Université de Montréal

Modélisation des boucles dans les protéines.

par

Sébastien Lefebvre

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

Avril, 1999

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Modélisation des boucles dans les protéines.

présenté par:

Sébastien Lefebvre

a été évalué par un jury composé des personnes suivantes:

Jean-Yves Potvin,	président-rapporteur
François Major,	directeur de recherche
El Mostapha Aboulhamid,	membre du jury

Mémoire accepté le: 99.07.05

SOMMAIRE

La détermination de la structure tridimensionnelle (3D) précise des boucles d'une protéine est un problème difficile. Le nombre de conformations qu'une séquence d'acides aminés peut adopter rend cette tâche ardue. Dans ce mémoire nous proposons un système de construction de boucles fondé sur l'approche de modélisation par satisfaction de contraintes. Une nouvelle contrainte spatiale est introduite, la *contrainte de fermeture* qui permet de sélectionner les meilleurs modèles parmi les boucles construites en mesurant leur précision d'insertion dans la structure. Une nouvelle unité de construction, le *descripteur géométrique de fragment* (DGF) est introduit; c'est une description géométrique d'un fragment de structure 3D. Une banque de DGF de deux résidus (dimères) extraite des boucles d'un sous-ensemble de structures de protéines connues est présentée. La première étape de la méthode de construction consiste à sélectionner le sous-ensemble de DGF qui permet de construire les modèles de boucles satisfaisant la contrainte de fermeture. La seconde étape consiste à construire les modèles de boucles qui satisfont aussi les autres contraintes introduites par l'usager. Le critère de sélection est la contrainte de fermeture. Nous présentons aussi un système de lecture de la Protein Data Bank (PDB) qui est utilisé dans l'analyse des structures 3D de protéines et pour l'extraction des DGF.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

PDB *Protein Data Bank*

PDBSelect25 *Protein Data Bank Select 25%*

ESS *Élément de structure secondaire*

PSC *Problème de satisfaction de contraintes*

DGF *Descripteur géométrique de fragments*

BD *Base de données*

BDDGF *Banque de données de DGF*

TUH *Matrice de transformation unitaire homogène.*

MTUH *Métrique pour les matrices de transformations unitaires homogènes.*

À ma famille

CHAPITRE 1

Le problème

L'étude des protéines permet de mieux comprendre les mécanismes cellulaires ainsi que les autres réactions biochimiques qui foisonnent à l'intérieur des organismes vivants. Les propriétés des protéines dépendent en partie de leur structure tridimensionnelle (3D), résultat du repliement de leur chaîne d'acides aminés. Le repliement d'une protéine forme une structure 3D compacte qui lui donne sa fonctionnalité biologique. Les protéines de structure composent les ensembles complexes tels les virus et tissus. Les enzymes, des agglomérats de protéines, fournissent des sites de liaisons permettant la catalyse de réactions biochimiques essentielles. Les protéines de transport contiennent des sites dédiés à la liaison chimique d'un composé hôte telle que l'hémoglobine qui transporte l'oxygène. Il existe également des protéines qui régularisent les fonctions de l'ADN telle que le répresseur *lac* de *E. coli* qui inhibe l'expression d'un gène. [1]

La fonction biologique d'une protéine repose sur sa capacité à interagir avec les composés chimiques qui se trouvent dans son environnement. Les interactions sont localisées sur des sites spécifiques qui se situent, le plus souvent, à la surface de la protéine là où se trouvent les boucles [21]. Par opposition, les éléments de structure secondaire (ESS) de la protéine se trouvent à l'intérieur de la structure 3D. Les boucles jouent donc un rôle important dans la fonction des protéines [21]. Par exemple, les protéines formant les anticorps possèdent une partie constituée de six boucles différentes qui leur permet de se combiner avec les antigènes. Les longueurs et compositions de ces boucles donnent aux anticorps leur

spécificité [22,27]. Les anticorps possèdent aussi des ESS dont la structure 3D est conservée dans les différents anticorps. Un autre exemple, la bactériorhodopsine, est une protéine trans-membranaire permettant le transfert de protons (atomes d'hydrogène positivement chargés) entre l'intérieur et l'extérieur de la membrane cellulaire. Cette protéine permet la transmission de signaux extérieurs à travers la membrane. Dans le cas de la bactériorhodpsine, les boucles sont placées de chaque côté du canal formé par sept ESS. Connaître la structure 3D de cette protéine et, plus particulièrement de ses boucles, est essentiel à la compréhension des signaux de transduction et des liaisons qu'elle établit avec des ligands externes [28,31].

1.1 La structure et la caractérisation des protéines

La structure d'une protéine est définie par quatre types d'information. Sa séquence d'acides aminés qui constituent la structure primaire. La localisation des boucles et ESS qui constituent la structure secondaire. Les coordonnées cartésiennes dans $\mathbb{R}^3$ de ses atomes qui constituent la structure 3D. Finalement, les interactions entre protéines qui constituent la structure quaternaire [2].

Il existe vingt acides aminés naturels. Ils se différentient par la chaîne latérale qui est lié au carbone alpha (C_α voir Figure 1) du squelette. Leur nom est abrégé en des codes à une lettre et à trois lettres (Tableau I). Les acides aminés sont divisés en trois classes: polaire, chargé et hydrophobe. Cette classification dépend de la nature chimique de la chaîne latérale [1]. Les quatre atomes (N,C_α,C,O) forment le squelette de la protéine (Figure 1).

La conformation d'un résidu représente une distribution 3D de ses atomes. La position relative des atomes O[i-1] et C[i-1] du résidu i-1, et les atomes N[i] et C_α[i] du résidu i, varient peu le long du squelette de la protéine. Les liens chimiques covalents qui les unissent sont très solides et permettent peu de mouvement. Il reste donc deux degrés de liberté pour décrire la position des atomes

TABLEAU I. Codes à une et trois lettres et classe physico-chimique pour les vingt acides aminés.

Code à trois lettres	Code à une lettre	nom de l'acide aminé	classe
ALA	A	Alanine	hydrophobe
CYS	C	Cystéine	polaire
ASP	D	Acide aspartique	chargé
GLU	E	Acide glutamique	chargé
PHE	F	Phenylalanine	hydrophobe
GLY	G	Glycine	hydrophobe
HIS	H	Histidine	polaire
ILE	I	Isoleucine	hydrophobe
LYS	K	Lysine	chargé
LEU	L	Leucine	hydrophobe
MET	M	Méthionine	hydrophobe
ASN	N	Asparagine	polaire
PRO	P	Proline	hydrophobe
GLN	Q	Glutamine	polaire
ARG	R	Arginine	chargé
SER	S	Sérine	polaire
THR	T	Thréonine	polaire
VAL	V	Valine	hydrophobe
TRP	W	Tryptophane	polaire
TYR	Y	Tyrosine	polaire

(C[i-1],N[i],C_α[i],C[i]) du squelette : l'angle de torsion $\phi[i]$(C[i-1],N[i],C_α[i],C[i]) et l'angle de torsion $\psi[i]$(N[i],C_α[i],C[i],O[i]). L'angle de torsion $\phi[i]$ représente le seul degré de liberté déterminant la position du squelette du résidu i par rapport au résidu i-1 (l'angle ψ étant un angle interne ou intra-résidu). La distance C[$i-1$]–

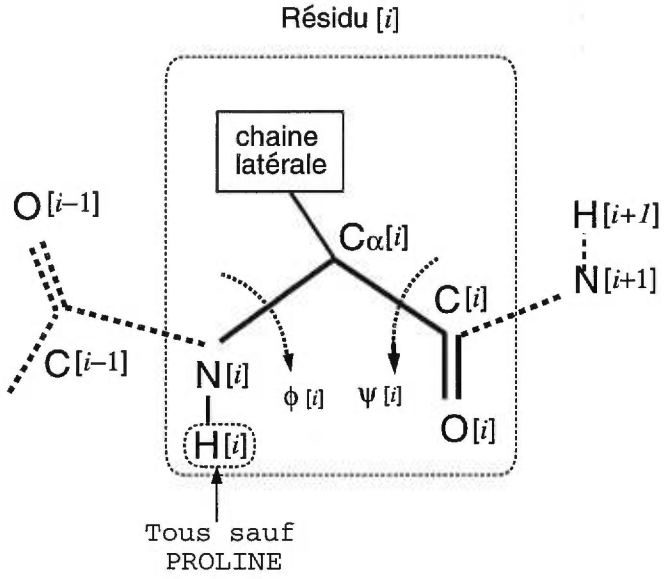

FIGURE 1. La structure chimique du squelette d'un résidu. $\phi[i]$ représente l'angle de torsion entre les atomes (C[i-1],N[i],C_α[i],C[i]) et $\psi[i]$ représente l'angle de torsion entre les atomes (N[i],C_α[i],C[i],O[i]).

N[i] est fixe à 1.33Å et les atomes $C_\alpha[i]$, C[i], O[i] et N[$i+1$] font partie du même plan. L'angle $\phi[i]$ est donc défini comme l'*angle de contact* entre les résidus $i-1$ et i.

Les protéines sont formées de deux types de régions, les ESS et les boucles. Les ESS possèdent deux conformations caractéristiques appelées hélice-α et brin-β. Une hélice-α est une conformation hélicoïdale comprenant 3.6 résidus par tour d'hélice et possèdant des ponts hydrogène entre l'oxygène du résidu i et l'azote du résidu i+4 (Figure 2). Un lien hydrogène représente les forces qui s'établissent entre un groupe donneur d'hydrogène, tel que "N-H", et un accepteur, tel que "C=O". Les ponts hydrogène entre les résidus i et $i+4$ stabilisent la position relative des résidus dans l'hélice.

Un brin-β représente une forme plus étendue d'une chaîne de résidus. Un

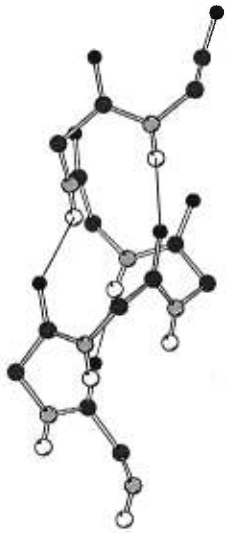

FIGURE 2. La conformation du squelette d'une hélice-α. Les couleurs des atomes : O = noir, C_α et C = gris fonçé, N = gris pâle, H = blanc. Les lignes avec trait simple représentent les ponts hydrogène entre les atomes O[i] et N[$i+4$].

feuillet-β est une collection de brins-β parallèles ou anti-parallèles, stabilisés par des ponts hydrogène entre les résidus des brins-β adjacents (Figure 3). D'un point de vue thermodynamique, les liens hydrogènes constituent un ensemble de forces limitant le domaine de conformations possibles qu'une région de la protéine peut adopter comme dans les ESS.

À l'opposé, une absence de ponts hydrogène rend le squelette plus flexible comme dans les boucles [25]. Les ESS (hélices-α et brins-β) sont reliés par des boucles, des segments de résidus sans apparence de structure organisée (Figure 4). Les résidus des deux ESS en contact avec la boucle sont appelés *résidu de contact* et l'atome C de la ESS1 et l'atome N de la ESS2 sont appelés les *points d'ancrage* (Figure 4). Les boucles représentent les régions les moins bien connues des protéines et les plus difficiles à modéliser car leurs angles (ϕ,ψ) varient beaucoup comparativement à ceux des ESS. D'un point de vue thermodynamique, les boucles sont plus susceptibles de changer de forme [21].

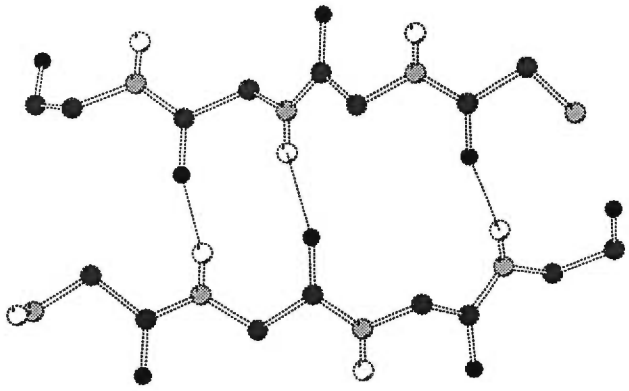

FIGURE 3. La conformation du squelette de deux brins-β formant un feuillet-β. Les couleurs des atomes : O = noir, C_α et C = gris fonçé, N = gris pâle, H = blanc. Les lignes avec trait simple représentent les ponts hydrogène entre les atomes O et N.

La plupart des structures 3D connues des protéines ont été déterminées par cristallographie aux rayons-X. Cette méthode exprérimentale permet d'extraire la position relative des atomes d'une protéine à partir d'un cristal de cette protéine. Un cristal est un agglomérat très ordonné d'une structure de protéine formant un solide. L'utilisation de cette technique dépend de la cristallisation des protéines. Obtenir des critaux de protéines s'avère une tâche difficile, voir impossible dans certains cas. De plus, le processus de détermination de la structure qui s'ensuit est complexe et nécessite parfois plusieurs années de travail. Ces deux facteurs limitent l'application de cette technique [1, 2].

Une autre technique est la résonnance magnétique nucléaire (RMN) qui permet d'extraire de l'information spatiale précise sur une protéine en solution. Cette technique élimine le problème d'obtenir des cristaux. Les résultats obtenus démontrent une précision comparable à la cristallographie aux rayons-X mais la RMN est limitée aux protéines d'environ 1400 résidus ou moins ($\sim$ 20 kDa) [2].

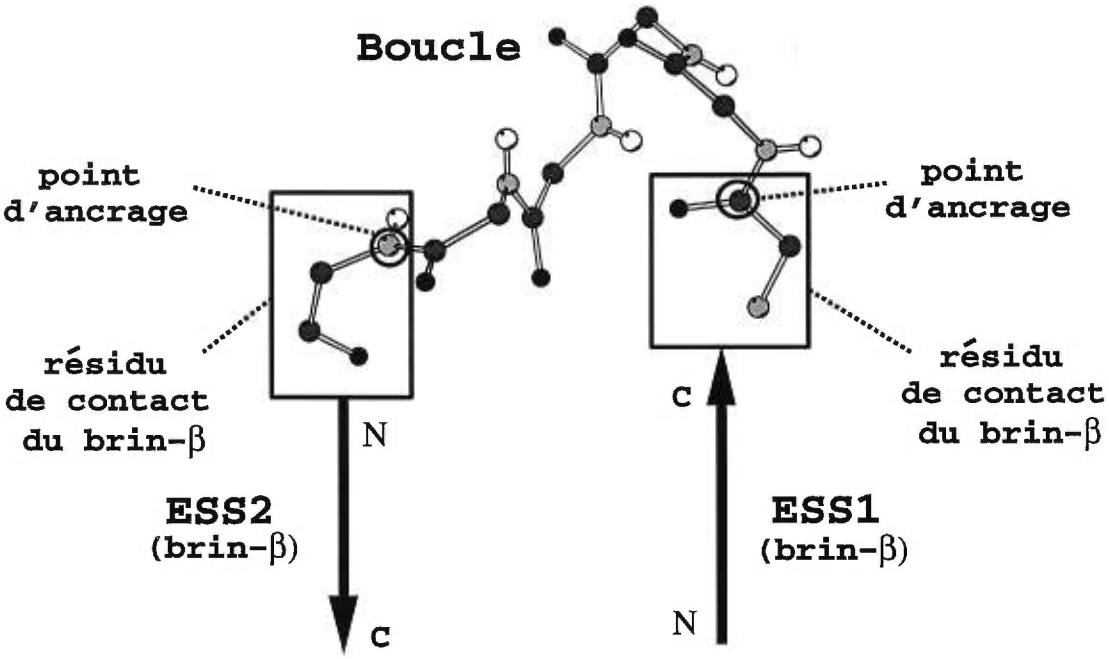

FIGURE 4. La conformation du squelette d'une boucle: Les couleurs des atomes : O = noir, C_α et C = gris fonçé, N = gris pâle, H = blanc.

Les structures qui ont été déterminées par ces deux techniques sont regroupées dans une banque de données appelée la Protein Data Bank (PDB) [3] qui contient actuellement près de 7 000 structures. Le nombre de nouvelles structures croît d'environ 215 par mois [3,17]. Parmi les 7 000 structures, environ les trois quarts sont pratiquement identiques et ont été obtenues par homologie de séquence. De plus, il existe des copies de la même structure résultant de différentes études biologiques et des différentes techniques de cristallographie utilisées [16].

Il existe une relation entre la structure 3D de la protéine et sa séquence d'acides aminés [26]. L'hypothèse thermodynamique sur le repliement des protéines prédit que la structure tertiaire d'une protéine est déterminée unique-

ment par sa séquence. La protéine adopterait la structure 3D représentant son état thermodynamique optimal, soit d'énergie potentielle minimale. La majorité des méthodes de modélisation de protéines se fondent sur cette hypothèse [2].

Les techniques expérimentales de cristallographie et de RMN n'arrivent pas à déterminer les structures du grand nombre de séquences de protéines générées par le projet du génôme humain (plus de 2 838 000 séquences ont été enregistrées jusqu'ici [7]) [6]. La mise au point de méthodes de modélisation de structures 3D est donc souhaitable.

Dans ce mémoire, le problème de modéliser la structure 3D des boucles est posé. Le point de départ de la modélisation 3D des boucles est une structure 3D des ESS. Dans un premier temps, nous passerons en revue les méthodes existantes. Ces méthodes se divisent en deux groupes: celles utilisant une fonction d'énergie potentielle (énergétiques) et celles basées sur des statistiques des structures connues (empiriques). Des résultats récents de Sudarsanam *et al.* [35] montrent que les deux types d'approches peuvent générer des solutions précises. Cependant les méthodes empiriques sont plus rapides.

La méthode de modélisation des boucles présentée dans ce mémoire fait partie de l'effort pour appliquer MC-SYM à la prédiction *de novo* des protéines. Par la suite, nous proposerons un système de construction de boucles fondé sur l'approche de modélisation par satisfaction de contraintes. Une nouvelle contrainte spatiale sera introduite, la *contrainte de fermeture* qui permet de sélectionner les meilleurs modèles parmis les boucles construites en mesurant leur précision d'insertion dans la structure. Une nouvelle unité de construction, le *descripteur géométrique de fragment* (DGF) sera introduite; c'est une description géométrique d'un fragment de structure 3D. Nous présenterons aussi un système de lecture de la Protein Data Bank (PDB) qui est utilisé dans l'analyse des structures 3D de protéines et pour l'extraction des DGF.

La section 2.1 présente les approches basées sur la dynamique moléculaire et la minimisation d'énergie qui utilisent un modèle de fonction d'énergie potentielle et l'équation du mouvement de Newton pour simuler le déplacement des atomes dans le temps. Deux méthodes seront présentées: "la relaxation des liens" [36] et le "recuit simulé" [37] dans lesquelles les atomes des boucles sont libres de se déplacer alors que ceux des ESS sont fixés. Les structures de basse énergie sont sélectionnées comme solutions. Les nombreux calculs de cette approche nécessitent beaucoup de temps CPU. Ces deux méthodes demandent des temps de calcul qui augmentent exponentiellement avec la taille de la boucle. Il est donc rarement possible de traiter le problème pour de grandes boucles ($>$ 15 résidus).

