Fichier mmCIF

```
data_102L
#
loop_
_atom_sites_footnote.id
_atom_sites_footnote.text
1
;RESIDUES 162 - 164 IN WILD-TYPE AND ALL MUTANT LYSOZYMES ARE EXTREMELY MOBILE.
THUS THE COORDINATES FOR THESE RESIDUES ARE VERY UNRELIABLE.  THIS ENTRY DOES NOT
INCLUDE RESIDUES 163 AND 164.
;
2 'SEO 901 FORMS AN S-S LINKAGE TO SEO 902.'
#
loop_
_audit_author.name
'Heinz, D.W.'
'Matthews, B.W.'
…

#
_database_PDB_matrix.entry_id          102L
_database_PDB_matrix.origx[1][1]       1.000000
_database_PDB_matrix.origx[1][2]       0.000000
_database_PDB_matrix.origx[1][3]       0.000000
_database_PDB_matrix.origx[2][1]       0.000000
_database_PDB_matrix.origx[2][2]       1.000000
_database_PDB_matrix.origx[2][3]       0.000000
_database_PDB_matrix.origx[3][1]       0.000000
_database_PDB_matrix.origx[3][2]       0.000000
_database_PDB_matrix.origx[3][3]       1.000000
_database_PDB_matrix.origx_vector[1]   0.00000
_database_PDB_matrix.origx_vector[2]   0.00000
_database_PDB_matrix.origx_vector[3]   0.00000
…
…
#
loop_
_entity.id
_entity.type
_entity.src_method
_entity.pdbx_description
_entity.formula_weight
_entity.pdbx_number_of_molecules
_entity.details
1 polymer     man 'T4 LYSOZYME'     18702.619 1   ?
2 non-polymer syn 'CHLORIDE ION'    35.453    2   ?
3 non-polymer syn 2-MERCAPTOETHANOL 78.129    2   ?
4 water       nat water             18.015    132 ?
#
loop_
_entity_poly_seq.entity_id
_entity_poly_seq.num
_entity_poly_seq.mon_id
1 1   MET
1 2   ASN
1 3   ILE
```

```
1 4   PHE
1 5   GLU
1 6   MET
1 7   LEU

…

#
_entity_poly.entity_id                          1
_entity_poly.type                               polypeptide(L)
_entity_poly.nstd_linkage                       no
_entity_poly.nstd_monomer                       no
_entity_poly.pdbx_seq_one_letter_code
;MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAAKSELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGIL
RNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTWD
AYKNL
;
_entity_poly.pdbx_seq_one_letter_code_can
;MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAAKSELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGIL
RNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTWD
AYKNL
;
#
…
#
loop_
_struct_asym.id
_struct_asym.pdbx_blank_PDB_chainid_flag
_struct_asym.pdbx_modified
_struct_asym.entity_id
_struct_asym.details
A Y N 1 ?
B N N 2 ?
C N N 2 ?
D N N 3 ?
E N N 3 ?
F N N 4 ?
…
#
loop_
_struct_conf.conf_type_id
_struct_conf.id
_struct_conf.pdbx_PDB_helix_id
_struct_conf.beg_label_comp_id
_struct_conf.beg_label_asym_id
_struct_conf.beg_label_seq_id
_struct_conf.pdbx_beg_PDB_ins_code
_struct_conf.end_label_comp_id
_struct_conf.end_label_asym_id
_struct_conf.end_label_seq_id
_struct_conf.pdbx_end_PDB_ins_code
_struct_conf.beg_auth_comp_id
_struct_conf.beg_auth_asym_id
_struct_conf.beg_auth_seq_id
_struct_conf.end_auth_comp_id
_struct_conf.end_auth_asym_id
_struct_conf.end_auth_seq_id
_struct_conf.pdbx_PDB_helix_class
_struct_conf.details
_struct_conf.pdbx_PDB_helix_length
```

```
HELX_P HELX_P1  1  ASN A 2   ? GLY A 12  ? ASN A 2   GLY A 12  1 ? 11
HELX_P HELX_P2  2  ALA A 41  A GLY A 52  ? ALA A 40A GLY A 51  1 ? 12
HELX_P HELX_P3  3  THR A 60  ? ASN A 82  ? THR A 59  ASN A 81  1 ? 23
HELX_P HELX_P4  4  LYS A 84  ? LEU A 92  ? LYS A 83  LEU A 91  1 ? 9
HELX_P HELX_P5  5  ASP A 93  ? GLY A 108 ? ASP A 92  GLY A 107 1 ? 16
HELX_P HELX_P6  6  MET A 107 ? GLY A 114 ? MET A 106 GLY A 113 1 ? 8
HELX_P HELX_P7  7  PHE A 115 ? GLN A 124 ? PHE A 114 GLN A 123 1 ? 10
HELX_P HELX_P8  8  ARG A 126 ? LYS A 136 ? ARG A 125 LYS A 135 1 ? 11
HELX_P HELX_P9  9  SER A 137 ? THR A 143 ? SER A 136 THR A 142 1 ? 7
HELX_P HELX_P10 10 THR A 143 ? GLY A 157 ? THR A 142 GLY A 156 1 ? 15
HELX_P HELX_P11 11 TRP A 159 ? LYS A 163 ? TRP A 158 LYS A 162 5 ? 5
…
#
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_alt_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_entity_id
_atom_site.label_seq_id
_atom_site.pdbx_PDB_ins_code
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.Cartn_x_esd
_atom_site.Cartn_y_esd
_atom_site.Cartn_z_esd
_atom_site.occupancy_esd
_atom_site.B_iso_or_equiv_esd
_atom_site.auth_seq_id
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_atom_id
_atom_site.pdbx_PDB_model_num
ATOM   1    N  N   . MET A 1 1   ? 44.061 -3.277 8.755   1.00 22.03  ? ? ? ? ?