Les angles ϕ et ψ servent de description à la forme d'une boucle. Nous présenterons deux méthodes empiriques utilisant ces angles : la première utilise une classification générale des angles (ϕ, ψ) [33] et la deuxième utilise une discrétisation de l'espace des paires d'angle (ϕ_{i+1}, ψ_i) [35]. La section 2.2 présente la modélisation empirique. Cette approche utilise l'information contenue dans la PDB pour construire une nouvelle boucle. La modélisation par homologie compare la séquence de la boucle à modéliser à celles des protéines connues à l'aide d'un algorithme d'alignement de séquence. La fonction de comparaison utilisée par les algorithmes d'alignement de séquence provient d'études phylogéniques qui déterminent la distance évolutive entre les séquences. Les coordonnées 3D connues d'une boucle dont la séquence est homologue, sont utilisées comme point de départ pour construire la nouvelle boucle. Ces deux séquences étant rarement parfaitement identiques, des ajustements pour les coordonnées 3D sont nécessaires pour les atomes qui diffèrent entre les deux séquences. Pour les séquences de boucles qui n'ont pas d'homologue dans la PDB, on peut trouver des boucles qui satisfont la distance entre les points d'ancrage et ajuster leurs coordonnées 3D pour la séquence de la boucle à modéliser.

Le chapitre 3 présente l'approche MC-SYM pour modéliser les protéines et plus particulièrement les boucles. L'approche MC-SYM consiste à traduire ce problème en un problème de satisfaction de contraintes (PSC) spatiales [8]. Le programme MC-SYM a été développé dans notre laboratoire.

Notre méthode, présentée dans le chapitre 4, s'inspire de la discrétisation de l'espace des paires d'angle (ϕ_{i+1}, ψ_i) [35]. La section 4.1 présente la description géométrique utilisée pour représenter une boucle de protéine. La section 4.2 présente les algorithmes de construction et la section 4.3 présente la contrainte conditionnelle utilisée dans les algorithmes.

Notre système est présenté dans le chapitre 5. La section 5.1 présente le système de lecture que nous avons développé pour extraire et analyser les fichiers en format PDB. Nous avons analysé pour notre étude des boucles un sous-ensemble de la PDB, qui est appelé PDBSelect25. PDBSelect25 contient 320 structures possédant moins de 25% d'identité de séquence [23, 24].

Le chapitre 6 présente les résultats de construction des boucles de différentes protéines.

CHAPITRE 2

Les approches existantes

2.1 La dynamique moléculaire

La "dynamique moléculaire" est une méthode numérique utilisée pour étudier la dynamique d'une protéine dans un environnement donné, par exemple en présence d'eau et d'autres éléments chimiques comme des ions. La structure 3D d'une protéine est représentée par l'ensemble de ses coordonnées cartésiennes atomiques, $P = \{\vec{p}_1, \ldots, \vec{p}_n\}$ où n est le nombre d'atomes, ($\vec{p}_i$ représente les coordonnées X, Y et Z de l'atome i). La structure 3D initiale est modifiée en fonction des équations du mouvement de Newton qui donnent la direction et les forces appliquées sur chaque atome. Ces forces sont obtenues à partir d'un modèle de fonction d'énergie potentielle, $E(\vec{p}_1, \ldots, \vec{p}_n)$ qui tient compte des forces d'attraction et de répulsion que subissent chaque atome du système. La forme analytique que prend E provient d'études structurelles sur les protéines et d'autres petites molécules. Une fonction typique prend la forme suivante [2]:

$$E(\vec{p}_1, \ldots, \vec{p}_n) = \varepsilon_{\text{lien}} + \varepsilon_{\text{ang}} + \varepsilon_{\text{tor}} + \varepsilon_{\text{vdw}} + \varepsilon_{\text{el}}. \tag{2.1}$$

où $\varepsilon_{\text{lien}}$ représente l'énergie associée aux liens atomiques, ε_{ang} représente l'énergie associée aux angles entre les liens atomiques, ε_{tor} représente l'énergie associée aux angles de torsion entre les liens atomiques, ε_{vdw} représente l'énergie de van der Waals qui tient compte des interactions de courtes distances entre les atomes, et ε_{el} représente l'énergie électrostatique qui tient compte des interactions Coulombiennes [2]. L'équation du mouvement de Newton est utilisée pour

traiter la dynamique des atomes :

$$F = -\nabla E(\vec{p}_1, \ldots, \vec{p}_n) = \frac{m\ \partial^2 \vec{p}_i(t)}{\partial t^2} \tag{2.2}$$

où F représente la force exercée sur la position p_i d'un atome au temps t. En pratique, l'équation 2.2 s'applique bien aux simulations de phénomènes dynamiques de l'ordre des nanosecondes avec un pas temporel de 1 fentoseconde. Le coût en calcul associé à ce type de simulation est sa principale limite. Le pas temporel est choisi en fonction de la vitesse de vibration des atomes. Il faut faire un compromis entre la précision de la simulation, la taille du pas temporel, et le temps de la simulation [2].

La température du système est un élément important dans la dynamique moléculaire. La température définit la quantité d'énergie fournie au sytème initialement. Plus la température est élevée et plus les atomes de la protéine vont recevoir de l'énergie et, par conséquent, seront flexibles. Ceci se traduit par une exploration plus grande de l'espace des conformations.

Il est possible d'ajouter au modèle de fonction d'énergie potentielle des termes pour représenter des contraintes spatiales. Ces contraintes spatiales peuvent provenir de résultats d'expériences effectuées en laboratoire. On peut observer expérimentalement la distance entre deux atomes. On peut alors ajouter un terme à la fonction E qui fait augmenter l'énergie lorsque ces deux atomes s'éloignent. La nouvelle fonction force ainsi les deux atomes à rester à proximité l'un de l'autre en pénalisant leur éloignement.

Le choix de la structure 3D de départ est déterminant, elle doit être à un minimum local de E. La dynamique moléculaire peut être utilisée pour chercher une structure lorsqu'on a suffisamment de contraintes de distances (plus de trois par atome). On retrouve cette situation avec des données de RMN. Des centaines d'expériences sont effectuées afin de recueillir des informations sur les positions relatives de plusieurs protons dans la structure. Ces informations sont

transformées en contraintes de distances qui sont introduites dans le modèle de fonction d'énergie potentielle. Les structures atteignant un minimum local et satisfaisant toutes ces contraintes sont proposées comme structures 3D.

Le modèle de fonction d'énergie potentielle sans la loi de Newton donne une méthode de "minimisation d'énergie". Le problème de trouver un minimum de la fonction E est un problème d'optimisation non-linéraire. Des algorithmes d'optimisation non-linéraires tels que la descente du gradient et la descente du gradient conjugué sont utilisés.

2.1.1 La "relaxation des liens"

La première étape consiste à générer une structure 3D de boucle aléatoirement. Ensuite, on l'ajuste afin de satisfaire la contrainte de distance entre les points d'ancrages, d_0 [30]. La distance entre les deux atomes N et C des résidus terminaux de la boucle générée est dénotée d. L'ajustement consiste à modifier par un facteur d/d_0 la distance de chaque lien N-C. Cette modification permet de rapprocher d de d_0 (violant cependant les longueurs canoniques des liens). Par la suite, on procède à une minimisation d'énergie ou à une dynamique moléculaire pour que les liens reprennent des valeurs canoniques [30]. Cette technique est limitée à la construction de petites boucles, soit moins de 10 résidus [36].

Pour évaluer la qualité d'un modèle, on utilise une métrique appelée RMSD (root mean square deviation) qui permet de comparer deux structures 3D, par exemple un modèle à celle du cristal. La RMSD est la racine carrée de la plus petite déviation moyenne au carré ou le plus petit écart-type entre la position des atomes de deux structures. L'équation 2.3 décrit la RMSD.

$$RMSD = \min\left(\sum_{i=1}^{n}\sqrt{\frac{var(i)}{n}}\right) \tag{2.3}$$

où

$$
\begin{aligned}
var(i) &= \sum_{i}^{n}(X_1[i] - X_2[i])^2 + (Y_1[i] - Y_2[i])^2 + (Z_1[i] - Z_2[i])^2 \\
X_1 &= \{x_i \mid \text{ coordonnée en X de l'atome } i \text{ du résidu } 1, i = 1, 2, ..., n\} \\
X_2 &= \{x_i \mid \text{ coordonnée en X de l'atome } i \text{ du résidu } 2, i = 1, 2, ..., n\} \\
Y_1 &= \{y_i \mid \text{ coordonnée en Y de l'atome } i \text{ du résidu } 1, i = 1, 2, ..., n\} \\
Y_2 &= \{y_i \mid \text{ coordonnée en Y de l'atome } i \text{ du résidu } 2, i = 1, 2, ..., n\} \\
Z_1 &= \{z_i \mid \text{ coordonnée en Z de l'atome } i \text{ du résidu } 1, i = 1, 2, ..., n\} \\
Z_2 &= \{z_i \mid \text{ coordonnée en Z de l'atome } i \text{ du résidu } 2, i = 1, 2, ..., n\} \\
n &= \text{ nombre d'atomes comparés}
\end{aligned}
$$

On obtient la valeur de RMSD une fois que l'on a trouvé la meilleur superposition 3D des deux structures. Certains résultats obtenus par Rose *et al.* sont indiqués au tableau II.

TABLEAU II. RMSD des meilleurs modèles de boucles construits par Rose *et al.* [36].

Code PDB	région	taille de la boucle	RMSD
2HPR	28-32	5	0.06
2HPR	27-33	7	1.85
2HPR	66-72	7	0.19
1UBQ	19-24	9	0.33
1UBQ	51-57	7	3.72
8PTI	13-19	7	1.45
8PTI	33-39	7	2.64

2.1.2 L'approche de recuit simulé

La méthode du recuit simulé est appliquée pour trouver le minimum de la fonction d'énergie potentielle [37]. Comme pour la "relaxation des liens", le point de départ est une conformation aléatoire. Le recuit simulé est une dynamique moléculaire où l'on utilise un facteur appelé *température* pour contrôler les régions de l'espace des conformations exploré par la structure. La température représente la quantité d'énergie que l'on donne (ou retire) aux atomes de la protéine pour se déplacer. Cette méthode conciste à chauffer (donner de l'énergie) énormément au début pour donner plus de flexibilité aux atomes et à refroidir (enlever de l'énergie) lentement par la suite. Lorsque la méthode atteint un minimum local de la fonction d'énergie, on ne peut plus enlever de l'énergie à la molécule. Un nouveau point de départ est choisit à l'aide de la température en "chauffant" la molécule. La molécule va donc explorer de nouveau des régions de haute énergie potentielle pour ensuite être refroidit vers un autre minimum local. Le processus de recherche du (ou des) minimum global de la surface d'énergie potentielle est la plupart du temps incalculable car il est inconnu *a priori* et la surface est beaucoup trop complexe. Le meilleur modèle de boucle est celui qui possède la plus faible valeur d'énergie potentielle après un temp de recherche donné.

Carlacci *et al.* [37] ont utilisé cette méthode et leurs résultats sont présentés dans le Tableau III. Ils obtiennent des RMSD entre 1.0 et 1.9Å pour des boucles de 5 à 9 résidus (voir tableau III).

2.2 La modélisation empirique

La modélisation empirique utilise l'information structurelle contenue dans les structures connues, telles que dans la PDB. On étudie la relation entre la séquence d'une boucle et sa structure 3D et on construit par induction de nouvelles boucles. On peut classifier les angles (ϕ,ψ) et prédire la conformation d'une

TABLEAU III. RMSD des modèles de boucles construits par Carlacci *et al* [35].

Code PDB	région	taille de la RSV	RMSD
4PTI	12-16	5	1.03
4PTI	11-17	7	1.61
4PTI	10-18	9	1.87

nouvelle boucle.

2.2.1 Modélisation par homologie

La *modélisation par homologie*, utilise la relation qui existe entre deux séquences similaires et leur structure 3D. Chotia *et al.* ont démontré que deux séquences dont 50% des acides aminés sont identiques ont la plupart du temps une RMSD inférieure ou égale à 1Å [13]. La modélisation par homologie est utilisée pour construire les boucles dont la séquence possède un homologue dans les structures connues.

La règle générale est que la séquence détermine la structure 3D et que la structure 3D détermine la fonction. Cependant, certaines modifications dans la séquence d'une protéine peuvent ne pas affecter sa structure 3D. Une protéine va subir des modifications à travers l'évolution des espèces et certaines modifications vont changer sa structure et d'autres pas. Les ensembles de séquences modifiées dont la structure 3D reste conservée forment des familles qui sont déterminées par les études phylogéniques. L'annexe C présente une technique d'alignement de séquences utilisée pour déterminer l'appartenance à une famille de séquences.

La Figure 5 représente les grandes étapes et le flux d'information de la

modélisation par homologie. En alignant les séquences d'une même famille, on

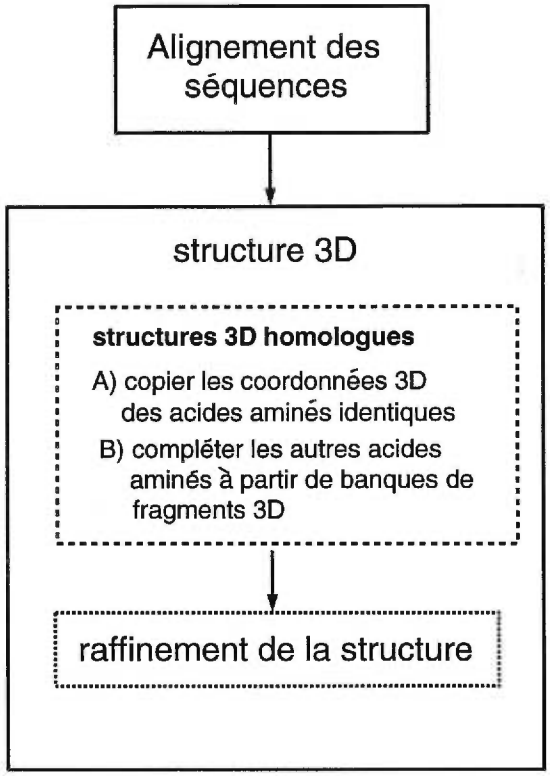

FIGURE 5. Les étapes de la modélisation par homologie de boucles [2].

peut identifier les régions conservées et utiliser les coordonnées 3D d'une séquence dont la structure 3D est connue. L'étape suivante consiste à ajouter les autres résidus. Une étape de raffinement utilisant une minimisation de l'énergie potentielle par exemple, permet de donner un modèle de boucles plus précis [36, 37]. Dans la plupart des systèmes, ajouter une boucle se fait par une recherche des fragments homologues de boucles qui rejoignent le mieux les deux ESS déjà fixés de la structure d'une protéine. La faiblesse de la modélisation par homologie est sa dépendance à la qualité de l'alignement de la nouvelle séquence aux familles existantes. Une nouvelle séquence de boucles n'appartient pas nécessairement à une famille connue.

2.2.2 La classification des angles (ϕ, ψ)

Donate *et al.* [15] ont proposé un modèle de construction de boucles en le représentant par un tuple de sept variables, $\langle seq, ESS1, ESS2, l, \mid \vec{d} \mid, \theta \rangle$. La variable *seq* représente la séquence de la boucle. Les variables $ESS1$ et $ESS2$ représentent les deux ESS adjacentes et peuvent prendre les valeurs 'H' pour hélice et 'E' pour brin-β. La Figure 6 représente les composantes géométriques du modèle. Pour compléter le modèle, Donate *et al.* [15] et Sun *et al.* [34] ont

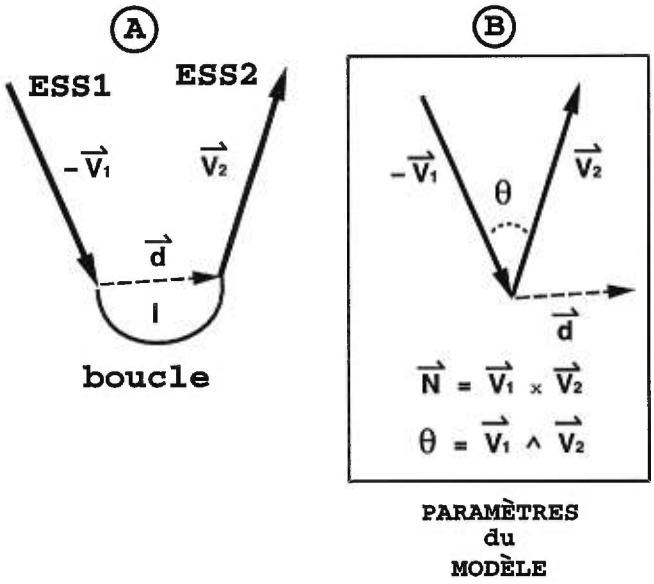

FIGURE 6. Modèle de boucles utilisé par Donate *et al.* [15] : $\langle seq, ESS1, ESS2, l, |\vec{d}|, \theta \rangle$. A) Les vecteurs $\vec{V}1$ et $\vec{V}2$ représentent les axes principaux représentant le sens (atome N vers atome C) du squelette et la direction de la ESS1 entrante et de la ESS2 sortante. La boucle est représentée par la courbe. La variable l représente le nombre de résidus de la boucle. Le vecteur $\vec{d}$ part de l'atome C du dernier résidu de la ESS1 et se termine à l'atome N du premier résidu de la ESS2. B) La variable θ représente l'angle entre $\vec{V}1$ et $\vec{V}2$ après que $\vec{V}1$ soit déplacé de $\vec{d}$.

proposé une classification des angles (ϕ, ψ) des résidus des boucles représentée à

la Figure 7. Par exemple, une boucle de trois résidus peut prendre la conformation "aab" signifiant que les deux premiers résidus possèdent un patron d'angles (ϕ, ψ) qui se trouve dans la région 'a' de la Figure 7 et que le troisième résidu possède un patron d'angles (ϕ, ψ) qui se trouve dans la région 'b' de la Figure 7. Ils ont étudié

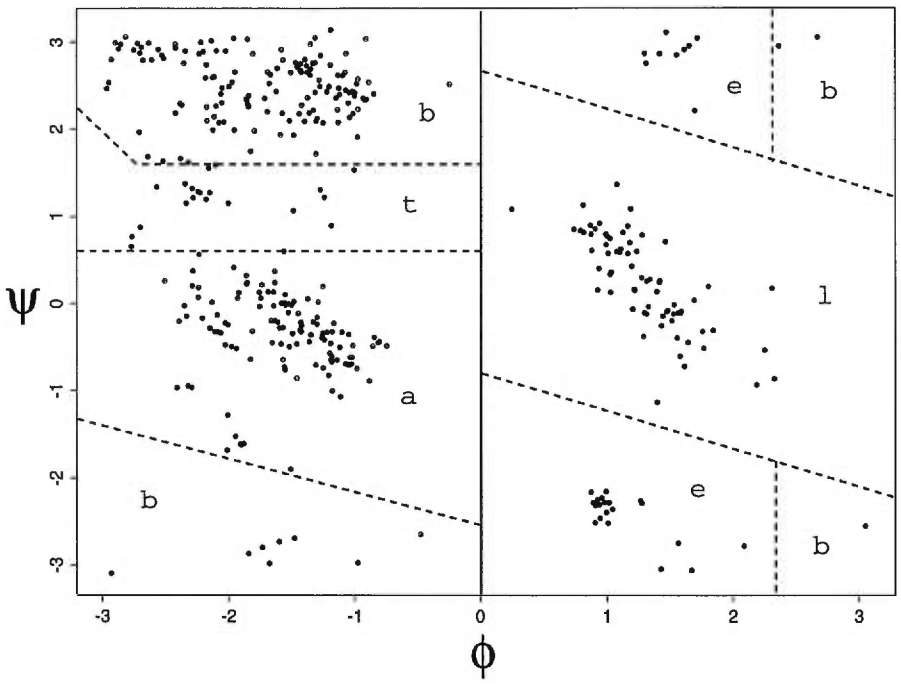

FIGURE 7. Distribution des angles (ϕ, ψ) (en radian) des résidus de boucles de taille 1 dans PDBselect25. Les lettres 'a', 'b', 'e', 'l' et 't' représentent les groupes définis par Donate *et al.* [15].

2024 boucles de tailles variant entre un et huit résidus de 223 protéines différentes. Les boucles sont ainsi regroupées en 161 classes de conformation représentant 63% des boucles des 223 protéines. Le reste des boucles, soit 37%, forment des cas isolés et ne font parties d'aucune classe. Par exemple, la classe "EaE", où E représente un brin-β contient 10 membres parmis les 2024 boucles [15]. Ils ont également observé que les résidus GLY, ASP, PRO, PHE et CYS sont les plus fréquents dans les boucles. La glycine (GLY) possède un squelette flexible, ne possédant

pas de chaîne latérale, et se retrouve dans toutes les classes. Les résidus PHE, ILE, CYS, TRP, VAL et MET favorisent des angles ϕ négatifs. Les résidus les moins fréquents dans les boucles de protéines sont la cystéine (CYS), l'histidine (HIS), le tryptophane (TRP) and la méthionine (MET).

La classification proposée par Donate *et al.* est utilisée dans un algorithme de prédiction de classe de structure de boucles [33]. On commence par chercher une boucle de taille et de séquence similaire dans la PDB. Si on trouve une telle boucle, on l'insère dans la structure à modéliser. Sinon, on cherche des structures de boucles dans la PDB dont les points d'ancrage se superposent bien avec ceux des ESS de la protéine à modéliser. Toutes les structures trouvées sont ensuite ajustées par régression linéaire et celles qui ne causent aucune collision avec les autres résidus sont conservées. On définit les meilleures structures par celles qui ont la plus petite déviation sur les points d'ancrage. Si ces étapes ne réussissent pas à trouver une solution, la classification des boucles mentionnée plus haut est utilisée pour trouver une classe de structures à la boucle à modéliser. A partir de cette classe, une structure grossière est construite puis raffinée par la dynamique moléculaire ou la minimisation d'énergie.

Voici un exemple d'application de la classification proposée par Donate *et al.*: étant donné une séquence "AGIL" d'une boucle connaissant les ESS adjacentes, une hélice-α et un brin-β, on peut obtenir l'information $\langle$ "AGIL", H, E, l, $|\vec{d}|$, θ $\rangle$. A partir de ce sextuplet, l'algorithme prédira que la boucle est dans la classe HabltE avec 50% de probabilité car parmi les boucles de taille 4 de la PDB reliant une hélice-α et un brin-β il y en a 50% qui se retrouvent dans cette classe. On peut maintenant construire une boucle grossière en prenant une valeur de (ϕ,ψ) pour chaque résidu à partir de la Figure 7. La structure de boucles est ensuite raffinée à l'aide de la minimisation d'énergie. L'algorithme fut appliqué sur un ensemble de cas tests indépendants de l'ensemble utilisé pour construire la classification (l'expérience du Jack Knife [29]). Cet ensemble contient

1785 boucles entre deux et cinq résidus dont 62% de ces boucles se trouvent dans une des 161 classes prédéfinies. L'algorithme prédit correctement l'appartenance d'une boucle à une classe dans 52% des cas. Cette méthode prédit la classe d'une boucle à modéliser que l'on peut ensuite utiliser pour obtenir une boucle grossière qu'il faut raffiner par minimisation d'énergie. Ceci permet de trouver de bon points de départ pour des méthodes d'énergie mais ne propose pas de modèle précis. Pour obtenir un modèle précis cette méthode fait face au même problème que les méthodes énergétiques, soit l'utilisation de longs temps de calculs.

2.2.3 Angles (ϕ_{i+1}, ψ_i)

Sudarsanam *et al.* [35] ont proposé une autre approche basée sur une banque d'angles (ϕ_{i+1},ψ_i) entre les résidus i et $i+1$ de toutes les séquences de boucles de la PDB représentant le lien $C[i]$-$N[i+1]$. La banque de données est générée à partir d'un ensemble de structure 3D non redondantes de la PDB, c'est-à-dire qu'aucune protéine de la même famille n'est utilisée. La méthode construit des structures de $n+8$ résidus dont n résidus proviennent de la boucle à modéliser et les huit autres proviennent des deux ESS adjacentes (quatre chacunes). La séquence des $n+8$ résidus est un segment continu de la protéine.

La première étape de la méthode de construction consiste à générer aléatoirement un ensemble de conformations des $n+8$ résidus du squelette à partir de la base de données des angles (ϕ_{i+1},ψ_i). Ils génèrent aléatoirement des milliers de modèles (en général entre 10 000 et 50 000). Ils superposent les huit résidus des ESS du modèle et ceux du cristal pour chacun de leur modèle. Ils ordonnent ensuite tous les candidats à partir de la valeur de la RMSD entre ces huit résidus des ESS du modèle et ceux du cristal. La structure avec la plus petite RMSD devient le meilleur modèle de boucles que propose cette méthode.

En prenant ce modèle, Sudarsanam *et al.* introduisent ce qu'ils appellent la

contrainte de fermeture. Cette contrainte définit la capacité d'une structrure de boucles à venir se joindre à ses deux ESS adjacentes dans la protéine.

Le Tableau IV contient la RMSD des meilleurs structures de boucles d'une série de protéines. Les résultats qu'ils ont obtenus est la motivation première pour construire notre base de données de fragments de deux résidus (dimère) qui remplace les angles ($\phi[i+1]$,$\psi[i]$) et développer une nouvelle représentation plus efficace de la contrainte de fermeture.

TABLEAU IV. RMSD des modèles de boucles construits par Sudarsanam *et al.* [35].

Code PDB	région	taille de la boucle	RMSD
4FXN	127-131	5	0.52
2PCY	80-84	5	0.42
4BPT	6-12	7	0.53
1MCP	56-62	7	2.34
1MCP	102-109	8	1.88
2CCY	31-39	9	1.63

Lorsque la séquence d'une protéine possède au moins 30% d'identité avec une autre, les chances de succès de la modélisation sont grandes [2] mais cette situation se présente dans seulement 15% à 20% des cas. [6]. Pour les autres séquences, il faut appliquer la prédiction *de novo* (vue plus bas) qui reste non-résolu [2]. Il est à noter que 80% des nouvelles séquences ne font pas partie d'une famille de protéines connues [6].

La Figure 8 présente les étapes impliquées dans la prédiction *de novo* qui permet de montrer le problème de la modélisation des boucles, posé dans ce mémoire, dans le contexte de la prédiction de la structure 3D de protéines à

l'aide de l'approche MC-SYM. La prédiction *de novo* de la structure 3D à partir

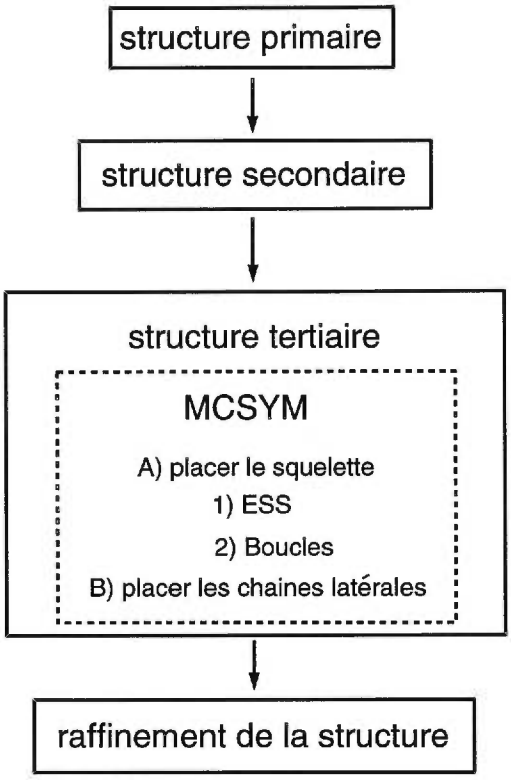

FIGURE 8. Les étapes de la prédiction *de novo* avec l'approche MC-SYM [8].

de la séquence se divise en deux étapes: 1) prédire la structure secondaire et 2) prédire la structure 3D. La prédiction de la structure secondaire est possible par des méthodes statistiques tels les réseaux de neurones [2]. La prédiction de la structure secondaire consiste à prédire l'appartenance de chaque acide aminé de la séquence à un ESS ou à une boucle.

Nous présenterons dans le prochain chapitre, l'approche MC-SYM à la prédiction *de novo* qui a été développée dans notre laboratoire. Cette approche est à la base du développement de notre méthode de modélisation des boucles de protéines.

CHAPITRE 3

L'approche MC-SYM

Dans ce chapitre nous présentons l'approche MC-SYM pour modéliser les protéines et plus particulièrement les boucles. L'approche MC-SYM consiste à traduire ce problème en un problème de satisfaction de contraintes (PSC) spatiales [8]. Le programme MC-SYM a été développé dans notre laboratoire, initialement pour construire des modèles de molécules d'acides ribonucléiques (ARN) [8]. Par la suite, ce programme fut modifié pour construire des protéines [6].

La Figure 9 représente les composantes nécessaires à la modélisation PSC. Le but premier de cette approche est de permettre une collaboration directe entre les expérimentalistes qui recueillent les informations structurelles et les modélisateurs. Les données expérimentales permettent de raffiner les modèles qui permettent de suggérer de nouvelles expériences [18].

La structure d'une protéine peut être approximée par l'arrangement de ses ESS dans $\mathbb{R}^3$. Par exemple, en milieux aqueux, les ESS se replient de manière compacte. Le programme MC-SYM utilise les relations spatiales entre les ESS des structures connues [6] pour en prédire de nouveaux [2]. La complétion des arrangements d'ESS s'effectue en deux étapes: 1) on ajoute les boucles et 2) on ajoute les atomes des chaînes latérales.

L'approche MC-SYM a pour but de construire l'ensemble des structures qui satisfont toutes les contraintes spatiales fournies par l'usager. Ces contraintes sont déduites de l'information recueillie sur la structure 3D de la protéine à

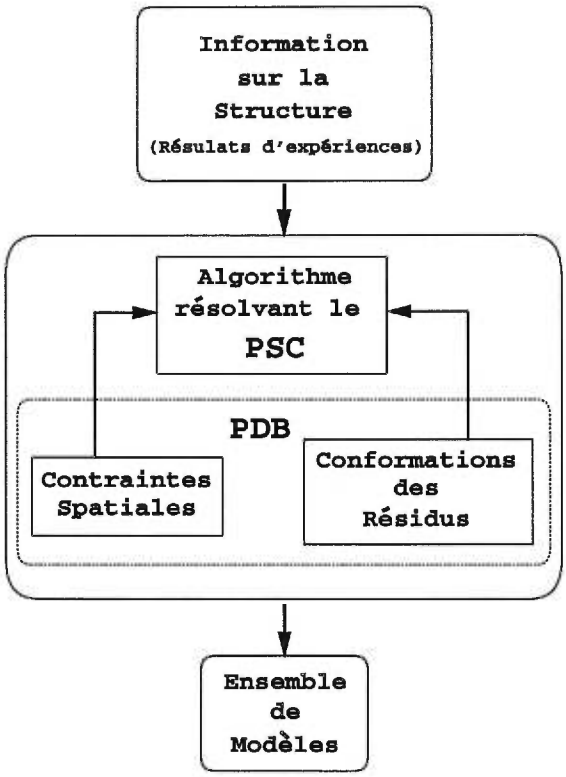

FIGURE 9. Graphe de flux d'information lors de la construction par satisfaction de contraintes [8].

modéliser. Par exemple, on peut observer par RMN que des protons d'un résidu de la séquence se trouve dans les environs de 5Å des protons d'un autre résidu. On peut ainsi introduire dans MC-SYM une contrainte sur le C_α d'un résidu qui doit se trouver à l'intérieur d'un rayon de 5Å du C_α d'un autre résidu.

Le système de construction MC-SYM comporte deux composantes de base: la conformation d'un résidu (Figure 1) qui inclut sa transformation unitaire homogène (TUH) permettant de le placer par rapport à un autre (Figure 10) et la contrainte spatiale.

Il existe deux cas pour l'affectation d'une conformation d'un résidu par rapport à un autre : le cas où les deux résidus sont adjacents et celui où ils ne le sont pas. La relation qui existe entre deux résidus adjacents, A et B, est représentée

par la TUH, $\mathrm{Cfo}_{\mathrm{A}\to\mathrm{B}}(\phi)$, définie comme suit

$$\mathrm{Cfo}_{\mathrm{A}\to\mathrm{B}}\ (\phi) = \frac{\mathrm{RotX}(60) \times \mathrm{RotY}(\phi) \times \mathrm{Ref}_{\mathrm{B}}(\mathrm{N}, \mathrm{C}_\alpha, \mathrm{C})}{\mathrm{Trans}(0, -1.33, 0) \times \mathrm{RotY}(180) \times \mathrm{RotX}(120) \times \mathrm{Ref}_{\mathrm{A}}(\mathrm{C}, \mathrm{C}_\alpha, \mathrm{O})} \tag{3.1}$$

où $\mathrm{Ref}_{\mathrm{B}}$(N,$\mathrm{C}_\alpha$,C) représente les trois axes de référence du résidu B, l'atome C_α sert d'origine, les vecteurs $\overrightarrow{\mathrm{C}_\alpha\ \mathrm{N}}$ et $\overrightarrow{\mathrm{C}_\alpha\ \mathrm{C}}$ forment le plan XY et la normale à ce plan forme l'axe Z (voir Figure 10). RotX(θ) représente la matrice de rotation autour de l'axe des X avec une rotation de θ. RtoY(θ) représente la matrice de rotation autour de l'axe des Y avec une rotation de θ.

A partir d'un résidu A déjà placé dans l'espace 3D, il suffit d'obtenir les coordonnées 3D d'un résidu B et un angle ϕ de contact. Placer un résidu B adjacent au résidu A à partir d'un ensemble de coordonnées 3D dénoté B', du résidu B, consiste à multiplier les coordonnées des atomes par la matrice de TUH

$$\text{Coordonnées3D(B)} = \mathrm{Cfo}_{\mathrm{A}\to\mathrm{B}'}(\phi') \times \text{Coordonnées3D(B')}\ . \tag{3.2}$$

La conformation d'un résidu est représentée par un tuple de 2 variables $\langle$ Coordonnées3D, ϕ $\rangle$. Le champ Coordonnées3D contient les coordonnées atomiques de la conformation. Le champ ϕ contient l'angle de contact avec le résidu précédent. Un ensemble de tuples $\{$ (Coordonnées3D(B), ϕ)$\}$ forme une banque de conformations pour le résidu B.

Pour placer une conformation à la suite d'une autre, il faut passer par une TUH qui utilise le système d'axes locaux des deux conformations (voir Figure 10). La première conformation sert de point de départ et provient de la structure déjà construite alors que la deuxième conformation sert de patron auquel il faut appliquer la transformation.

Le système d'axes local de toute conformation est défini par trois atomes a1, a2 et a3 (ex: C_α, C et O). L'atome a2 sert d'origine, les vecteurs $\overrightarrow{\mathrm{a2\ a1}}$ et $\overrightarrow{\mathrm{a2\ a3}}$ forment le plan XY et la normale à ce plan forme l'axe Z. L'inverse du

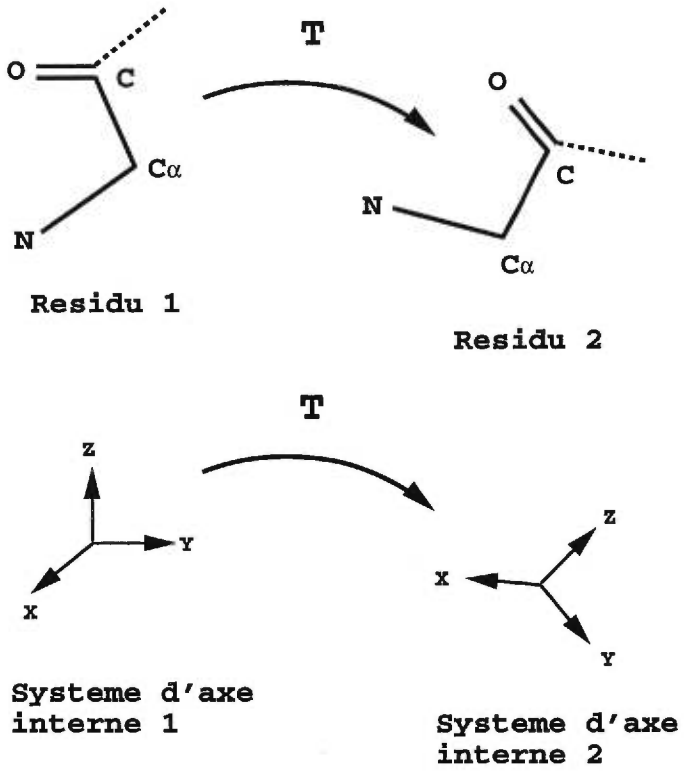

FIGURE 10. TUH permettant de placer un résidu par rapport à un autre.

référentiel d'un résidu X, $\mathrm{Ref}_{\mathrm{X}}^{-1}(\mathrm{a1}, \mathrm{a2}, \mathrm{a3})$ est défini comme étant la translation et la rotation qu'il faut appliquer aux trois atomes a1, a2 et a3 afin d'aligner le système d'axes local avec le système d'axes global. La relation spatiale qui existe entre deux résidus A et B est définie comme suit,

$$\mathrm{Tfo}_{\mathrm{A}\to\mathrm{B}} = \mathrm{Ref}_{\mathrm{A}}^{-1}(\mathrm{a1}, \mathrm{a2}, \mathrm{a3}) \times \mathrm{Ref}_{\mathrm{B}}(\mathrm{b1}, \mathrm{b2}, \mathrm{b3}), \tag{3.3}$$

Une TUH est une matrice dont son inverse est égale à sa transposée. L'équation 3.4 décrit une telle matrice.

$$\mathrm{Tfo}_{\mathrm{A}\to\mathrm{B}} = \begin{bmatrix} Rot_{11} & Rot_{12} & Rot_{13} & 0 \\ Rot_{21} & Rot_{22} & Rot_{23} & 0 \\ Rot_{31} & Rot_{32} & Rot_{33} & 0 \\ Trans_1 & Trans_2 & Trans_3 & 1 \end{bmatrix} \tag{3.4}$$

où

Rot_{ij} représente les éléments de la matrice de rotation avec $(i, j = 1, 2, 3)$

$Trans_i$ représente les éléments du vecteur de translation avec $(i = 1, 2, 3)$

Deux séries d'atomes sont utilisées soient (N, C_α, C) ou (C_α, C, O). Ces atomes caractérisent la structure 3D de la protéine.

Il est possible d'appliquer une transformation entre deux résidus non adjacents dans la séquence. Par exemple, une relation, $\text{Tfo}_{\text{A1}\rightarrow\text{B1}}$, entre la conformation du résidu A, dénotée A1, et celle du résidu B, dénotée B1, est extraite de la PDB. $\text{Tfo}_{\text{A1}\rightarrow\text{B1}}$ s'applique à une conformation du résidu B, dénoté B2, à partir d'une conformation du résidu A, dénoté A2, comme suit,

$$\begin{aligned}\text{Coordonnées3D}(\text{B2}_{\text{absolue}}) &= \left[\text{Ref}^{-1}_{\text{B2}}(\text{b1}, \text{b2}, \text{b3}) \times \text{Tfo}_{\text{A1}\rightarrow\text{B1}} \times \text{Ref}_{\text{A2}}(\text{a1}, \text{a2}, \text{a3})\right] \\ &\quad \times \text{Coordonnées3D}(\text{B2}_{\text{relatif}}) \ . \qquad (3.5)\end{aligned}$$

La transformation permettant de placer un résidu par rapport à un autre est représentée par un tuple de 4 variables ⟨ Coordonnées3D, Tfo, R1, R2 ⟩. Le champ Coordonnées3D contient les coordonnées atomiques de la conformation du résidu R2. Le champ Tfo contient la TUH observée entre les résidu R1 et R2. Un ensemble $\{(\text{Tfo}_{\text{R1}\rightarrow\text{R2}}, \text{Coordonnées3D(R2)})\}$ forme une base de données permettant de positionner différentes conformations de R2 à partir de n'importe quelle conformation de R1.