1   MET A N   1
ATOM   2    C  CA  . MET A 1 1   ? 43.619 -1.924 8.869   1.00 14.21  ? ? ? ? ?
1   MET A CA  1
ATOM   3    C  C   . MET A 1 1   ? 42.195 -1.991 9.394   1.00 18.00  ? ? ? ? ?
1   MET A C   1
ATOM   4    O  O   . MET A 1 1   ? 41.523 -2.983 9.193   1.00 16.68  ? ? ? ? ?
1   MET A O   1
ATOM   5    C  CB  . MET A 1 1   ? 43.727 -1.353 7.458   1.00 17.98  ? ? ? ? ?
1   MET A CB  1

…
```

Annexe 3 : Exemple de Fichier PDBML

```
<?xml version="1.0" encoding="UTF-8" ?>
<PDBx:datablock  datablockName="102L"
            xmlns:PDBx="http://deposit.pdb.org/pdbML/pdbx.xsd"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://deposit.pdb.org/pdbML/pdbx.xsd pdbx.xsd">
   <PDBx:atom_sites_footnoteCategory>
      <PDBx:atom_sites_footnote id="1">
         <PDBx:text>RESIDUES 162 - 164 IN WILD-TYPE AND ALL MUTANT LYSOZYMES ARE
EXTREMELY MOBILE.  THUS THE COORDINATES FOR THESE RESIDUES ARE VERY UNRELIABLE.
THIS ENTRY DOES NOT INCLUDE RESIDUES 163 AND 164.</PDBx:text>
      </PDBx:atom_sites_footnote>
      <PDBx:atom_sites_footnote id="2">
         <PDBx:text>SEO 901 FORMS AN S-S LINKAGE TO SEO 902.</PDBx:text>
      </PDBx:atom_sites_footnote>
   </PDBx:atom_sites_footnoteCategory>
   <PDBx:audit_authorCategory>
      <PDBx:audit_author name="Heinz, D.W."></PDBx:audit_author>
      <PDBx:audit_author name="Matthews, B.W."></PDBx:audit_author>
   </PDBx:audit_authorCategory>
…
   <PDBx:database_PDB_matrixCategory>
      <PDBx:database_PDB_matrix entry_id="102L">
         <PDBx:origx11>1.000000</PDBx:origx11>
         <PDBx:origx12>0.000000</PDBx:origx12>
         <PDBx:origx13>0.000000</PDBx:origx13>
         <PDBx:origx21>0.000000</PDBx:origx21>
         <PDBx:origx22>1.000000</PDBx:origx22>
         <PDBx:origx23>0.000000</PDBx:origx23>
         <PDBx:origx31>0.000000</PDBx:origx31>
         <PDBx:origx32>0.000000</PDBx:origx32>
         <PDBx:origx33>1.000000</PDBx:origx33>
         <PDBx:origx_vector1>0.00000</PDBx:origx_vector1>
         <PDBx:origx_vector2>0.00000</PDBx:origx_vector2>
         <PDBx:origx_vector3>0.00000</PDBx:origx_vector3>
      </PDBx:database_PDB_matrix>
   </PDBx:database_PDB_matrixCategory>
 …

   <PDBx:entity_poly_seqCategory>
      <PDBx:entity_poly_seq entity_id="1" num="1"
mon_id="MET"></PDBx:entity_poly_seq>
      <PDBx:entity_poly_seq entity_id="1" num="2"
mon_id="ASN"></PDBx:entity_poly_seq>
…
   <PDBx:entity_polyCategory>
      <PDBx:entity_poly entity_id="1">
         <PDBx:type>polypeptide(L)</PDBx:type>
         <PDBx:nstd_linkage>no</PDBx:nstd_linkage>
         <PDBx:nstd_monomer>no</PDBx:nstd_monomer>

<PDBx:pdbx_seq_one_letter_code>MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAAKSELDKA
IGRNTNGVITKDEAEKLFNQDVDAAVRGIL
RNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTWD
AYKNL</PDBx:pdbx_seq_one_letter_code>
```

```
<PDBx:pdbx_seq_one_letter_code_can>MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAAKSE
LDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGIL
RNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTWD
AYKNL</PDBx:pdbx_seq_one_letter_code_can>
      </PDBx:entity_poly>
   </PDBx:entity_polyCategory>
      …
   <PDBx:pdbx_nonpoly_schemeCategory>