Les TUH entre les résidus hydrophobes des ESS des protéines ont été extraites et sont utilisées par MC-SYM pour placer les ESS. Les TUH entre les résidus adjacents sont utilisées pour construire les boucles.

La deuxième composante de base de MC-SYM est la contrainte spatiale. La contrainte spatiale est le seul critère utilisé par MC-SYM pour éliminer des candidats. La contrainte spatiale, dénotée $\text{cst}[i, j, a, b] = (r1, r2)$, s'applique entre

les deux résidus i et j tels que les atomes 'a' et 'b' se retrouvent à l'intérieur de l'intervalle de distance $[r1, r2]$. L'ensemble des contraintes spatiales applicable à une structure en construction est défini comme suit:

$$\begin{aligned} \text{CST} \;=\; \{\; & cst[i,j,a,b] = (r1,r2) \mid i,j \text{ sont des positions de résidu tel que} \\ & i \neq j,\; a,b \text{ sont les atomes respectifs des résidus } i \text{ et } j, \\ & r1, r2 \text{ sont des réels tel que } r1 \leq r2 \;\}\;. \end{aligned} \tag{3.6}$$

Lorsqu'un nouveau résidu à la position k est construit, l'ensemble des contraintes $\{\; cst[i,j,a,b] \mid cst \in CST, i = k \text{ ou } j = k\}$ est vérifié. L'ensemble CST représente une conjonction de contraintes que la structure doit respecter.

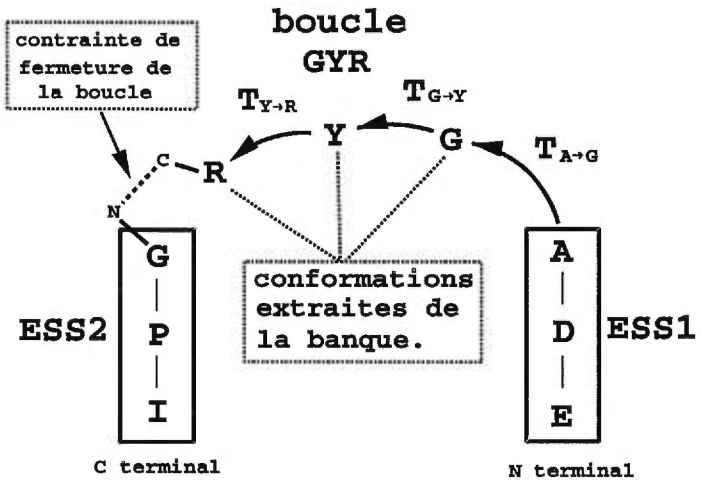

FIGURE 11. Exemple de construction d'une boucle avec MC-SYM. T représente les TUH utilisées pour placer les conformations des résidus de la boucle "GYR".

La Figure 11 présente un exemple d'une construction de boucle avec MC-SYM. Le système utilise un algorithme de retour-arrière pour construire la boucle à partir d'une banque de conformations pré-calculées. Chaque résidu de la boucle possède sa banque de conformations. L'algorithme place une conformation à la fois en l'attachant à la conformation du résidu i-1 (ou i+1 selon la direction choisie). Si la conformation satisfait toutes les contraintes spatiales, on la garde

et on passe au prochain résidu à construire. Sinon, on rejette cette conformation et on essaie avec la prochaine de la banque. Si aucune conformation de la banque ne satisfait toutes les contraintes spatiales alors on retourne en arrière en rejetant la conformation courante du résidu i-1 (ou i+1). On continue ainsi la construction de tous les résidus de la boucle. Chaque feuille de l'arbre représente une structure cohérente et forme l'ensemble des solutions obtenues avec MC-SYM.

Le dernier résidu de la boucle est souvent mal placé par rapport au résidu de contact. Un modèle de boucle provenant de MC-SYM doit donc subir une étape de raffinement par la minimisation d'énergie pour en faire un modèle précis. La rapidité de construction d'un modèle 3D dépend de la taille des banques, du nombre de résidus de la boucle et de la probabilité à satisfaire les contraintes [11]. Les banques de conformations de MC-SYM dépassent 200 000 conformations pour l'ensemble des acides aminés.

Le programme MC-SYM permet de construire un ensemble de modèles pour une boucle à modéliser sans aucune analyse ni critère de sélection autres que les contraintes spatiales spécifiées par l'usager. De ce point de vue, les solutions sont toutes équivalentes. La méthode présentée au chapitre 4, que nous avons développée, est une spécialisation de l'algorithme de construction des boucles qui utilise une banque de fragments spécifiques aux boucles et introduisant une métrique permettant de comparer les modèles.

CHAPITRE 4

La méthode de construction proposée

Ce chapitre présente une nouvelle méthode de construction des boucles développée à partir de l'approche PSC, pour compléter un modèle de protéine contenant uniquement ses ESS. La section 4.1 présente un modèle géométrique de boucle. Dans la section 4.2 nous présentons deux algorithmes qui se trouvent au coeur de la méthode de construction.

4.1 Le modèle

Pour étudier les boucles de protéines, nous avons choisi un modèle qui décrit leur forme générale (Figure 12). Ce modèle est une extension du modèle de Donate *et al.* de la Figure 6. Nous avons modifié leur modèle afin de comprendre l'importance des trois degrés de liberté $(\vec{d}, \theta, \chi)$ (Figure 12) que possèdent les deux vecteurs $(\vec{V1}, \vec{V2})$ l'un par rapport à l'autre, soit une translation et deux rotations.

Le modèle de Donate *et al.* ne tient pas compte de l'angle de torsion entre les vecteurs ESS1 et ESS2 qui est représenté par l'angle χ. Sans cet angle, la géométrie du système "ESS1-boucle-ESS2" n'est pas complètement définie. Les travaux de Sudarsanam *et al.* [35] ont démontré l'importance de connaître l'orientation complète des deux ESS adjacentes à la boucle si on veut la modéliser précisément. Sudarsanam *et al.* [35] ont aussi montré qu'une petite variation de l'angle ϕ entraîne un déplacement important de la ESS2 lors d'un raffinement

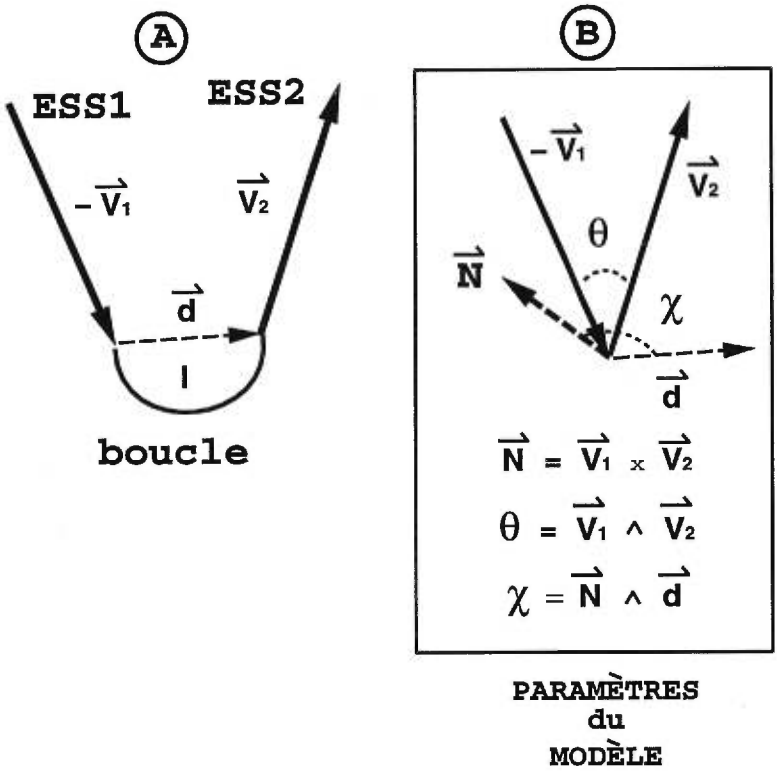

FIGURE 12. Paramètres d'une boucle $\langle l, |\vec{d}|, \theta, \chi \rangle$. A) Représentation vectorielle de la boucle et des paramètres $\vec{d}$ et l. B) Représentation des paramètres θ et χ. L'angle χ représente l'angle de torsion entre $\vec{V}1$ et $\vec{V}2$ le long de l'axe du vecteur $\vec{d}$. Les autres paramètres sont définis à la Figure 6.

par mécanique moléculaire. Autrement dit, si on déplace le point d'ancrage, on déplace toute la ESS2 qui s'y rattache.

Le squelette d'une boucle peut adopter plusieurs structures 3D car il n'est pas restreint par des ponts hydrogène comme pour les ESS mais la distribution des angles (ϕ, ψ) doit permettre de joindre les deux ESS.

Le Tableau V contient les classes de valeurs pour les variables du modèle de la Figure 12 pour des boucles de longueur 1, telles qu'on retrouve dans PDBselect25. Les valeurs du tableau V indiquent qu'il est possible de discriminer les classes de

boucles en utilisant la variable χ. Par exemple si $|\vec{d}| \cong 5.7$ Ået $\theta \cong 1.4$ degré, la valeur de χ permet de distinguer entre la classe "HaH" ($\phi = 1.78$ degré) et la classe "HlH" ($\phi = 1.24$ degré). Ainsi, l'utilisation de l'angle χ augmente la capacité de notre modèle pour caractériser une boucle.

TABLEAU V. Classe pour des boucles de longueur 1.

Classe	$\|\vec{d}\|$ (Å)	θ (degré)	χ (degré)
HaH	5.70	1.46	1.78
HbH	6.70	1.31	1.69
HtH	5.95	1.21	1.33
HeH	7.00	0.73	0.73
HlH	5.64	1.35	1.24
EaH	5.89	1.55	1.84
EbH	6.68	1.34	1.14
EtH	6.40	1.13	2.29
EeH	6.40	0.63	1.82
ElH	5.58	1.76	0.85
HaE	5.88	1.33	1.61
HbE	6.66	0.79	1.54
HtE	6.15	0.76	0.90
HeE	6.83	0.42	1.99
HlE	5.76	0.54	1.41
EaE	5.55	0.60	1.17
EbE	6.55	1.20	1.05
EtE	5.75	1.78	0.60
EeE	6.70	0.39	0.89
ElE	5.50	0.65	1.10

La Figure 13 représente la TUH entre les atomes (Cα,C,O) entre deux résidus en contact. On utilise la TUH pour encoder les trois variables, (d, θ, χ), de notre modèle. Cette transformation spatiale est utilisée pour définir une contrainte de fermeture d'une boucle. Une boucle est construite à partir d'un point de départ

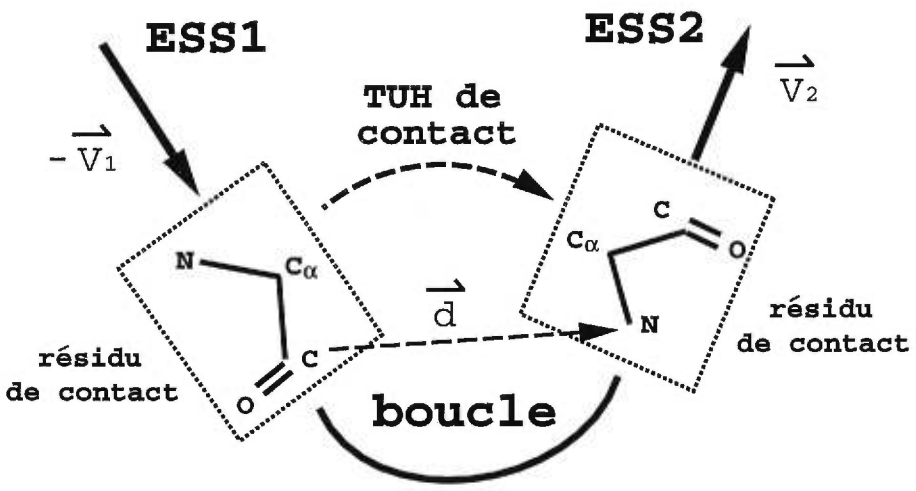

FIGURE 13. Les composantes (d, θ, χ) du modèle de la Figure 12 sont remplacées par une transformation unitaire homogène appelée *TUH de contact.*

dans le premier ESS, ESS1, et d'un point terminal dans le second ESS, ESS2. Le résidu terminal fait parti des résidus qui sont construits par notre méthode. Seul le résidu de départ n'est pas construit. On obtient un point de départ identique et un point terminal différent pour chaque modèle et donc une TUH de contact différente. Ainsi, on peut comparer la TUH de contact de la structure sans boucle et la TUH de contact d'un modèle de boucle. La distance entre ces deux TUH de contact représente la qualité de fermeture d'une boucle. La métrique MTUH (présentée plus bas) est utilisée pour évaluer cette qualité de fermeture. Un score de fermeture de 0 représente une fermeture parfaite.

4.2 Les algorithmes de construction

La méthode de construction des boucles utilise une banque de fragments extraits de la PDBselect25 et un algorithme de retour-arrière. Les fragments utilisés sont constitués de paires de résidus (dimères) représentés par des *descripteurs géométriques de fragments* (DGF).

La relation entre deux résidus adjacents dans une séquence peut être représentée par une TUH définie par trois atomes (ex: C_α, C,O) et les deux conformations des cinq atomes (N,H, C_α, C,O) de chaque résidu. Pour les dimères, les DGF contiennent cette TUH et ces deux conformations. A noter que deux DGF de dimère peuvent se différentier par leur TUH ou bien par leur deux conformations car seule la relation entre les atomes (C_α, C,O) des deux résidus est décrite par la TUH. Deux DGF peuvent donc avoir une TUH identique mais des conformations différentes à cause des atomes N et H des deux dimères.

On peut étendre la définition d'un DGF à la liste des acides aminés du fragment, la liste des conformations de ces résidus, et la liste des TUH entre les atomes (C_α, C,O). Cette définition permet de représenter n'importe quel fragment d'une protéine.

Notre méthode utilise une banque de DGF (BDDGF) de dimères pour construire les boucles. On utilise la métrique MTUH pour trier chaque BDDGF de dimères de façon à obtenir les DGF les plus différents au début et les plus similaires à la fin de chaque banque de DGF. La figure 14 illustre le concept pour une BDDGF d'un dimère donné. Le tri de la BDDGF garanti le meilleur échantillonnage peu importe le nombre de DGF utilisés lors de la recherche en profondeur. MTUH est également utilisée pour comparer les TUH de contact pour les boucles modélisées.

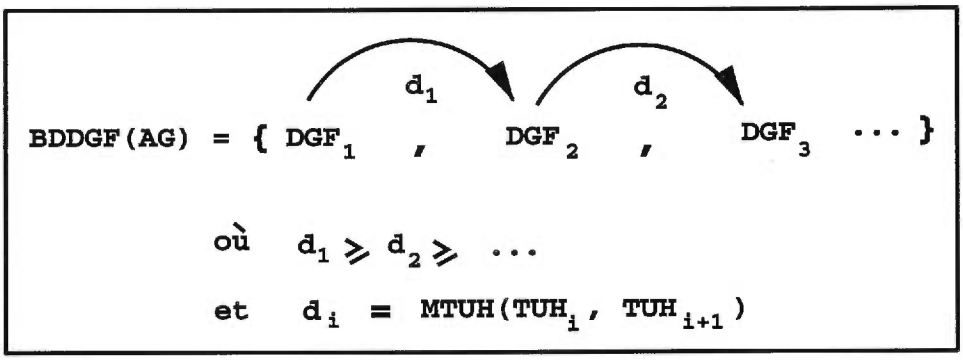

FIGURE 14. Tri de la BDDGF d'un dimère (ex: ALA-GLY) à l'aide de MTUH.

La métrique MTUH est représentée par l'équation 4.1.

$$
\begin{aligned}
MTUH(tfo1, tfo2) &= \delta Trans + \delta Rot \times \left(1.0 - \frac{\delta Trans}{(dT1 + dT2)}\right) \\
\delta Trans &= \sqrt{\sum_i (tfo1.Trans_i - tfo2.Trans_i)^2} \\
\delta Rot &= \sqrt{\sum_i \sum_j (tfo1.Rot_{ij} - tfo2.Rot_{ij})^2} \qquad (4.1) \\
dT1 &= \sqrt{\sum_i (tfo1.Trans_i)^2} \\
dT2 &= \sqrt{\sum_i (tfo2.Trans_i)^2}
\end{aligned}
$$

Les valeurs $\delta Trans/(dT1 + dT2)$ oscillent entre 0 et 1. Ces valeurs permettent de moduler l'effet de la rotation en fonction de la translation. Lorsque $\delta Trans = 0$, la contribution de la rotation devient importante et lorsque $\delta Trans = dT1 + dT2$ la contribution perd de son importance.

La Figure 15 illustre la construction d'une boucle de longueur trois avec la BDDGF. La boucle est construite à partir du résidu de contact ESS1, en insérant un dimère à la fois, jusqu'au résidu de contact ESS2, inclusivement.

La construction utilise le premier résidu pour placer le second, le second

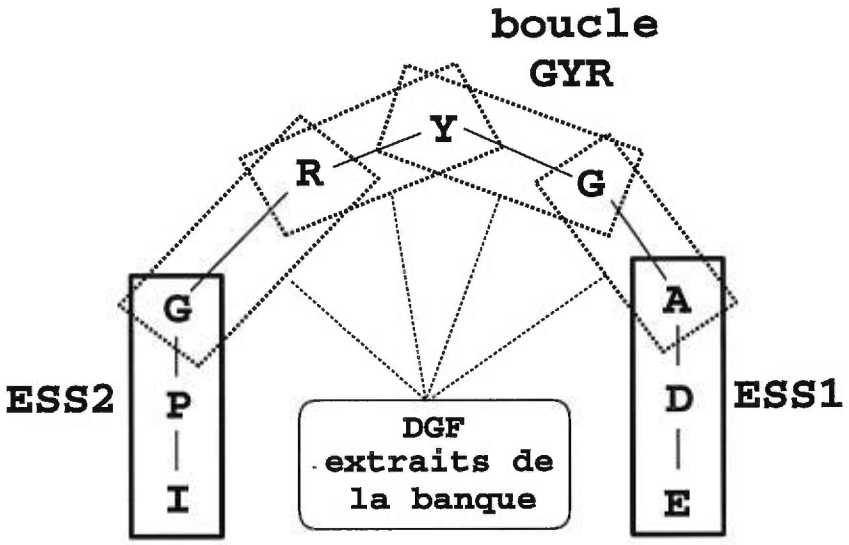

FIGURE 15. Construction 3D d'un modèle de boucle avec la banque de DGF. Les DGF sont dans les boîtes en pointillées.

pour placer le troisième, et ainsi de suite de gauche à droite sur la figure. Par exemple, le DGF "AG" place G et "GY" place Y, etc. Le dernier DGF, "RG", place G qui est le dernier résidu du modèle. La TUH entre le premier et dernier résidu du modèle est ensuite comparée à celle de la TUH de contact. Si le modèle satisfait la contrainte de fermeture alors cette boucle est pré-sélectionnée comme étant une boucle qui s'ajuste bien au reste de la structure. On génère ensuite les coordonnées de la boucle et on vérifie qu'elle satisfait les autres contraintes fournies par l'usager. Si ce n'est pas le cas, la boucle est rejetée.