      <PDBx:pdbx_nonpoly_scheme asym_id="B" ndb_seq_num="1">
         <PDBx:entity_id>2</PDBx:entity_id>
         <PDBx:mon_id>CL</PDBx:mon_id>
         <PDBx:pdb_seq_num>173</PDBx:pdb_seq_num>
         <PDBx:auth_seq_num>173</PDBx:auth_seq_num>
         <PDBx:pdb_mon_id>CL</PDBx:pdb_mon_id>
         <PDBx:auth_mon_id>CL</PDBx:auth_mon_id>
      </PDBx:pdbx_nonpoly_scheme>
…
   </PDBx:pdbx_nonpoly_schemeCategory>
   <PDBx:struct_asymCategory>
      <PDBx:struct_asym id="A">
         <PDBx:pdbx_blank_PDB_chainid_flag>Y</PDBx:pdbx_blank_PDB_chainid_flag>
         <PDBx:pdbx_modified>N</PDBx:pdbx_modified>
         <PDBx:entity_id>1</PDBx:entity_id>
      </PDBx:struct_asym>
      <PDBx:struct_asym id="B">
         <PDBx:pdbx_blank_PDB_chainid_flag>N</PDBx:pdbx_blank_PDB_chainid_flag>
         <PDBx:pdbx_modified>N</PDBx:pdbx_modified>
         <PDBx:entity_id>2</PDBx:entity_id>
      </PDBx:struct_asym>
   …
   </PDBx:struct_asymCategory>

   <PDBx:struct_confCategory>
      <PDBx:struct_conf id="HELX_P1">
         <PDBx:conf_type_id>HELX_P</PDBx:conf_type_id>
         <PDBx:pdbx_PDB_helix_id>1</PDBx:pdbx_PDB_helix_id>
         <PDBx:beg_label_comp_id>ASN</PDBx:beg_label_comp_id>
         <PDBx:beg_label_asym_id>A</PDBx:beg_label_asym_id>
         <PDBx:beg_label_seq_id>2</PDBx:beg_label_seq_id>
         <PDBx:end_label_comp_id>GLY</PDBx:end_label_comp_id>
         <PDBx:end_label_asym_id>A</PDBx:end_label_asym_id>
         <PDBx:end_label_seq_id>12</PDBx:end_label_seq_id>
         <PDBx:beg_auth_comp_id>ASN</PDBx:beg_auth_comp_id>
         <PDBx:beg_auth_asym_id>A</PDBx:beg_auth_asym_id>
         <PDBx:beg_auth_seq_id>2</PDBx:beg_auth_seq_id>
         <PDBx:end_auth_comp_id>GLY</PDBx:end_auth_comp_id>
         <PDBx:end_auth_asym_id>A</PDBx:end_auth_asym_id>
         <PDBx:end_auth_seq_id>12</PDBx:end_auth_seq_id>
         <PDBx:pdbx_PDB_helix_class>1</PDBx:pdbx_PDB_helix_class>
         <PDBx:pdbx_PDB_helix_length>11</PDBx:pdbx_PDB_helix_length>
      </PDBx:struct_conf>
      <PDBx:struct_conf id="HELX_P2">
         <PDBx:conf_type_id>HELX_P</PDBx:conf_type_id>
         <PDBx:pdbx_PDB_helix_id>2</PDBx:pdbx_PDB_helix_id>
         <PDBx:beg_label_comp_id>ALA</PDBx:beg_label_comp_id>
         <PDBx:beg_label_asym_id>A</PDBx:beg_label_asym_id>
         <PDBx:beg_label_seq_id>41</PDBx:beg_label_seq_id>
         <PDBx:pdbx_beg_PDB_ins_code>A</PDBx:pdbx_beg_PDB_ins_code>
         <PDBx:end_label_comp_id>GLY</PDBx:end_label_comp_id>
```

```
            <PDBx:end_label_asym_id>A</PDBx:end_label_asym_id>
            <PDBx:end_label_seq_id>52</PDBx:end_label_seq_id>
            <PDBx:beg_auth_comp_id>ALA</PDBx:beg_auth_comp_id>
            <PDBx:beg_auth_asym_id>A</PDBx:beg_auth_asym_id>
            <PDBx:beg_auth_seq_id>40A</PDBx:beg_auth_seq_id>
            <PDBx:end_auth_comp_id>GLY</PDBx:end_auth_comp_id>
            <PDBx:end_auth_asym_id>A</PDBx:end_auth_asym_id>
            <PDBx:end_auth_seq_id>51</PDBx:end_auth_seq_id>
            <PDBx:pdbx_PDB_helix_class>1</PDBx:pdbx_PDB_helix_class>
            <PDBx:pdbx_PDB_helix_length>12</PDBx:pdbx_PDB_helix_length>
         </PDBx:struct_conf>
...
      </PDBx:struct_confCategory>
      <PDBx:atom_siteCategory>
         <PDBx:atom_site id="1">
            <PDBx:group_PDB>ATOM</PDBx:group_PDB>
            <PDBx:type_symbol>N</PDBx:type_symbol>
            <PDBx:label_atom_id>N</PDBx:label_atom_id>
            <PDBx:label_alt_id xsi:nil="true" />
            <PDBx:label_comp_id>MET</PDBx:label_comp_id>
            <PDBx:label_asym_id>A</PDBx:label_asym_id>
            <PDBx:label_entity_id>1</PDBx:label_entity_id>
            <PDBx:label_seq_id>1</PDBx:label_seq_id>
            <PDBx:Cartn_x>44.061</PDBx:Cartn_x>
            <PDBx:Cartn_y>-3.277</PDBx:Cartn_y>
            <PDBx:Cartn_z>8.755</PDBx:Cartn_z>
            <PDBx:occupancy>1.00</PDBx:occupancy>
            <PDBx:B_iso_or_equiv>22.03</PDBx:B_iso_or_equiv>
            <PDBx:auth_seq_id>1</PDBx:auth_seq_id>
            <PDBx:auth_comp_id>MET</PDBx:auth_comp_id>
            <PDBx:auth_asym_id>A</PDBx:auth_asym_id>
            <PDBx:auth_atom_id>N</PDBx:auth_atom_id>
            <PDBx:pdbx_PDB_model_num>1</PDBx:pdbx_PDB_model_num>
         </PDBx:atom_site>
         <PDBx:atom_site id="2">
            <PDBx:group_PDB>ATOM</PDBx:group_PDB>
            <PDBx:type_symbol>C</PDBx:type_symbol>
            <PDBx:label_atom_id>CA</PDBx:label_atom_id>
            <PDBx:label_alt_id xsi:nil="true" />
            <PDBx:label_comp_id>MET</PDBx:label_comp_id>
            <PDBx:label_asym_id>A</PDBx:label_asym_id>
            <PDBx:label_entity_id>1</PDBx:label_entity_id>
            <PDBx:label_seq_id>1</PDBx:label_seq_id>
            <PDBx:Cartn_x>43.619</PDBx:Cartn_x>
            <PDBx:Cartn_y>-1.924</PDBx:Cartn_y>
            <PDBx:Cartn_z>8.869</PDBx:Cartn_z>
            <PDBx:occupancy>1.00</PDBx:occupancy>
            <PDBx:B_iso_or_equiv>14.21</PDBx:B_iso_or_equiv>
            <PDBx:auth_seq_id>1</PDBx:auth_seq_id>
            <PDBx:auth_comp_id>MET</PDBx:auth_comp_id>
            <PDBx:auth_asym_id>A</PDBx:auth_asym_id>
            <PDBx:auth_atom_id>CA</PDBx:auth_atom_id>
            <PDBx:pdbx_PDB_model_num>1</PDBx:pdbx_PDB_model_num>
         </PDBx:atom_site>
...