C'est le rôle de l'algorithme de pré-sélection de parcourir la banque de DGF de dimères afin d'y extraire toutes les combinaisons de dimères qui satisfont la contrainte de fermeture. La TUH entre le premier et le dernier résidu du modèle est évaluée en multipliant les TUH des DGF formant le modèle. Aucun résidu n'est construit car on utilise seulement la TUH de chaque DGF. Une fois la TUH d'un modèle calculée, on la compare avec la TUH de contact. Si la distance entre les deux est plus petite qu'un seuil (contrainte de fermeture) alors le modèle est

conservé car la boucle résultante va bien s'insérer dans la structure existante. La pré-sélection des DGF est l'étape la plus coûteuse en temps de calculs.

L'algorithme de pré-sélection, présenté à la Figure 16 est un algorithme de retour-arrière nécessitant l'utilisation de deux fonctions: Step et MTUH. La fonction Step permet de modifier le parcours de recherche de l'algorithme alors que la fonction MTUH permet l'élagage de branches. Cette étape permet d'extraire tous les modèles de boucles qui satisfont la contrainte de fermeture et sont représentés par une liste de DGF. Cet algorithme retourne la liste des modèles cohérents.

Pour une fouille exhaustive de l'arbre, la fonction Step ne fait que se déplacer en largeur d'un noeud à la fois. La fouille complète de l'arbre de recherche pour la boucle [36-38] de longueur trois de la crambine (1CBN) se fait à l'intérieur de quinze minutes de calculs sur un processeur R10000 (180 Mhz) de SGI. Au-delà de trois résidus, il faut utiliser une heuristique pour obtenir des résultats la journée même.

La fonction StepExh, décrite par l'équation 4.2, est utilisée pour parcourir l'arbre de façon exhaustive.

$$\text{StepExh}(i, \acute{E}tat) \left\{ \acute{E}tat[i] = \acute{E}tat[i] + 1 \right\} \tag{4.2}$$

La fonction StepRand, décrite par l'équation 4.3, est utilisée pour parcourir l'arbre de façon aléatoire. A chaque fois que l'agorithme de recherche retourne en arrière, une fonction aléatoire est appelée.

$$\text{StepRand}(i, \acute{E}tat) \begin{cases} i = i - 1 \text{ si } rand[0,1] < ProbUp \\ \acute{E}tat[i] = \acute{E}tat[i] + rand[\acute{E}tat[i]] \end{cases} \tag{4.3}$$

$rand[a, b]$ génère un nombre aléatoire entre a et b.

$ProbUp \in [0, 1]$ est la probabilité de remonter d'un niveau.

La fonction StepPart, décrite par l'équation 4.4, est utilisée pour parcourir

un sous-ensemble des enfants de chaque noeud de l'arbre.

$$\text{StepPart}(i, \acute{E}tat; Part) \begin{cases} \acute{E}tat[i] = Max[i] \text{ si } \acute{E}tat[i] > Part \times Max[i] \\ \acute{E}tat[i] = \acute{E}tat[i] + 1 \text{ sinon} \end{cases} \tag{4.4}$$

$Part \in [0, 1]$ est la pourcentage des index parcourus.

$Max[i]$ est l'index maximum que $\acute{E}tat[i]$ peut prendre.

Le parcours partiel des noeuds permet de parcourir plus rapidement l'espace de recherche au détriment de la qualité des échantillons. On obtient alors moins de modèles mais ils ont une plus grande distance RMSD entre eux. Les fonctions StepRand et StepPart sont utilisées pour construire les boucles de longueurs supérieures à quatre.

La seconde étape de notre méthode consiste à générer les coordonnées atomiques de nos modèles à partir de la pré-sélection. Les modèles sont construits à partir des conformations des DGF et les contraintes spatiales de l'usager sont vérifiées. La Figure 17 présente l'algorithme de génération des coordonnées des modèles de boucles. Cet algorithme prend en entrée une structure sans boucles et retourne la structure initiale et un ensemble de solutions.

L'utilisation des DGF simplifie les calculs de la construction et donne, en général, de bons résultats (voir section 6). Les solutions de notre méthode sont ordonnées d'après leur score de fermeture à partir de leur TUH de contact. Le meilleur modèle est celui qui possède le plus petit score de fermeture.

RE = Le résidu de contact de ESS1.

RS = Le résidu de contact de ESS1.

n = taille de la boucle.

$Seq = \{R_i \mid i = 1, \ldots, n+2, R_1 = RE, R_{n+2} = RS, R_i \in$ séquence de la boucle $\}$.

$DGFSet$ = banque de DGF d'un dimère défini par $DGFSet.Seq$

$DGFSuperSet = \{DGFSet \mid i = 1, \ldots, n, DGFSet.Seq = \{Seq[i], Seq[i+1]\}$

$TUHRef$ = la TUH de contact de la structure à complèter.

$TUHModel$ = la TUH de contact du modèle.

$État = \{E_1, \ldots, E_n \mid DGFSuperSet[i] \in DGFSuperSet,$

$DGFSuperSet[i][E_i] \in DGFSuperSet[i]\}$

CF = la distance maximale acceptée entre $TUHRef$ et $TUHModel$ soit la contrainte de fermeture.

Sol = liste de tous les $État$ de $DGFSet$ satisfaisant CF.

I = la matrice identité

$MTUH(TUH1, TUH2)$ = métrique permettant d'évaluer la distance entre deux transformations $TUH1$, et $TUH2$.

$Step(i, État)$ = fonction qui effectue un pas dans l'arbre de recherche. Elle peut modifier le niveau i et la valeur de $État[i]$

$i = 1$;

Tant que ($i > 1$)

  Si ($i = DGFSuperSet.Size()$) **alors**

    $TUHModel = I$

    Boucle ($État[i] = 1..DGFSuperSet$[i].Size())

      $TUHModel = TUHModel * DGFSuperSet[i][État[i]].TUH[1]$;

    Si ($MTUH(TUHRef, TUHModel) < CF$) **alors** $Sol = Sol \cup État$

    $Step(i, État)$;

  Sinon Si ($État[i] > DGFSuperSet[i].Size()$) **alors**

    $État[i] = 0$;

    Si ($i = 0$) **Retourner** Sol;

    $i = i - 1$;

    $Step(i, État)$;

  Sinon $Step(i, État)$;

Fin Tant que

FIGURE 16. Pseudo-code pour l'algorithme de pré-sélection des motifs. La fonction *Step* permet d'appliquer différentes techniques de parcours de l'arbre de recherche.

Config = liste des conformations de la structure.

DGFSuperSet = liste des ensembles de DGF dans l'ordre de construction.

DGFSuperSet$[i, j]$ représente le $i$ème ensemble de DGF utilisant son $j$ème DGF.

ÉtatS = liste des états pré-sélectionnés.

CST = liste des contraintes pour chaque résidu à construire.

Sol = $\{$*État* | satisfaisant *CST*$\}$.

Placer(*Config*,*DGFSuper*$[i, j]$,*CST*) = fonction qui place *DGFSuperSet*$[i, j]$ dans *Config* tout en s'assurant de satisfaire *CST*.

Extraire(*DGFSuperSet*$[i]$,*CFG*) = fonction qui extrait de *Config* les résidus construits de *DGFSuperSet*$[i]$.

PasDeSolution = -1

Boucle ($i = 1 \ldots$ | *ÉtatS* |)

  Boucle ($j = 1..$ | *ÉtatS* $[i]$ |)

    Si ($\neg$ *Placer*(*Config*,*DGFSuperSet*$[j$,*ÉtatS*$[i, j]]$,*CST*)) **alors**

      Boucle ($k = 1..(j - 1)$) *Extraire*(*DGFSuperSet*$[j]$,*Config*);

      j = *PasDeSolution*;

      Sort de la Boucle ($j = 1..$ | *ÉtatS* $[i]$ |) ;

  Fin Boucle;

  Si ($j \neq$ *PasDeSolution*) **alors**

    Sol = *Sol* $\cup$ *ÉtatS*$[i]$

    Boucle ($j = 1..$ | *ÉtatS*$[i]$ |) *Extraire*(*DGFSuperSet*$[j]$,*Config*);

Fin Boucle

FIGURE 17. Pseudo-code pour l'algorithme de construction spécifique aux boucles.

La variable $\delta Trans$ représente la distance entre les points d'ancrage. La variable δRot représente la distance euclidienne entre les deux matrices de rotation.

4.3 Les contraintes conditionnelles

Notre méthode utilise des contraintes conditionnelles, c'est-à-dire une contrainte spatiale qui s'applique lorsqu'une autre contrainte, la condition, est satisfaite. Une contrainte conditionnelle est satisfaite si et seulement si elle et sa condition sont satisfaites ou bien si sa condition n'est pas satisfaite.

Cette nouvelle forme de contrainte permet à l'usager d'utiliser des conjonctions de disjonctions pour représenter des contraintes spatiales. Une contrainte conditionnelle se définie comme suit:

$$\begin{aligned} \text{CSTC} \quad = \quad & \neg\, CND \vee CST, \quad \text{tel que} \\ & CND \text{ est la condition} \\ & CST \text{ est la contrainte à satisfaire} \end{aligned} \tag{4.5}$$

Lorsque CND de la relation 4.5 est faux, la contrainte conditionnelle devient une contrainte standard.

Ce nouveau type de contrainte réduit le nombre de vérifications à effectuer comme pour la vérification des collisions entre paires d'atomes. On n'a qu'à définir une condition que les résidus soient à l'intérieur d'une certaine distance, avant de calculer la distance entre deux atomes de ces résidus, par exemple 16Å. La distance entre deux résidus A et B, $d(A, B)$, est évaluée par la distance entre leurs C_α.

Ce type de contrainte permet également d'exprimer des contraintes RMN. Par exemple, prenons trois résidus A, B et C d'une structure à modéliser. L'information structurelle d'expériences de RMN indique un contact entre deux de ces trois résidus et est représentée par une conjonction de disjonctions de

contraintes spatiales. La relation ((A et B) ou exclusif (B et C) ou exclusif (A et C)) est représentée avec quatre contraintes conditionnelles:

$$
\begin{aligned}
CSTC1 &= \neg\,(d(A,B) \geq 16\text{Å}) \vee ((d(A,C) \geq 16\text{Å}) \wedge (d(B,C) \geq 16\text{Å})) \\
CSTC2 &= \neg\,(d(A,C) \geq 16\text{Å}) \vee ((d(A,B) \geq 16\text{Å}) \wedge (d(B,C) \geq 16\text{Å})) \\
CSTC3 &= \neg\,(d(B,C) \geq 16\text{Å}) \vee ((d(A,B) \geq 16\text{Å}) \wedge (d(A,C) \geq 16\text{Å})) \\
CSTC4 &= \neg\,((d(A,B) \geq 16\text{Å}) \wedge (d(A,C) \geq 16\text{Å})) \vee (d(B,C) \geq 16\text{Å})
\end{aligned}
$$

CHAPITRE 5

Le système de construction

Nous avons présenté une combinaison de deux algorithmes de retour-arrière et d'une structure de DGF pour construire des modèles de boucles. En général, il est possible d'utiliser différentes combinaisons d'algorithmes de retour-arrière et de structures de données chacune impliquant un programme différent. Dans ce contexte, il devient intéressant d'avoir une librairie flexible et extensible de primitives permettant le développement de nouveaux programmes de construction de structures 3D de protéines. C'est pour cette raison que nous avons implanté une librairie de primitives dans le language C++ avec la librairie STL, une source de structures de données polymorphiques avec gestion de mémoire intégrée.

La Figure 18 représente l'essentiel du système.

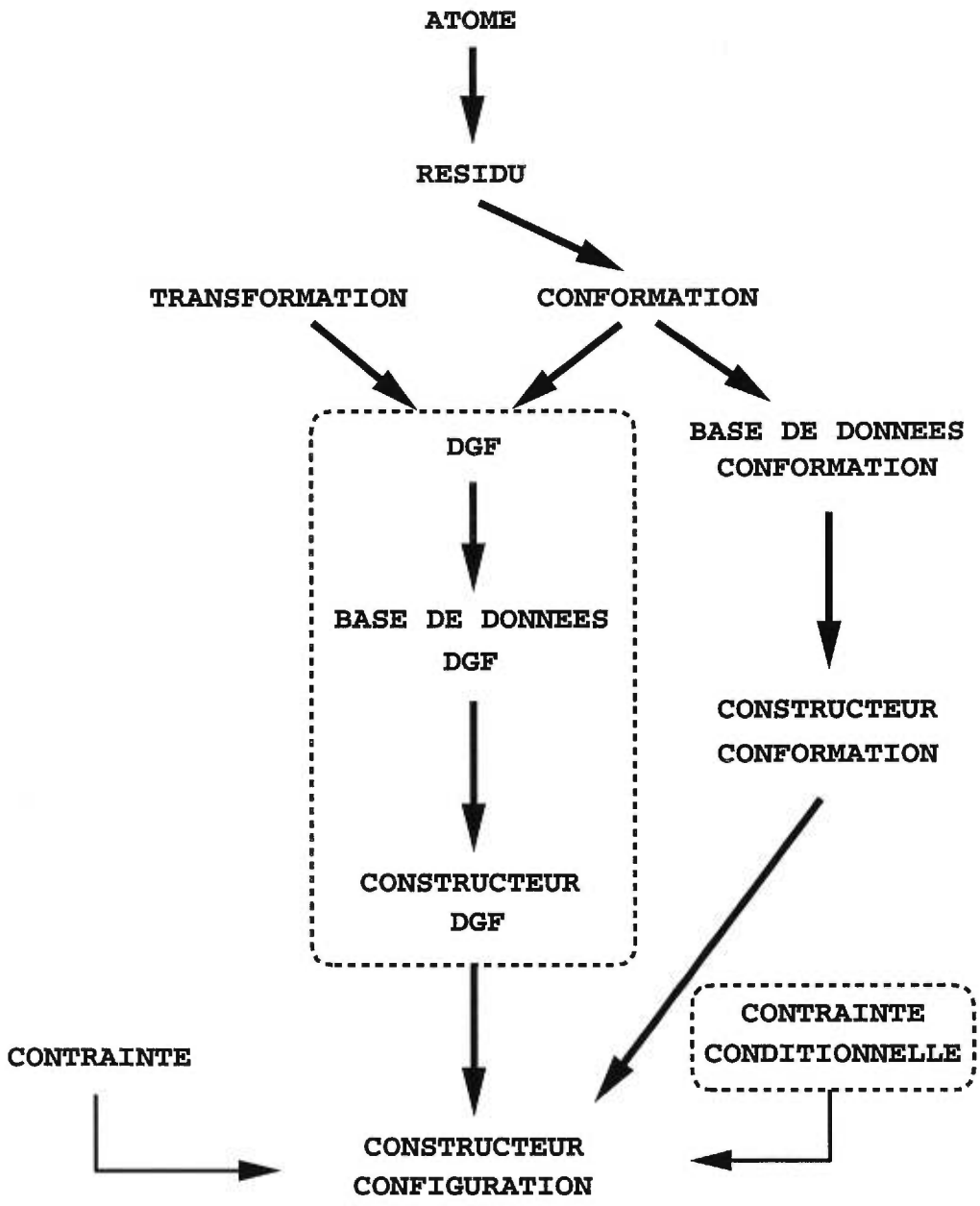

FIGURE 18. Représentation des dépendances entre les composantes du système de construction de boucles. Les cadres en pointillés indiquent les nouveaux concepts par rapport à MCSYM.

Notre sytème contient les classes suivantes:

Atome représente la plus petite unité du système de construction.

Résidu contient un ensemble d'atomes dont quatre essentiels (N, C_α, C, O) qui constituent le squelette de la protéine.

Conformation est une instance tridimensionnelle d'un résidu. Elle est extraite d'un fichier PDB.

Configuration contient une séquence, une liste de structures secondaires et une liste des coordonnées 3D des résidus. Elle est extraite d'un fichier PDB.

Transformation contient une matrice de rotation et un vecteur de translation. Elle permet de placer un résidu par rapport à un autre.

DGF contient toute l'information nécessaire à son application; séquence d'acides aminés, liste de conformations de chaque acide aminé, liste de transformations unitaires homogènes, liste des paires de positions associées aux TUH. Il est extrait de PDB.

Contrainte est un intervalle de distance permis entre deux atomes.

Contrainte conditionnelle contient deux ensembles de contraintes. Le premier ensemble de contraintes sert de condition à l'application du deuxième ensemble de contraintes.

Bases de données de X représentent toutes les classes permettant de construire, de lire, d'écrire et d'ordonner un ensemble de données X. En pratique, elles sont construites à partir des structures de la PDB.

Constructeurs de X représentent toutes les méthodes numériques ainsi que les algorithmes reliés à l'application d'une donnée X.

L'implantation complète de notre système prend la forme de deux librairies en C++ utilisant STL : une librairie de lecture de la PDB et une librairie de construction 3D de protéines. Le système de construction de boucles, *LoopBuilder*, utilise ces deux librairies.

5.1 Le système de lecture de la PDB

Un fichier PDB comporte toute l'information associée à une forme cristalline d'une ou d'un ensemble de protéines. L'information concernant la structure se retrouve distribuée parmi plusieurs enregistrements dans un fichier ASCII. Voici les principaux enregistrements d'un fichier PDB :

SEQRES contient la séquence d'acides aminés de la protéine.

HELIX contient l'intervalle des résidus qui font partie d'une structure en forme d'hélice.

SHEET contient l'intervalle des résidus d'un brin-β faisant partie d'une structure formant un feuillet.

MODEL permet de séparer les structures de plusieurs modèles. Ce champ est utilisé pour les structures déterminées par RMN.

ATOM contient les coordonnées et l'information 3D pour chaque atome de la protéine.

La syntaxe pour chacun de ces champs est donnée dans les références [4] et [5]. Orengo *et al.* [17] affirme que PDB croit au rythme de 215 nouvelles structures chaque mois. La taille de PDB se chiffre aux alentours de 7 000 structures.

Hobohm *et al.* [23, 24] ont pré-sélectionné un ensemble de fichiers PDB, PDBselect25, que nous avons utilisé pour extraire l'information sur les dimères.

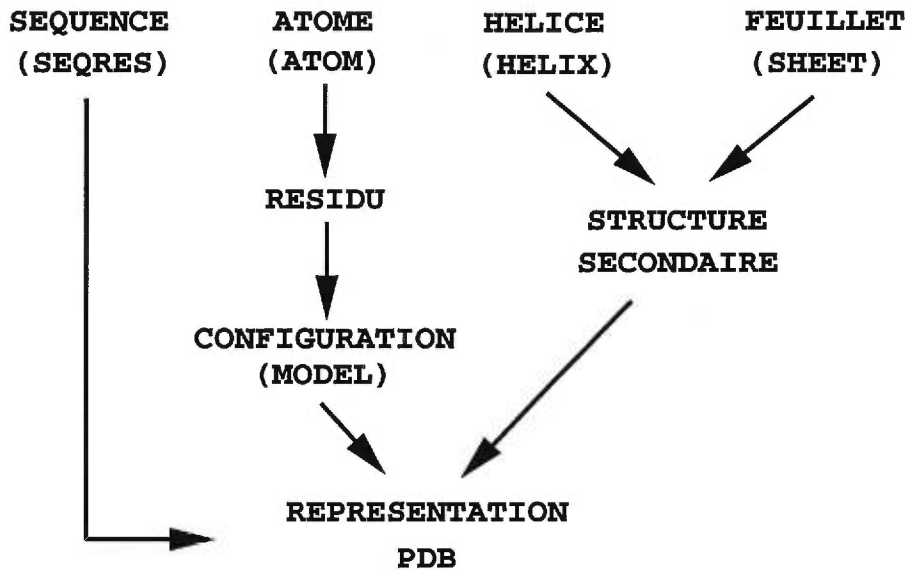

FIGURE 19. Représentation de l'information traitée par le système de lecture de fichiers PDB. Les flèches représentent le flot d'information pendant la lecture du fichier. Les enregistrements de la PDB sont mis entre parenthèses.