       </PDBx:atom_site>
```

Annexe 4 : Structure de la base de données de Helix Explorer :

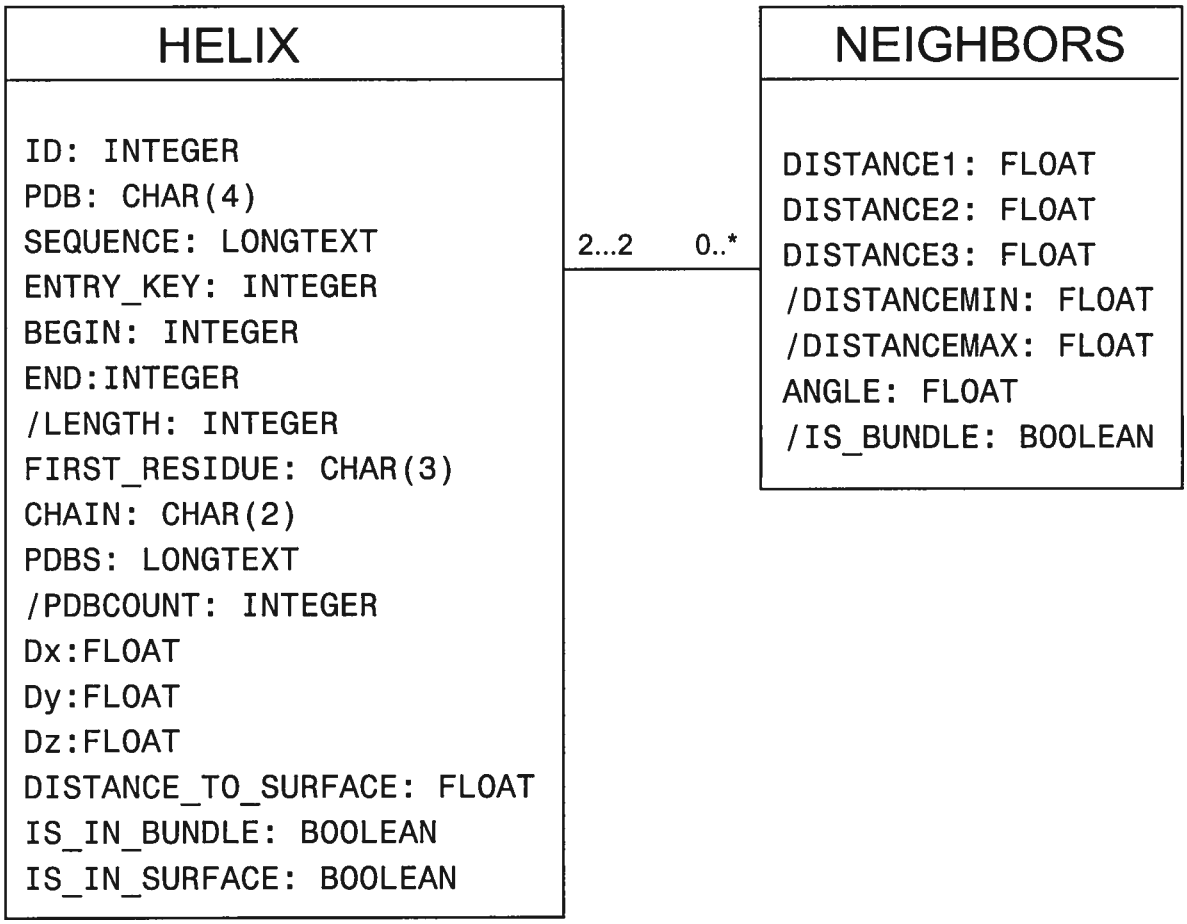

Diagramme UML reliant les deux tables HELIX et NEIGHBORS dans Helix Explorer. Les 11 premières colonnes de la table HELIX sont obtenues par le moyen de requêtes SQL sur la base PDBase. Toutes les autres colonnes de la table HELIX et celles de la table NEIGHBORS sont calculées indirectement depuis la base PDBase. ID est la clef primaire de la table HELIX. ENTRY_KEY est une clef étrangère (de PDBase).