La Figure 19 représente les composantes principales du système de lecture de fichier PDB que nous avons développé. Il existe une classe, permettant de lire et d'écrire dans un fichier, pour chacun des enregistrements.

PDB ne valide pas parfaitement la sémantique des structures qui causent des ambigüités entre les enregistrements ATOMS et SEQRES par exemple. Les fichiers PDB contiennent aussi des erreurs de syntaxe.

Une fois en mémoire, il est possible d'effectuer des tests spécifiques sur la sémantique. Par exemple, notre lecteur compare l'information de `SEQUENCE` avec celle de `RESIDU`. Un algorithme d'alignement de séquences est utilisé pour déterminer s'il y a présence de gaps, d'insertions ou de deletions.

Le lecteur vérifie aussi que les structures dans `STRUCTURE SECONDAIRE` ne se chevauchent pas. Il détecte les cas de substitutions de résidus en comparant

les distances entre les deux ensembles d'atomes.

Le programme *ReadPDB* détecte les erreurs dans les fichiers PDB et utilise la classe `C_ParserPDB` pour lire les fichiers PDB. La Figure 20 contient le résultat affiché par le programme ReadPDB une fois exécuté sur le fichier pdb1cbn.ent. Dans cet exemple, le fichier contient des insertions de résidus qui sont rejetées par ReadPDB car les atomes de ces résidus entrent en collision avec d'autres. La séquence de la crambine ne contient pas ces résidus mais l'auteur du fichier a réalisé une étude de la crambine où ces résidus ont été substitués. Le fichier pdb1cbn.ent devrait contenir deux structures avec deux séquences différentes pour ainsi faire correspondre les enregistrements SEQRES et ATOMS.

L'annexe B contient le code source des composantes du système de lecture, *ReadPDB*, et l'interface à l'usager de chaque classe.

```
Running ReadPDB :
chain_id ' '
LOADING Chain Id ' '
        48 residus loaded
 *************  PARSER CONTENT  ***************
>>>>>>>> File = pdb1cbn.ent
>>>>>>>> chain id = ' ' has 48 residus
>>>>>>>> Insertions :
            SER 22B -> rejected overlap with PRO 22
            ILE 25B -> rejected overlap with LEU 25
>>>>>>>> Sequences :
SEQRES   1     46  THR THR CYS CYS PRO SER ILE VAL ALA ARG SER ASN PHE
SEQRES   2     46  ASN VAL CYS ARG LEU PRO GLY THR PRO GLU ALA LEU CYS
SEQRES   3     46  ALA THR TYR THR GLY CYS ILE ILE ILE PRO GLY ALA THR
SEQRES   4     46  CYS PRO GLY ASP TYR ALA ASN
>>>>>>>> Secondary structures :
SHEET    2  S1 3 THR     1  CYS     4 -1
HELIX    1  H1 ILE      7  PRO     19  1
HELIX    2  H2 GLU     23  THR     30  1
SHEET    1  S1 3 CYS    32  ILE     35  0
SHEET    1  S2 1 THR    39  PRO     41  0
SHEET    3  S1 3 ASN    46  ASN     46 -1
 Number of Configurations : 1
 *************  END PARSER CONTENT  ************
 Secondary structure distribution :
Chain id = ' '
E [1-4], L [5-6], H [7-19], L [20-22], H [23-30], L [31-31],
E [32-35], L [36-38], E [39-41], L [42-45], E [46-46]
```

FIGURE 20. Sortie du programme *ReadPDB*, appelé avec le fichier pdb1cbn.ent. Ce programme utilise la classe `C_ParserPDB` (Figure 46).

5.2 La conformation d'un résidu

La conformation d'un résidu est représentée par la classe `C_Confo`. Cette classe contient une instance de la classe `C_Residu` à laquelle on ajoute les angles (ϕ, ψ). La classe `C_Residu` contient plusieurs instances de la classe `C_Atom` représentant les coordonnées 3D des atomes. La Figure 21 présente les spécifications de la classe `C_Point3D` permettant de gérer l'information des coordonnées 3D des atomes de la protéine. Cette classe et la classe `C_Transfo` contiennent les opérateurs de base pour manipuler les coordonnées 3D des atomes. Les spécifications de la classe `C_Atom` sont présentés à la Figure 22. Toutes les instances de la classe `C_Atom` passent par une seule instance globale de la classe `C_GCAtom` (Figure 23). Cette classe s'occupe de la gestion mémoire des objets `C_Atom`.

La classe `C_Residu` (Figure 24) sert de représentation pour les acides aminés. Une structure 3D de protéine est représentée avec un vecteur d'objets `C_Residu`. Toutes les instances de la classe `C_Residu` passe par une seule instance globale de la classe `C_GCResidu` (Figure 25). Cette classe s'occupe de la gestion dynamique de mémoire des objets `C_Residu`.

Les spécifications pour la classe `C_Confo` sont présentées à la Figure 26. Les boucles sont construites en attachant des instances `C_Confo` les unes aux autres. Toutes les manipulations numériques sont effectuées par les méthodes définies dans les classes `C_Point3D` et `C_Transfo` (voir section 5.3). Toutes les instances de la classe `C_Confo` passent par une seule instance globale de la classe `C_GCConfo` (Figure 27). Cette classe s'occupe de la gestion dynamique de mémoire des objets `C_Confo`.

La Figure 28 contient les spécifications de la classe `C_ConfoSet` permettant la gestion des conformations. Le système utilise vingt instances de la classe `C_ConfoSet`, soit une pour chaque acide aminé.

Toute la gestion dynamique de la mémoire du système est effectuée par une série d'instances uniques pour chaque type d'objets. La longueur des boucles varient de 1 à 20 résidus mais la taille de la mémoire reste constante.

La classe `C_ConfoBuild` (Figure 29) ajoute à une structure 3D un résidu. Un index dans la base de données de conformations est nécessaire. Le résidu est placé par la fonction Build. Le système vérifie ensuite si les contraintes associées à ce nouveau résidu sont satisfaites avec la fonction SatisfyConstraint. Si c'est le cas, la partie `C_Residu` de l'objet `C_Confo` (instance `_BuildConfo`) est copiée dans le vecteur de `C_Residu` représentant la structure courante. Le constructeur de résidus applique Build sur une base de données `C_ConfoSet`. Le constructeur de DGF, discuté plus loin, utilise des objets `C_ConfoBuild` pour bâtir une structure 3D de boucle.

```
class C_Point3D
{
public:
  friend class C_Transfo;
  // ***************** Contenu **********************
  float _pX, _pY, _pZ;
  // ***************** Methodes *********************
  // Constructeur +++++
  C_Point3D (void) ;
  C_Point3D (float X,float Y,float Z) ;
  // Modificateurs ++++
  void SetXYZ (float X, float Y, float Z);
  void SetX (float nX);
  void SetY (float nY);
  void SetZ (float nZ);
  void SetToUnity (void);
  void SetToZero (void);
  void Shift (float shift_x,float shift_y,float shift_z);
  C_Point3D & Transform (const C_Transfo * T);
  // operateur +++++
  const C_Point3D& operator= (const C_Point3D & right);
  int operator== (const C_Point3D & right) const;
  C_Point3D operator- (const C_Point3D & right) const;
  C_Point3D operator+ (const C_Point3D & right) const;
  float operator* (const C_Point3D& right) const; // produit croise
  C_Point3D operator* (float right) const;
  friend C_Point3D operator* (float left,const C_Point3D & right);
  C_Point3D operator/ (float right) const ;
  float operator| (const C_Point3D & right) const; // distance
  float operator|| (const C_Point3D & right) const; // distance au carre
  C_Point3D operator- (void) const;
  C_Point3D & operator+= (const C_Point3D & right);
  C_Point3D & operator-= (const C_Point3D & right);
  C_Point3D & operator*= (const C_Point3D & right);
  int Empty (void) const ;
  int IsZero (void) const ;
  int Within (float X,float Y,float Z,float d) const ;
  int Within (float X,float Y,float Z,float d1,float d2) const ;
  // acces au contenu +++++
  float GetX (void) const;
  float GetY (void) const;
  float GetZ (void) const;
  float GetPhi (void) const ;
  float GetTheta (void) const ;
  float Length (void) const ;
  float GetAngleDeg (const C_Point3D * A,const C_Point3D * C ) const ;
  float GetAngleRad (const C_Point3D * A,const C_Point3D * C ) const ;
  static float GetTorsion (const C_Point3D * p1,const C_Point3D * p2,
   const C_Point3D * p3,const C_Point3D * p4);
  // Lecture ecriture +++
  friend ostream& operator<<(ostream& out,const C_Point3D& right);
  friend istream& operator>>(istream& in,C_Point3D& right);
};
```

FIGURE 21. Spécifications pour la classe C_Point3D.

```
extern const int ATOM_N;
extern const int ATOM_CA;
extern const int ATOM_C;
extern const int ATOM_O;
// Cette classe herite des coordonnees 3D et des methodes associes
// provenant de C_Point3D
class C_Atom : public C_Point3D
{
public:
  // ********** Contenu propre a C_Atom *************
  int _Atom;
  // ********** Methodes **********************
  // constructeurs +++++
  C_Atom (void) ;
  C_Atom (int atom,float x,float y,float z) ;
  // Modificateurs ++++++
  int Reset (void);
  int Reset (int atom_token,float x,float y,float z);
  void SetAtom (int atom_token);
  // Operateurs  +++++
  C_Atom& operator= (const C_Atom & X);
  C_Atom& operator= (const C_AtomPdb & X);
  int operator== (const C_Atom & X) const;
  int operator== (const C_AtomPdb & X) const;
  int Empty (void) const;
  int Backbone (void) const;
  int Nucleotide (void) const;
  int Same3D (const C_Atom* X) const;
  int Within (const C_Atom* X,float d) const;
  int Within (const C_Atom* X,float d1,float d2) const;
  int SameAtom (int name) const;
  int SameAtom (const C_Atom *X) const;
  // Acces au contenu  ++++++
  int GetAtom(void) const;
  // Impression ++++
  friend ostream& operator<< (ostream& out,const C_Atom& X);
  int PrintPDB(ostream &out,int serial,char id,char res,int respos) const;
};
```

FIGURE 22. Spécifications pour la classe C_Atom.

```
class C_GCAtom {
public:
  // *************** Contenu ******************
  vector < C_Atom* > _VA; // liste d'objets pres a etre recycle
  // ***************** Methodes *****************
  // Constructeurs +++++
  C_GCAtom ( int nombre_atom );
  // Modificateurs +++++
  C_Atom * Pop (int atom) ;
  int ReSize (int nombre_atom); // nombre d'objets libres en memoire
  void Collect (C_Atom * X) ;
  void Flush (void) ; // vide la memoire
  // Acces au contenu ++++
  int Size (void) const { return _VA.size(); };
};
```

FIGURE 23. Spécifications pour la classe `C_GCAtom`. Le rôle de cette classe est d'allouer et de recycler tous les objets `C_Atom`.

```
class C_Residu {
public:
  // *********  Contenu ****************************
  char _code; // code a une lettre
  int _pos; // position dans la sequence
  vector < C_Atom* > _VAtom; // liste des atomes du residu
  // ********* Methodes ************************
  // constructeurs +++++
  C_Residu (void) ;
  C_Residu (char code1,int pos) ;
  C_Residu (const C_ResiduPdb & ResPdb) ;
  // Modificateurs +++++
  virtual int Reset (void) ;
  int Reset (char code1,int pos);
  void SetPosition (int Pos) ;
  void SetResidu (char Res) ;
  int Add (const C_Atom* X);
  int Add (const C_AtomPdb* X);
  // operateurs ++++
  C_Residu& operator= (const C_Residu & X);
  C_Residu& operator= (const C_ResiduPdb & X);
  int operator== (const C_Residu & X) const;
  int operator== (const C_ResiduPdb & X) const;
  const C_Atom* operator[] (int index) const;
  const C_Atom* operator() (int atom) const;
  void Shift (float x,float y,float z);
  void Shift (const C_Transfo * T) ;
  // place le residu par rapport a Prev ou Next
  int ConnectC (const C_Residu * Prev,float phi);
  int ConnectN (const C_Residu * Next,float phi);
  int Empty (void) const ;
  int IsIn (int atom,int &index) const;
  int IsNucleotide (void) const;
  int SameAtom (const C_Residu * X) const;
  int Same3D (const C_Residu * X) const;
  int BackboneIn (void) const;
  int Colliding(const C_Residu * X) const;
  int Colliding(const C_Atom * X) const;
  int Satisfy (const C_Residu * X,const vector<C_AtomConstraint> & VAC,
       float VdW) const;
  int Within(const C_Atom* X,float d) const;
  int Within(const C_Atom* X,float d1,float d2) const;
  int Within(const C_Atom * X,int atom,float d) const;
  // Acces au contenu +++++++
  int Size(void) const;
  char GetResidu (void) const;
  int GetPosition (void) const;
  int GetAtom (vector<C_Atom*> & VA) const;
  int GetAtom (vector<const C_Atom*> & VA) const;
  float GetPsi (void) const;
  float GetPhi (const C_Residu * X) const;
  // Impression ++++
  friend ostream& operator<< (ostream& out,const C_Residu& X) ;
  int PrintPdb (ostream& out,char id,int & serial) const;
  int PrintPdbLast (ostream& out,char id,int & serial) const;
  int PrintBackbonePdb (ostream& out,char id,int & serial) const;
  int PrintBackbonePdbLast (ostream& out,char id,int & serial) const;
};
```

FIGURE 24. Spécifications pour la classe C_Residu.

```
class C_GCResidu {
public:
  // ********** Contenu **********************
  vector < C_Residu* > _VR;
  // ********** Methodes *********************
  // Constructeurs +++++
  C_GCResidu( int nombre_residu );
  // Modificateurs +++++
  C_Residu * Pop (void) ;
  int ReSize (int n);
  void Collect (C_Residu * X) ;
  void Flush (void) ;
  // Acces au contenu ++++++
  int Size (void) const { return _VR.size(); };
};
```

FIGURE 25. Spécification pour la classe `C_GCResidu`. Le rôle de cette classe est d'allouer et de recycler tous les objets `C_Residu`.

```
// Cette herite du contenu et des methode de la classe C_Residu
class C_Confo : public C_Residu {
public:
  // ***** Contenu propre a C_Confo **********
  // Position de la donnee dans la base de donnees
  long _FilePos;
  float _phi,_psi;
  char _PPclass; // classe (phi,psi) de Donate et al.
  char PdbFile[15]; // fichier de provenance
  char _SS; // a quelle structure secondaire il appartient
  // ******************
  // Methodes globales pour interfacer avce MCSYM
  static int ReadMCSYM (istream &in,C_Confo & X) ;
  static int ReadMCSYMPhiPsi (istream &in,C_Confo & X) ;
  static int WriteMCSYM (ostream &out,const C_Confo & X) ;
  // Constructeurs ++++
  C_Confo (void) ;
  C_Confo (istream &IN) ;
  C_Confo (float phi,const char *PDB_file,long FilePos) ;
  // Modificateurs +++++
  int SetPdbFile (const char *PDB_file) ;
  void SetFilePos (long pos);
  void SetPhi (float phi);
  void SetPPClass (char PPclass);
  void SetSS (char SS);
  void SetResidu (const C_Residu & X);
  void SetResiduBackbone (const C_Residu & X);
  virtual int Reset (void) ;
  int Reset (istream& in) ;
  int ResetPhiPsi (istream& in) ;
  // Operateurs ++++
  C_Confo& operator= (const C_Confo &X);
  int operator== (const C_Confo &X) const;
  // Acces au contenu ++++
  float GetPhi (void) const { return _phi; }
  float GetPsi (void) const { return _psi; }
  long GetFilePos (void) const { return _FilePos; }
  char GetPPclass (void) const { return _PPclass; }
  char GetSS (void) const { return _SS; }
  const char* GetPdbFile (void) const { return _PdbFile; }
  // Impression +++++
  friend ostream& operator<< (ostream& out,const C_Confo & X) ;
};
```

FIGURE 26. Spécifications pour la classe `C_Confo`.

```
class C_GCConfo {
public:
  // ************* Contenu ************************
  vector < C_Confo* > _VC;
  // ************* Methodes ************************
  // Constructeurs ++++
  C_GCConfo( int nombre_confo );
  // Modificateurs
  C_Confo * Pop (void) ;
  int ReSize (int n);
  void Collect (C_Confo * X) ;
  void Flush (void) ;
  // Acces au contenu ++++++
  int Size (void) const { return _VC.size(); };
};
```

FIGURE 27. Spécifications pour la classe `C_GCConfo`. Le rôle de cette classe est d'allouer et de recycler tous les objets `C_Confo`.

```
class C_ConfoSet
{
private:
  // ******** Contenu avec acces limite **************
  int _NextLoadPos;
  char * _FileName; // fichier de la base de donnees
  ifstream * _FilePointer; // pointeur dans le fichier
  vector < long > _ConfoIndex; // position relative des donnees
  vector<float> _Phi,_Psi;
  vector < int > _Index2IndexLoaded;
  vector < int > _IndexLoaded2Index;
  int _Limit; // limite le nombre de confo present en memoire
  vector < C_Confo * > _Confo;
public:
  // ************Contenu *************************
  char _PhiPsiClass; // Classe de Donate et al.; '*' = any
  // ************ Methodes ***********************
  // Constructeurs +++++++
  C_ConfoSet (void) ;
  C_ConfoSet (const char * filename,int limit,char ppclass) ;
  // Modificateurs +++++
  int Reset (void) ;
  int Reset (const char * filename,int limit,char ppclass) ;
  int ResetFilePointer (void) ;
  int SetFileName (const char *name) ;
  int Load (int from,int to) ;
  int Load (void) ;
  int LoadIndex (void) ;
  int ReadIndexFile (const char *name) ;
  int ReadIndexFile (const vector<char> & name) ;
  int Resize (int limit);
  int SortPhiPsiDistance (void);
  // Operateurs ++++
  C_ConfoSet& operator= (const C_ConfoSet &X);
  int operator== (const C_ConfoSet &X) const;
  const C_Confo * operator[] (int index) ;
  int Empty (void) const;
  int IsIn (int index) const;
  int IsLoaded (int index) const;
  // Acces au contenu ++++++
  int Size (void) const ;
  int LoadedSize (void) const ;
  ifstream * GetFilePointer (void) ;
  const char * GetFileName (void) const ;
  char GetPhiPsiClass (void) const ;
  int GetIndex(float phi,float psi,int & index) const;
  int GetIndex(float phi,float psi,float limit,int & index) const;
  // Impression ++++
  int Print (ostream& out) ;
  int PrintIndex (ostream& out) const ;
  int PrintLoaded (ostream& out) const ;
  int PrintLoadedIndex (ostream& out) const ;
};
```

FIGURE 28. Spécifications pour la classe `C_ConfoSet`. Cette classe contient un ensemble d'objets `C_Confo` et les méthodes pour gérer une base de données de conformations.

```
class C_ConfoBuild {
public:
  // ***************** Contenu ***********************
  int _DepPos; // position du residu d'appui dans la sequence
  int _Pos; // position du residu a construire
  int _DepIndex,_PosIndex; // index des vecteurs
  int _Max; // nombre maximum de conformation a utiliser
  int _Index; // index dans la BD
  C_ConfoSet * _ConfoSet; // base de donnees
  C_Confo _BuildConfo; // conformation a placer dans la structure 3D
  // Contraintes spatiales associees au residu a construire
  C_ResiduConstraint _RC;
  // ***************** Methodes ***********************
  // Constructeurs ++++
  C_ConfoBuild (void);
  C_ConfoBuild (int pos,int posindex,char ss,int max_confo,int index,
int deppos,int depindex,C_ConfoSet * set);
  // Modificateurs ++++
  int Reset (void);
  int Load (int index);
  int LoadBackbone (int index); // seulement le squelette
  // Place _BuildConfo dans la structure VRes sans la modifier
  int Build (const vector<const C_Residu*> & VRes) ;
  int SetIndex (int index);
  int SetMax (int Max);
  int SetPos (int Pos);
  int SetDepPos (int DepPos);
  int SetConfoSet (C_ConfoSet * CS);
  int SetResiduConstraint (C_ResiduConstraint & RC);
  // Operateurs ++++
  C_ConfoBuild & operator= (const C_ConfoBuild & X);
  int operator== (const C_ConfoBuild & X) const;
  int Empty (void) const;
  // test si _BuildConfo satisfait les contraintes dans _RC
  int SatisfyConstraint (const C_Residu & X) const;
  int SatisfyConstraint (const vector<const C_Residu*> & VRes) const;
  // Acces au contenu +++++
  const C_ConfoSet * GetConfoSet (void);
  const C_Residu * GetBuiltConfo (void);
  int GetMax (void) const;
  int GetPos (void) const;
  int GetDepPos(void) const;
  C_ResiduConstraint & GetResiduContrainte (void);
};
```

FIGURE 29. Spécifications pour la classe `C_ConfoBuild`. Cette classe insère une conformation de résidu dans une structure 3D.