	Field	Type	Collation	Attributes	Null	Default
☐	**pdb**	varchar(4)	ascii_general_ci		No	
☐	**id**	int(11)			No	0
☐	**name**	varchar(11)	ascii_general_ci		Yes	*NULL*
☐	**sequence**	longtext	ascii_general_ci		Yes	*NULL*
☐	**entry_key**	int(11)			No	0
☐	**begin**	smallint(6)			Yes	*NULL*
☐	**end**	smallint(6)			Yes	*NULL*
☐	**length**	smallint(6)			Yes	*NULL*
☐	**first_residu**	char(3)	ascii_general_ci		Yes	*NULL*
☐	**chain**	char(2)	ascii_general_ci		Yes	*NULL*
☐	**pdbs**	longtext	ascii_general_ci		Yes	*NULL*
☐	**pdbcount**	bigint(21)			No	0
☐	**dx**	float			No	0
☐	**dy**	float			No	0
☐	**dz**	float			No	0
☐	**distance_to_surface**	float			No	0
☐	**is_in_bundle**	binary(1)			No	0
☐	**is_in_surface**	binary(1)			No	0

Structure du tableau des structures secondaires

	Field	Type	Collation	Attributes	Null	Default	Extra
☐	**helix1**	longtext	ascii_general_ci		No		
☐	**helix2**	longtext	ascii_general_ci		No		
☐	**key1**	int(11)			No	0	
☐	**key2**	int(11)			No	0	
☐	**distance1**	float			No	0	
☐	**distance2**	float			No	0	
☐	**distance3**	float			No	0	
☐	**distancemin**	float			No	0	
☐	**distancemax**	float			No	0	
☐	**angle**	float			No	0	
☐	**pdb**	varchar(4)	ascii_general_ci		No		
☐	**chain**	char(2)	ascii_general_ci		No		
☐	**is_bundle**	binary(1)			No	0	

Structure du tableau des paires (ou des voisins)

Note : Le code SQL qui suit sert à la construction des tableaux précédents. Les colonnes *dx*, *dy*, *dz*, *distance_to_surface*, *distance1*, *distance2*, *distance3* et *angle* de ces deux tableaux sont remplies indépendemment par d'autres programmes dont le code (en Java) n'est pas rapporté ici.

```
DROP TABLE IF EXISTS `_HELICES_tmp_CAN`;
CREATE TEMPORARY TABLE `_HELICES_tmp_CAN`
(
SELECT       MMS_ENTRY.id AS pdb,
       STRUCT_CONF.struct_conf_key AS id,
       STRUCT_CONF.id AS name,
       ENTITY_POLY.pdbx_seq_one_letter_code_can AS sequence,
       MMS_ENTRY.entry_key,
       STRUCT_CONF.beg_label_seq_id AS begin,
       STRUCT_CONF.end_label_seq_id AS end,
       STRUCT_CONF.pdbx_PDB_helix_length AS length,
       STRUCT_CONF.beg_auth_comp_id AS first_residu,
       STRUCT_CONF.beg_label_asym_id AS chain
FROM   ENTITY_POLY, STRUCT_CONF, MMS_ENTRY, STRUCT_ASYM
WHERE
       MMS_ENTRY.entry_key = ENTITY_POLY.entry_key AND
       MMS_ENTRY.entry_key = STRUCT_CONF.entry_key AND
       MMS_ENTRY.entry_key = STRUCT_ASYM.entry_key AND
       STRUCT_CONF.conf_type_id = 'HELX_P' AND
```

```
        STRUCT_ASYM.id = STRUCT_CONF.beg_label_asym_id AND
        STRUCT_ASYM.entity_id = ENTITY_POLY.entity_id

);

UPDATE _HELICES_tmp_CAN SET Sequence = REPLACE(Sequence,'\n','');
DROP TABLE IF EXISTS `_HELICES_tmp_CAN_RESTRICTED`;
CREATE TABLE `_HELICES_tmp_CAN_RESTRICTED`
(
SELECT        _HELICES_tmp_CAN.pdb,
        _HELICES_tmp_CAN.id,
        _HELICES_tmp_CAN.name,
        SUBSTRING(    _HELICES_tmp_CAN.sequence,
                      _HELICES_tmp_CAN.begin,
                      _HELICES_tmp_CAN.end - _HELICES_tmp_CAN.begin + 1
                ) AS sequence,
        _HELICES_tmp_CAN.entry_key,
        _HELICES_tmp_CAN.begin,
        _HELICES_tmp_CAN.end,
        _HELICES_tmp_CAN.length,
        _HELICES_tmp_CAN.first_residu,
        _HELICES_tmp_CAN.chain
FROM    _HELICES_tmp_CAN
);

DROP TABLE `_HELICES_tmp_CAN`;
DROP TABLE IF EXISTS `_HELIX`;
RENAME TABLE `_HELICES_tmp_CAN_RESTRICTED` TO `_HELIX`;

ALTER TABLE `_HELIX` CHANGE `begin` `begin` SMALLINT NULL DEFAULT NULL;