5.3 La transformation unitaire homogène

Cette composante du système est représentée dans la classe `C_Transfo` (Figure 30). Cette classe contient les neuf valeurs de la matrice de transformation. Des instances de `C_Transfo` se retrouvent dans la classe représentant les descripteurs géométriques de fragments (DGF).

```
class C_Transfo {
private:
  friend class C_Point3D;
public:
  // ************* Contenu **********************
  // ELEM contient la matrice de rotation de dimension 3
  // et le vecteur de translation par rapport au systeme d'axe absolue.
  // Ce qui donne une matrice 4 par 3.
  float elem[4][3];
  // ************** Methodes ***********************
  // Constructeur ++++
  C_Transfo( float a00, float a01, float a02,
     float a10, float a11, float a12,
     float a20, float a21, float a22,
     float a30, float a31, float a32 );
  // Modificateurs ++++++
  C_Transfo & SetToIdentity (void);
  C_Transfo & Rot (float angle,const C_Point3D* axis);
  C_Transfo & Translate (float x,float y,float z);
  C_Transfo & InverseOrtho (void);
  C_Transfo & Align(const C_Point3D * p1,const C_Point3D * p2,
    const C_Point3D * p3 );
  // Retourne la TUH entre (p1,p2,p3) et (p4,p5,p6)
  C_Transfo & Referentiel(const C_Point3D * p1,const C_Point3D * p2,
  const C_Point3D * p3,const C_Point3D * p4,
  const C_Point3D * p5,const C_Point3D * p6);
  // Retourne la TUH qui ramene le system d'axe (p1,p2,p3)
  // dans le referentiel absolue et applique la TUH de cet objet.
  C_Transfo & ApplyTransfo ( const C_Point3D *p1,const C_Point3D *p2,
     const C_Point3D *p3 );
  // Retourne la TUH qui place les axes (N2,C2,CA2)
  // par rapport aux axes (CA1,C1,O1) en applicant la TUH de cet objet.
  // Construction du residu i : C[i-1]-N[i].
  C_Transfo & ConnectConfo (const C_Point3D * C1,const C_Point3D * CA1,
    const C_Point3D * O1,const C_Point3D * N2,
    const C_Point3D * CA2,const C_Point3D * C2,
    float connecting_phi);
  // OPERATEURS ++++++
  C_Transfo & operator= (const C_Transfo & T);
  int operator== (const C_Transfo & T) const;
  C_Transfo operator* (const C_Transfo & T) const ;
  C_Transfo operator- (const C_Transfo & T) const ;
  C_Transfo operator+ (const C_Transfo & T) const ;
  float GetTranslation (void) const ;
  float RMS_align (const vector< const C_Point3D* > & To,
   const vector< const C_Point3D* > & From );
  // Metrique MTUH
  float Distance (const C_Transfo & T) const ;
  float DistanceRot (const C_Transfo & T) const ;
  float DistanceTrans (const C_Transfo & T) const ;
  // Lecture et ecriture +++++++
  friend ostream& operator<< (ostream& out,const C_Transfo & T);
  friend istream& operator>> (istream& in,C_Transfo & T);
};
```

FIGURE 30. Spécifications pour la classe `C_Transfo`. Cette classe contient la matrice de transformation unitaire, les méthodes associées et la métrique MTUH.

5.4 Le descripteur géométrique de fragments (DGF)

Le DGF est une structure de données pour l'information contenue dans un fichier PDB. L'utilisation du DGF facilite la manipulation des fragments avec les classes `C_DGF`, `C_DGFSet` et `C_DGFBuild`. La classe `C_DGF` est présentée à la Figure 31. L'information géométrique concernant un fragment est décrite dans cette classe. Nous avons utilisé cette classe pour construire la base de données de dimères. Toutes les instances de cette classe passent par une seule instance globale de la classe `C_GCDGF` (Figure 32). Cette classe s'occupe de la gestion dynamique de la mémoire des objets `C_DGF`.

La classe `C_DGFSet` (voir Figure 33) est l'interface avec les bases de données de DGF. Le système possède une instance de la classe `C_DGF` par dimère lors de la construction d'une boucle (incluant les points d'ancrage).

La classe `C_DGFBuild` (Figure 34) permet de placer un fragment de la banque de DGF dans une structure de protéine. La méthode Build prend comme argument la structure 3D, sous forme d'un vecteur de `C_Residu`, et le point de départ de la construction. Cette méthode va construire le fragment et le sauvegarder dans le vecteur `_VBC` sans modifier le contenu de la structure 3D passée en argument. On peut ensuite vérifier si le fragment satisfait les contraintes dans le vecteur `_VRC`, dans quel cas le contenu de `_VBC` est copié dans un vecteur décrivant la structure. Les résidus du fragment sont construits à l'aide des équations 3.2 et 3.5.

```
class C_DGF {
public:
  // ***************** Contenu ***********************
  long _FilePos;
  vector<vector<int> > _VCIndex; // liste des index de la BD de confo
  vector<vector<char> > _VCFile; // liste des fichiers de la BD de confo
  vector<char> _Seq; // sequence du fragment
  vector<char> _VSS; // structure secondaire du fragment
  vector<int> _VtfoPos1; // index relatif du premier system d'axes
  vector<int> _VtfoPos2; // index relatif du deuxieme system d'axes
  vector<int> _VBuild; // index relatif des residus construits
  vector<C_Transfo> _Vtfo; // liste des TUH
  vector<vector<int> > _VAxe; // liste des deux systemes d'axes de chaque TUH
  char _PdbFile[15]; // provenance du fragment
  // ***************** Methodes ***********************
  // Constructeur +++++
  C_DGF (ifstream &IN) ;
  C_DGF (const char *DataBase_file,long FilePos) ;
  // Modificateurs +++++
  int SetPdbFile (const char * file);
  int SetSeq (const vector<char> Seq);
  int SetSS (const vector<char> SS);
  int SetTfo (const vector<C_Transfo> vtfo);
  int SetPos1 (const vector<int> VPos1);
  int SetPos2 (const vector<int> VPos2);
  int Reset (ifstream& in) ;
  int Reset (const char *DataBase_file,long FilePos) ;
  // Operateurs +++++
  C_DGF& operator= (const C_DGF & X);
  int operator== (const C_DGF & X) const;
  // Acces au contenu +++++
  int Size (void) const;
  const char* GetPdbFile (void);
  char GetSeq (int index);
  char GetSS (int index);
  const C_Transfo * GetTfo (int index);
  int GetPos1 (int index);
  int GetPos2 (int index);
  // Lecture et ecriture ++++
  int Print (ostream& out) const;
  int Read (ifstream & in);
  int Write (ostream & out);
};
```

FIGURE 31. Spécifications pour la classe C_DGF. Cette classe contient l'information pour décrire un fragment 3D de protéine.

```
class C_GCDGF {
public:
  // *************** Contenu  **************
  vector < C_DGF* > _VP; // banque de DGF
  // ***************** Methodes ************
  // Constructeurs ++++
  C_GCDGF( int nombre_en_memoire );
  // Modificateurs ++++
  C_DGF * Pop (void) ;
  int ReSize (int n);
  void Collect (C_DGF * X) ;
  void Flush (void) ;
  // Acces au contenu +++++
  int Size (void) const { return _VP.size(); };
};
```

FIGURE 32. Spécifications pour la classe C_GCDGF. Il existe un seul objet de cette classe dans le système qui s'occupe d'allouer et de recycler tous les objets C_DGF.

```
class C_DGFSet {
private:
  // ******** Contenu a acces limite ******************
  int _NextLoadPos;
  char * _FileName; // la BD
  ifstream * _FilePointer;
  vector < long > _DGFIndex;
  vector<vector<char> > _VSeq,_VSS; // la sequence et sa structure secondaire
  vector < int > _Index2IndexLoaded;
  vector < int > _IndexLoaded2Index;
  int _Limit;
  vector < C_DGF * > _DGF; // les DGF
public:
  // ******** Methodes ***********
  // Constructeurs +++++
  C_DGFSet (const char * filename,int limit) ;
  // Modificateurs +++++
  int Reset (const char * filename,int limit) ;
  int ResetFilePointer (void) ;
  int SetFileName (const char *name) ;
  int Load (int from,int to) ;
  int Load (void) ;
  int LoadIndex (void) ; // lit seulement les indexes dans le fichier
  int ReadIndexFile (const char *name) ;
  int Resize (int limit); // le nombre de DGF en memoire
  // Operateurs ++++
  C_DGFSet& operator= (const C_DGFSet &X);
  int operator== (const C_DGFSet &X) const;
  const C_DGF * operator[] (int index) ;
  int Empty (void) const;
  int IsIn (int index) const;
  int IsLoaded (int index) const;
  // Acces au contenu ++++
  int Size (void) const ;
  int LoadedSize (void) const ;
  ifstream * GetFilePointer (void) ;
  const char * GetFileName (void) const ;
  // Retourne une liste ordonnee des DGF etant donnee une metrique
  int GetSorted (vector<int> & Vindex,
 float Metric(const C_DGF &,const C_DGF &) ) ;
  // Impression ++++
  int Print (ostream& out) ;
  int PrintIndex (ostream& out) const ;
  int PrintLoaded (ostream& out) const ;
  int PrintLoadedIndex (ostream& out) const ;
};
```

FIGURE 33. Spécifications pour la classe `C_DGFSet`. Cette classe gère un ensemble de DGF.

```
class C_DGFBuild {
public:
  // ***************** Contenu ******************
  // pour la sequence de construction
  vector<char> _VSS,_Seq; // la sequence et sa structure secondaire
  vector<int> _VDP,_VP; // position d'appui et position a construire
  // pour la base de donnees
  int _Max,_Index;
  C_DGFSet * _DGFSet; // BD de DGF
  C_DGF _DGF; // le DGF
  vector<C_Confo*> _VBC; // fragment construit
  vector<C_ResiduConstraint> _VRC; // contrainte
  // *************** Methodes ************************
  // Constructeurs +++++
  C_DGFBuild (void);
  // Modificateurs ++++
  int Reset (void) ;
  int SetSeq (const vector<char> & Seq);
  int SetSS (const vector<char> & SS);
  int SetIndex (int index) ;
  int SetDepPos (const vector<int> VDP);
  int SetPos (const vector<int> VP);
  int SetDGFSet (C_DGFSet * DGFSet);
  int SetRC (C_ResiduConstraint & RC);
  int Build (int index_depart,vector<const C_Residu*> & VRes) ;
  int BuildBackbone (int index_depart,vector<const C_Residu*> & VRes) ;
  int BuildNAdd (int index_depart,vector<const C_Residu*> & VRes) ;
  int BuildNAddBackbone (int index_depart,vector<const C_Residu*> & VRes) ;
  // Operateurs +++++
  C_DGFBuild & operator= (const C_DGFBuild & X);
  int operator== (const C_DGFBuild & X) const;
  int SatisfyConstraint (void) const;
  int Empty (void) const;
  // Acces au contenu +++++
  vector<const C_Residu *> & GetFragment (void);
};
```

FIGURE 34. Spécifications pour la classe C_DGFBuild. Cette classe construit et ajoute des fragments à une structure 3D.

5.5 La contrainte spatiale et conditionnelle

La Figure 35 présente les spécifications pour la classe `C_AtomConstraint` qui sert à emmagasiner l'information des atomes et la distance entre ces derniers. La fonction `Satisfy` permet de vérifier si deux atomes satisfont la contrainte spatiale contenue dans l'objet `C_AtomConstraint`.

```
class C_AtomConstraint {
private:
  // ************* Contenu a acces restreint ************
  float _lower,_upper;
  int _atom1,_atom2;
public:
  // ************* Methodes *****************
  // Constructeurs ++++
  C_AtomConstraint(int atom1,int atom2,float d1,float d2) ;
  // Modificateurs ++++
  int Reset(int atom1,int atom2,float d1,float d2) ;
  int SetAtoms(int atom1,int atom2) ;
  int SetRange(float d1,float d2) ;
  // operateurs +++++
  C_AtomConstraint& operator= (const C_AtomConstraint &X) ;
  int operator== (const C_DistanceConstraint &X) const;
  int IsIn(float d) const ;
  int IsIn(int atom) const;
  int Satisfy(const C_Atom &X,const C_Atom &Y) const;
  int Empty(void) const ;
  // Acces au contenu ++++++
  int GetAtom1(void) const;
  int GetAtom2(void) const;
  float GetLower(void) const;
  float GetUpper(void) const;
};
```

FIGURE 35. Spécifications pour la classe `C_AtomConstraint`.

```
class C_ConditionConstraint {
public:
  // ********** Contenu **********
  vector<C_AtomConstraint> _VCons,_VCond;
  // ********** Methodes **********
  // Constructeurs ++++
  C_ConditionConstraint (const C_ConditionConstraint & X);
  // Modificateurs +++++
  int Reset (void);
  int AddCond (const C_AtomConstraint & AC) ; // condition
  int AddCons (const C_AtomConstraint & AC) ; // contrainte
  // operateurs +++++
  C_ConditionConstraint & operator= (const C_ConditionConstraint & X);
  int operator== (const C_ConditionConstraint & X) const;
  int Empty() const;
  int IsInCond (int atom,int & index) const;
  int IsInCons (int atom,int & index) const;
  int IsInCond (const C_AtomConstraint & AC,int & index) const;
  int IsInCons (const C_AtomConstraint & AC,int & index) const;
  // Acces au contenu ++++
  int GetSizeCond (void);
  int GetSizeCons (void);
  int GetAtomCond (int index);
  int GetAtomCons (int index);
};
```

FIGURE 36. Spécifications pour la classe `C_ConditionConstraint`.

La Figure 36 présente les spécifications pour la classe `C_ConditionConstraint` qui sert à emmagasiner l'information de la liste des objets `C_AtomConstraint` formant les conditions et la liste des objets `C_AtomConstraint` formant les contraintes spatiales. La Figure 37 présente les spécifications pour la classe `C_ResiduConstraint` qui sert à emmagasiner l'information sur la

```
class C_ResiduConstraint {
public:
  // ************* Contenu ************************
  C_ConditionConstraint _Global; // contrainte global a tous les atomes
  vector<int> _VPos; // la position des autres residus
  vector<C_ConditionConstraint> _VSpecific; // contrainte specifique
  float _VdW; // distance de Van der Waal
  // ************** Methodes **********************
  // Constructeurs ++++
  C_ResiduConstraint (const C_ResiduConstraint& X) ;
  // Modificateurs ++++
  int SetGlobal (const C_ConditionConstraint & CC);
  int AddSpecific (const C_ConditionConstraint & CC,int pos);
  // operateurs ++++
  C_ResiduConstraint & operator= (const C_ResiduConstraint &X) ;
  int operator== (const C_ResiduConstraint &X) const;
  int Empty (void) const;
  int IsIn (int Pos,int & index) const;
  int SatisfyGlobal (const C_Residu & Ref,const C_Residu & R2) const;
  int SatisfySpecific (const C_Residu & Ref,const C_Residu & R2) const;
  int Satisfy (const C_Residu & Ref,const C_Residu & R2) const;
  // Acces au contenu ++++
  int Size(void) const;
  int GetPos (int index);
  C_ConditionConstraint & GetSpecific (int index);
  C_ConditionConstraint & GetGlobal (void);
};
```

FIGURE 37. Spécifications pour la classe C_ResiduConstraint.

liste des contraintes conditionnelles de résidus, où on associe un objet C_ResiduConstraint à chaque résidu.

CHAPITRE 6

Les résultats

Dans ce chapitre nous présentons les résultats obtenus avec notre système de construction *LoopBuilder* pour la crambine (1CBN) et la phosphotransférase porteur d'histidine (2HPR) que nous comparerons avec d'autres méthodes. Nous verrons les conditions initiales, les banques de DGF et les fonctions STEP utilisées.

6.1 Conditions initiales

La construction de boucles dépend d'un ensemble de paramètres initiaux, telles que la banque de DGF (BDDGF) et des distances empiriques interatomiques du squelette des boucles (voir Tableau VI). Des intervalles [min,max] de distances telles qu'obsvervées dans les structures expérimentales servent de contraintes pour la construction des boucles, de sorte que tous les modèles ne satisfaisant pas ces contraintes soient rejetés.

La PDBSelect25 contient 2 700 boucles dont la longueur varie entre 1 et 20 résidus. Le Tableau VII présente la distribution des boucles de longueur 1 à 8 de la PDBSelect25: 39% des boucles relient deux brins-β, 43% relient un brin-β et une hélice-α et 19% relient deux hélices-α.

La banque de DGF de dimères possède 400 combinaisons de paires d'acides aminés, le tableau VIII présente leur distribution. La BDDGF contient environ 40

TABLEAU VI. Distances entre différents atomes du squelette dans les boucles des structures de la PDBSelect25.

Atome résidu[i]	Atome résidu[$i+n$]	n	moyenne	écart-type	minimum	maximum
C	C	1	3.2	0.2	2.4	4.1
C	C	2	5.7	0.7	3.0	8.2
C	C	3	7.7	1.5	3.6	11.4
C	C	4	9.6	2.2	4.1	14.2
C	C	5	11.0	2.9	3.9	17.6
C	C	6	12.5	3.5	4.2	20.9
C	C	7	13.5	4.0	3.9	23.7
Cα	Cα	1	3.8	0.06	2.7	4.0
Cα	Cα	2	6.1	0.6	4.0	7.6
Cα	Cα	3	7.7	2.0	4.0	10.0
O	N	1	2.25	0.03	2.0	2.5
C	N	1	1.32	0.02	1.2	1.4

TABLEAU VII. Distribution des boucles de la PDBSelect25 selon leur longueur et type de ESS.

ESS1-ESS2	Longueur de la boucle							
	1	2	3	4	5	6	7	8
brin-β-brin-β	27	164	104	118	148	72	69	81
hélice-α-brin-β	20	43	81	80	65	77	48	23
brin-β-hélice-α	85	67	73	55	55	35	25	20
hélice-α-hélice-α	76	39	52	58	61	41	38	24
Total	208	313	310	311	329	225	180	148

TABLEAU VIII. Distribution des DGF de la BDDGF étant donné un score de similarité calculé avec la MTUH.

Score de similarité (MTUH)	Nombre de DGF
1.0	2900
0.5	6700
0.3	27800
0.1	40000

TABLEAU IX. Distribution des DGF de dimères les plus nombreux de la base de données BDDGF.