ALTER TABLE `_HELIX` CHANGE `end` `end` SMALLINT NULL DEFAULT NULL;
ALTER TABLE `_HELIX` CHANGE `length` `length` SMALLINT NULL DEFAULT NULL;
ALTER TABLE `_HELIX` CHANGE `pdb` `pdb` VARCHAR( 4 ) CHARACTER SET ascii COLLATE
ascii_general_ci NOT NULL;
ALTER TABLE `_HELIX` CHANGE `name` `name` VARCHAR( 11 ) CHARACTER SET ascii
COLLATE ascii_general_ci NULL DEFAULT NULL;
ALTER TABLE `_HELIX` CHANGE `first_residu` `first_residu` VARCHAR( 3 ) CHARACTER
SET ascii COLLATE ascii_general_ci NULL DEFAULT NULL;
ALTER TABLE `_HELIX` CHANGE `sequence` `sequence` LONGTEXT CHARACTER SET ascii
COLLATE ascii_general_ci NULL DEFAULT NULL ;


CREATE TEMPORARY TABLE `_HELIX_PDB_tmp` AS
        SELECT              sequence, GROUP_CONCAT( DISTINCT (pdb) ) AS pdbs,
count( * ) AS pdbscount
        FROM                _HELIX
        GROUP BY            sequence;

ALTER TABLE `_HELIX_PDB_tmp` ADD INDEX ( `sequence` ( 10 ) );

CREATE TABLE `_HELICES_WITH_PDBS_tmp`
AS SELECT           _HELIX.*, _HELIX_PDB_tmp.pdbs, _HELIX_PDB_tmp.pdbscount AS
pdbcount
FROM                        _HELIX, _HELIX_PDB_tmp
WHERE                       _HELIX.sequence=_HELIX_PDB_tmp.sequence;


DROP TABLE IF EXISTS `_HELIX`;
RENAME TABLE `_HELICES_WITH_PDBS_tmp` TO `_HELIX`;
```

```
ALTER TABLE `_HELIX` ADD `dx` FLOAT NOT NULL DEFAULT '0' AFTER `pdbcount` ,
                                                                    ADD `dy`
FLOAT NOT NULL DEFAULT '0' AFTER `dx` ,
                                                                    ADD `dz`
FLOAT NOT NULL DEFAULT '0' AFTER `dy` ,
                                                                    ADD
`distance_to_surface` FLOAT NOT NULL DEFAULT '0' AFTER `dz` ;

ALTER TABLE `_HELIX` ADD `is_in_bundle` BINARY DEFAULT '0' NOT NULL ;
ALTER TABLE `_HELIX` ADD `is_in_surface` BINARY DEFAULT '0' NOT NULL ;

ALTER TABLE `_HELIX` ADD INDEX `entry_key_index` ( `entry_key` ) ;
ALTER TABLE `_HELIX` ADD INDEX `id_index` ( `id` ) ;
ALTER TABLE `_HELIX` ADD INDEX `pdb_index` ( `pdb` (4) ) ;


CREATE TABLE `_COOR_Oct26`
(
SELECT ATOM_SITE_Ca.entry_key,
       _HELIX.id,
       _HELIX.pdb,
       _HELIX.name,
       ATOM_SITE_Ca.label_comp_id,
       ATOM_SITE_Ca.label_seq_id,
       ATOM_SITE_Ca.cartn_x,
       ATOM_SITE_Ca.cartn_y,
       ATOM_SITE_Ca.cartn_z,
       _HELIX.chain
FROM   _HELIX, ATOM_SITE_Ca
WHERE 0
);

SELECT ATOM_SITE_Ca.entry_key,
       _HELIX.id,
       _HELIX.pdb,
       _HELIX.name,
       ATOM_SITE_Ca.label_comp_id,
       ATOM_SITE_Ca.label_seq_id,
       ATOM_SITE_Ca.cartn_x,
       ATOM_SITE_Ca.cartn_y,
       ATOM_SITE_Ca.cartn_z,
       _HELIX.chain
INTO OUTFILE 'coordinates_new.tab'
FROM   _HELIX, ATOM_SITE_Ca
WHERE
       _HELIX.entry_key = ATOM_SITE_Ca.entry_key AND
       CAST(ATOM_SITE_Ca.label_seq_id AS UNSIGNED) >= CAST(_HELIX.begin AS
UNSIGNED) AND
       CAST(ATOM_SITE_Ca.label_seq_id AS UNSIGNED) <= CAST(_HELIX.end AS UNSIGNED)
AND
       _HELIX.chain = ATOM_SITE_Ca.label_asym_id
;

LOAD DATA INFILE 'coordinates_new.tab' INTO TABLE _COOR_Oct26;

ALTER TABLE `_COOR_Oct26` ADD INDEX `id_index` ( `id` ) ;
ALTER TABLE `_COOR_Oct26` ADD INDEX `pdb_index` ( `pdb` );
```

```
CREATE TABLE `_PAIRS` (
                    `helix1` LONGTEXT NOT NULL ,
                    `helix2` LONGTEXT NOT NULL ,
                    `key1` INT NOT NULL ,
                    `key2` INT NOT NULL ,
                    `distance1` FLOAT NOT NULL ,
                    `distance2` FLOAT NOT NULL ,
                    `distance3` FLOAT NOT NULL ,
                    `distancemin` FLOAT NOT NULL ,
                    `distancemax` FLOAT NOT NULL ,
                    `angle` FLOAT NOT NULL ,
                    `pdb` VARCHAR( 4 ) NOT NULL,
                    `chain` VARCHAR( 2 ) NOT NULL,
                    `is_bundle` BINARY DEFAULT '0' NOT NULL
) TYPE = MYISAM ;

ALTER TABLE `_PAIRS` ADD PRIMARY KEY ( `key1` , `key2` ) ;
ALTER TABLE `_PAIRS` ADD INDEX `key2_index` ( `key2` ) ;

CREATE TABLE _BUNDLES AS SELECT `key1`, `key2` FROM _PAIRS WHERE  `distance2`<=10
AND `angle`<=10;
ALTER TABLE `_BUNDLES` ADD INDEX `key1_index` ( `key1` ) ;
ALTER TABLE `_BUNDLES` ADD INDEX `key2_index` ( `key2` ) ;

UPDATE _HELIX SET `is_in_bundle`=1 WHERE _HELIX.Id = ANY(SELECT `key1` FROM
_BUNDLES UNION SELECT `key2` FROM _BUNDLES);

UPDATE _PAIRS SET `is_bundle`=1 WHERE `distance2`<=10 AND `angle`<=10;

UPDATE _HELIX SET `is_in_surface`=1 WHERE _HELIX.distance_to_surface<= 2.05;


DROP TABLE IF EXISTS `_PARAMETERS`;
CREATE TABLE IF NOT EXISTS `_PARAMETERS` (
       `hxlastupdate` varchar( 20 ) NOT NULL ,
       `hxdbcount` int( 11 ) NOT NULL ,
       `hxpdbcount` int( 11 ) NOT NULL ,
       `hxpairscount` int( 11 ) NOT NULL ,
       `hxaveragesize` float NOT NULL
) ENGINE = MEMORY DEFAULT CHARSET = latin1;

CREATE TEMPORARY TABLE `_ALL_PDB` AS SELECT DISTINCT(pdb) FROM `_HELIX` GROUP BY
pdb;

INSERT _PARAMETERS VALUES ( (SELECT CURDATE()),

                    (SELECT count(*) FROM `_HELIX`),

                           (SELECT count(*) FROM `_ALL_PDB`),

                           (SELECT count(*) FROM `_PAIRS`),

                           (SELECT AVG(`length`) FROM _HELIX)

                           );
```

Annexe 5 : Calcul des axes des hélices

Note : Le code Java qui suit est une implémentations de l'algorithme *parametric least-square*. Les implémentations des classes *Helix*, *Helix.Monomer*, *Helix.Point*, *HxAxis* ne sont pas fournies ici. Le code ci-bas est néanmoins auto-explicatif.

```
public static HxAxis getHxAxis(Helix helix) {

        Helix.Monomer[] residues = helix.getAllResidues();
        int hxlength = residues.length;

        float D = 0;
        float dx = 0, dy = 0, dz = 0;
        float px = 0, py = 0, pz = 0;

        float Eti = 0, Eti2 = 0;
        float Etirix = 0,Erix = 0,Etiriy = 0,Eriy = 0,Etiriz = 0,Eriz = 0;

        for (int i=0; i<hxlength; i++) {
               Eti += i;
               Eti2 += i*i;

               Erix += residues[i].xcoor;
               Etirix += i*residues[i].xcoor;
               Eriy += residues[i].ycoor;
               Etiriy += i*residues[i].ycoor;
               Eriz += residues[i].zcoor;
               Etiriz += i*residues[i].zcoor;
        }

        D = (Eti2 * hxlength) - (Eti * Eti);

        dx = ((Etirix * hxlength) - (Eti * Erix))/D;
        dy = ((Etiriy * hxlength) - (Eti * Eriy))/D;
        dz = ((Etiriz * hxlength) - (Eti * Eriz))/D;

        px = ((Eti2 * Erix) - (Etirix * Eti))/D;
        py = ((Eti2 * Eriy) - (Etiriy * Eti))/D;
        pz = ((Eti2 * Eriz) - (Etiriz * Eti))/D;

        Helix.Point p = new Helix.Point(px,py,pz);

        float Nd = (float) Math.sqrt(dx*dx + dy*dy + dz*dz);
        Helix.Point d = new Helix.Point(dx/Nd,dy/Nd,dz/Nd);

        return new HxAxis(p,d);
}
```