Séquence des dimères	Nombre de DGF
GG	232
SG	189
AG	168
GS	165
DG	159

000 fragments dont 2 600 possèdent plus d'une instance dans PDBSelect25. Dans ce cas, deux fragments sont considérés similaires si le score de la MTUH est plus petit ou égal à 0.1. Un seuil de 0.1 représente une valeur de RMSD d'environ 0.05 entre les six atomes du fragment. Lorsque le score de similarité est égal à 1.0, les DGF de la BDDGF se regroupent en 2 900 fragments. Le nombre de fragments croît lorsque le score de similarité décroît.

Le tableau IX représente la distribution des fragments les plus nombreux de la base de données de dimères. Tous les fragments de la BDDGF sont distants d'au moins 0.1 selon notre mesure MTUH.

Serait-il plus avantageux d'utiliser des DGF de trimères ?
Le nombre de DGF de trimères que nous avons extrait est d'environ 31000 dont 435 possèdent plus d'une instance. La BDDGF de dimères possède environ 100 fragments pour chaque dimère. Par contre, la BDDGF de trimères possède seulement environ 4 fragments pour chaque trimère. La différence justifie le choix d'une BDDGF de dimères car il manque de fragments pour décrire tous les trimères.

Comme pour tout système empirique, notre système est basé sur le fait que l'algorithme de construction utilisant notre BDDGF soit capable de construire n'importe quelle boucle de structures faisant partie de l'ensemble de départ. La BDDGF de dimères satisfait ce postulat selon le critère *JackKnife* [29]. La BDDGF de dimères est composée de toutes les structures de boucles de la PDB-Select25 sauf celle à modéliser. Par exemple, nous allons prendre la boucle [36-38] de la crambine et la re-construire à l'aide de la BDDGF sans la crambine. Si la BDDGF incluait les fragments dans la boucle [36-38] de la crambine alors le meilleur modèle construit avec ces fragments aurait un score de fermeture et une RMSD de zéro.

Les modèles générés contiennent des fermetures inférieures à 0.4 et des RMSD avec la structure cristalline inférieures à 0.8 Å. La Figure 38 contient les valeurs RMSD de ces modèles en fonction de leur fermeture. Ce diagramme est représentatif des ensembles solutions de notre système pour toutes les boucles de longueur 3.

6.2 Les fonctions Step

La fonction StepPart permet d'évaluer l'effet du pourcentage exploré sur la qualité des modèles construits. Le tableau X contient les résultats obtenus avec la fonction StepPart sur deux boucles de la protéine 2HPR, la phosphotransférase porteur de l'histidine possèdant six boucles: [9-15], [27-31], [38-39], [44-46], [52-59]

TABLEAU X. RMSD des trois meilleurs modèles proposés selon leur score de fermeture, pour deux boucles de la protéine 2HPR. La fonction StepPart est utilisée et les modèles possèdent tous un score de fermeture inférieur ou égal à 0.4 MTUH.

Boucles	RMSD 1 (Å) (fermeture MTUH)	RMSD 2 (Å) (fermeture MTUH)	RMSD 3 (Å) (fermeture MTUH)	% des noeuds visités
28-32	1.957 (2.086)	2.07 (3.114)	-	5%
28-32	2.477 (0.605)	2.186 (0.682)	2.025 (0.891)	10%
28-32	1.077 (0.216)	2.241 (0.261)	1.379 (0.318)	20%
28-32	1.292 (0.102)	1.077 (0.216)	1.141 (0.255)	30%
28-32	1.292 (0.102)	1.545 (0.144)	2.115 (0.166)	50%
67-69	-	-	-	5%
67-69	-	-	-	10%
67-69	-	-	-	20%
67-69	0.540 (0.237)	0.676 (0.272)	0.418 (0.301)	40%
67-69	0.893 (0.160)	0.476 (0.190)	0.540 (0.237)	60%
67-69	0.893 (0.160)	0.476 (0.190)	0.567 (0.190)	80%
67-69	0.432 (0.154)	0.893 (0.160)	0.476 (0.190)	100%

et [67-69].

Pour la boucle [28-32], une exploration de 20% des noeuds permet d'obtenir un modèle à 1.077Å de RMSD avec le cristal avec un score de fermeture de 0.216 (mesure de MTUH). Si le nombre de noeuds est augmenté à 30%, un modèle à 0.102 de fermeture est trouvé mais la RMSD de ce modèle est moins bonne que la meilleure solution trouvée avec 20%. Si on augmente l'exploration à 50% on trouve trois modèles dont la fermeture est inférieure ou égale à 0.166. Par contre, le meilleur modèle du point de vue de la RMSD n'est plus parmi les trois meilleurs du point de vue de la fermeture. Nos résultats dénotent que notre méthode génère un ensemble de modèles dont les meilleurs en RMSD ne sont pas nécessairement

les meilleurs scores de fermeture. Cependant, s'il existe un modèle semblable au cristal parmi l'ensemble des modèles plausibles, alors son score de fermeture sera le meilleur. Le score de fermeture est un critère de sélection raisonnable.

Les valeurs de RMSD des modèles obtenues pour la boucle [67-69], quant à elles, s'améliorent lorsque le score de fermeture diminue. Encore une fois, plus on explore de noeuds et plus le nombre de modèles avec une bonne fermeture augmente.

La relation fermeture versus RMSD est illustrée à la Figure 38. Les meilleurs modèles en RMSD possèdent aussi les meilleurs scores de fermeture. Lorsque le score de fermeture dépasse 0.5, des modèles de mauvaises RMSD apparaissent. Les modèles dont le score de fermeture augmente représentent des boucles qui s'éloignent de leur deuxième point d'ancrage.

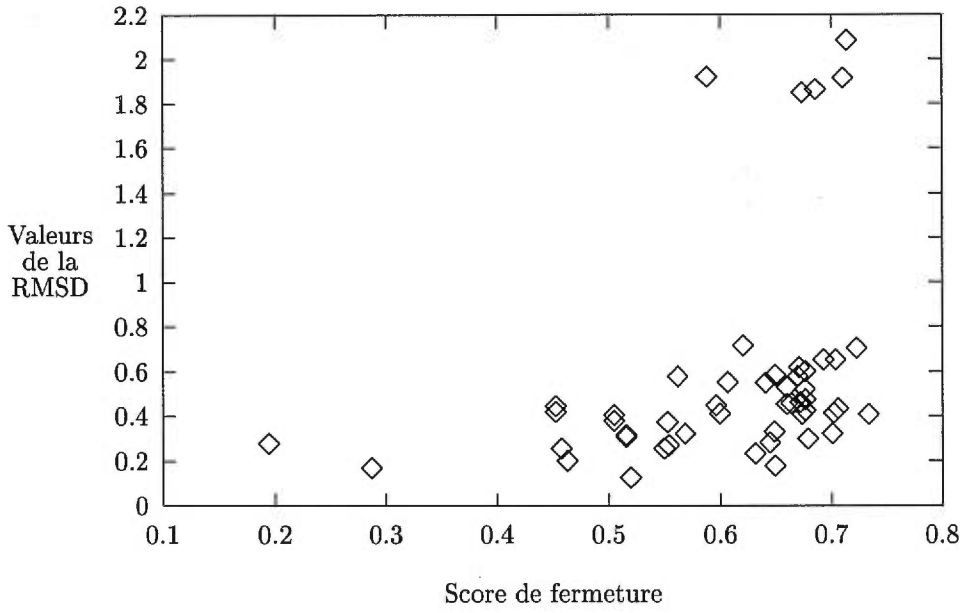

FIGURE 38. Distribution des valeurs de RMSD versus les scores de fermeture pour la boucle [36-38] de la crambine (1CBN).

6.3 Étude de la crambine (1CBN)

La crambine est une protéine provenant des graines de chou abyssinien (massif Éthiopien) dont le code est 1CBN [32]. Cette protéine possède 5 boucles: [5-6], [20-22], [31-31], [36-38] et [42-45]. La boucle [42-45] implique un arbre de recherche avec fouille exhaustive de $323 \times 262 \times 108 \times 98 \times 163 \approx 1.4 \times 10^{11}$ noeuds de pré-sélection et une possibilité de construire $323 \times 262 \times 108 \times 98 \approx 8.5 \times 10^{8}$ structures.

La distribution des DGF de la BDDGF utilisée pour construire la boucle [42-45] est présentée dans le tableau XI. L'arbre de recherche contient environ 1.4×10^{11} noeuds dans le pire cas où tous les fragments de fermeture ne satisfont pas la contrainte de fermeture, et environ 8.5×10^{8} noeuds dans le meilleur cas où le premier fragment de fermeture de la BDDGF(ALA-ASN) satisfait la contrainte de fermeture. La Figure 39 contient une superposition de 9 modèles obtenus par

TABLEAU XI. Banque de données de DGF utilisée pour la boucle[42-45] de la crambine (1cbn).

Résidu	Type de région	DGF	Nombre d'enregistrement
PRO	brin-β		
GLY	boucle	PRO-GLY	323
ASP	boucle	GLY-ASP	262
TYR	boucle	ASP-TYR	108
ALA	boucle	TYR-ALA	98
ASN	brin-β	ALA-ASN[b]	163

[b]DGF utilisé pour évaluer la fermeture de la boucle

une recherche aléatoire. Il existe deux groupes de modèles: $\psi = 365^0$ et $\psi = 230^0$ pour GLY à la position 42. Parmis les 323 DGF PRO-GLY pour la position 42,

la fouille aléatoire retourne seulement deux classes de DGF qui permettent de construire la boucle.

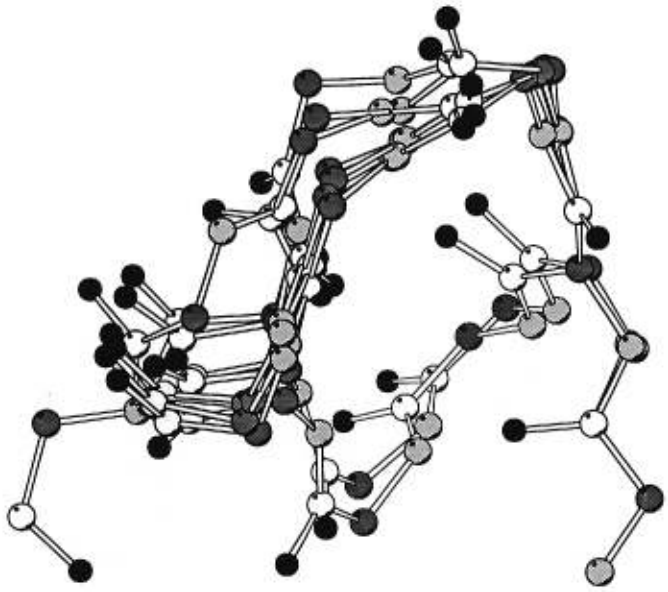

FIGURE 39. Modèles de la boucle[42 – 45] de la crambine (1cbn). Les couleurs des atomes : O = noir, C_α = gris fonçé, N = gris pâle, C = blanc. La boucle est construite de N vers C terminal. Les deux résidus terminaux repésentent les points d'ancrage.

La meilleure solution a une RMSD de 0.489Å. A partir de notre BDDGF des dimères, on peut reconstruire la conformation de la boucle du cristal pour toutes les boucles de la crambine sauf la [42-45] dont l'exploration exhaustive de l'espace des conformations est trop coûteuse (une boucle à trois résidus prend 15 minutes alors qu'une boucle à quatre prend trois jours sur un processeur R10000 à 180 Mhz de SGI).

TABLEAU XII. RMSD des modèles des boucles de la crambine (1cbn) avec score de fermeture inférieur ou égale à 0.4 MTUH.

Boucles	moyenne (Å)	écart-type (Å)	minimum	maximum
5-6	0.16	0.04	0.10	0.23
20-22	0.39	0.15	0.13	0.7
31-31	0.162	-	-	-
36-38	0.29	0.15	0.15	0.586
42-45	0.489	-	-	-

6.4 Étude de la phosphotransférase porteur de l'histidine (2HPR)

La phosphotransférase porteur de l'histidine (2HPR) est une protéine provenant de la bactérie *escherichia coli*. Cette protéine contient 88 résidus et fait partie du système de transport des sucres dans la bactérie. La protéine contient 6 boucles: [9-15], [27-31], [38-29], [44-46], [52-59] et [67-69].

La distribution des DGF de la BDDGF utilisée pour construire cette boucle est présentée au Tableau XIII. L'arbre de recherche possède environ 6.6×10^9 noeuds dans le pire cas et 3.0×10^7 noeuds dans le meilleur cas (voir l'exemple précédent pour la description du meilleur et pire cas).

Le tableau XIV contient les valeurs de RMSD pour les trois meilleurs modèles sélectionnés avec leur score de fermeture.

TABLEAU XIII. Distribution des DGF utilisés pour la boucle [67-69] de la phosphotranférase (2HPR).

Résidu	Type de région	DGF	Nombre d'enregistrement
SER	brin-β		
GLY	boucle	SER-GLY	410
ALA	boucle	GLY-ALA	319
ASP	boucle	ALA-ASP	232
GLU	hélice-α	ALA-GLU[a]	217

[a]DGF utilisé pour évaluer la fermeture de la boucle

TABLEAU XIV. RMSD des trois meilleurs modèles proposés selon leur score de fermeture pour chaque boucle de la phosphotransférase (2HPR).

Boucles	RMSD 1 (Å) (fermeture MTUH)	RMSD 2 (Å) (fermeture MTUH)	RMSD 3 (Å) (fermeture MTUH)	% des noeuds visités
9-15	2.23 (0.235)	3.226 (0.278)	3.557 (0.319)	10%
27-31	0.970 (0.297)	2.00 (0.319)	0.992 (0.319)	30%
38-39	0.523 (0.267)	0.595 (0.352)	0.639 (0.391)	100%
44-46	0.803 (0.165)	0.686 (0.173)	0.663 (0.241)	100%
52-59	3.672 (0.439)	5.257 (0.447)	5.137 (0.562)	5%
67-69	0.432 (0.154)	0.893 (0.160)	0.476 (0.190)	100%

6.5 Comparaisons avec les autres méthodes

Le Tableau XV contient les valeurs de RMSD des modèles obtenus par différentes méthodes. Ce tableau contient les RMSD de modèles choisis avec le score de fermeture et des meilleurs modèles obtenus avec notre méthode. Par

TABLEAU XV. RMSD des modèles de boucles pour différentes méthodes. Région[1] = région modélisée par notre système; Région[2] = région modélisée par les autres méthodes; RMSD[1] = Meilleur modèle sélectionné par score de fermeture; RMSD[2] = Meilleur modèle sélectionné par RMSD; RMSD[3] = Meilleur modèle des autres méthodes.

Code PDB	Région[1]	RMSD[1]	RMSD[2]	% BDDGF	Région[2]	RMSD[3]
2HPR	27-31	0.97	0.495	30	27-32	0.06[b]
2HPR	67-69	0.432	0.238	100	66-72	0.19[b]
2PCY	33-35	2.87	0.51	100	80-84	0.42[a]
2CCY	31-34	0.955	0.955	30	31-39	1.63[a]
1UBQ	18-22	0.765	0.765	30	19-24	0.33[b]
1UBQ	51-55	0.468[c]	-	-	51-57	3.72[b]
8PTI	8-17	6.45	6.45	5	13-19	1.45[b]
8PTI	36-44	-	-	-	33-39	2.64[b]

[a]Sudarsanam *et al.* [35]
[b]Rose *et al.* [36]
[c]Avec SetRand(30%).

exemple, le meilleur modèle de la boucle [27-31] de la protéine 2HPR possède une RMSD de 0.495Å. Le modèle obtenu par Rose *et al* [36] possède une RMSD de 0.06Å. Le calcul de Rose utilise la dynamique moléculaire, ce qui est la méthode la plus précise des méthodes de construction dans ce cas.

Noter que notre meilleur modèle est très près de celui de Rose. Le meilleur

modèle de la boucle [67-69] de la protéine 2HPR possède une RMSD de 0.238Å. Rose *et al.* ont obtenu un modèle avec une RMSD de 0.19Å.

La Figure 40 représente la distribution des RMSD de nos meilleurs modèles, c'est-à-dire qui ont un score de fermeture inférieur à 0.4 mesuré avec MTUH. Pour

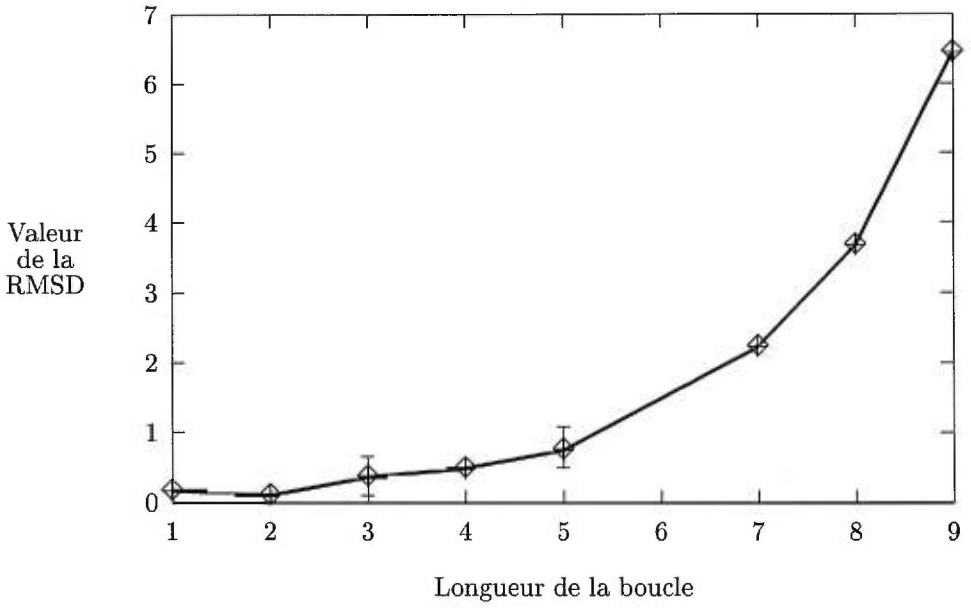

FIGURE 40. Distribution des meilleures valeurs de la fonction RMSD pour des modèles de boucles dont le score de fermeture est inférieur à 0.4 MTUH. Les résultats de boucles plus longues que 4 résidus proviennent de sous-ensembles de la BDDGF car une recherche exhaustive devient trop longue à calculer. Une boucle à deux résidu prend 10 secondes, à trois résidus prend 15 minutes alors qu'une boucle à quatre prend trois jours sur un processeur R10000 à 180 Mhz de SGI.

les boucles de longueurs inférieures à six, nos modèles sont proches des cristaux. Les résultats pour les boucles de longueurs supérieures à cinq résidus proviennent de calculs n'utilisant qu'une faible fraction de la BDDGF; StepPart(5% à 20%) ou StepRand.

CHAPITRE 7

La conclusion

Notre motivation pour développer une méthode de modélisation des boucles est de compléter les structures noyaux générées par le programme MC-SYM dans la cadre de la prédiction *de novo* de structures 3D de protéines.

Nous avons traduit le problème de prédiction de la structure 3D d'une boucle en un problème de satisfaction de contraintes spatiales. La construction est réalisée par une fouille en profondeur d'un arbre de structures possibles à la recherche de celles qui satisfont un ensemble de contraintes spatiales. L'arbre de recherche est parcouru par un algorithme de retour-arrière. Les niveaux de l'arbre représentent un nouveau fragment à ajouter à une structure partielle. Si une contrainte n'est pas satisfaite, la branche de l'arbre qui contient le noeud incohérent est élaguée et toutes les structures du sous-arbre ayant ce noeud comme racine sont éliminées. Une fouille qui se rend à une feuille de l'arbre résulte en un modèle complet de la boucle. Cette approche a été implantée dans le programme LoopBuilder.

Le facteur de branchement de l'arbre de recherche dépend de la taille des banques de fragments utilisées. Nous avons introduit le concept du *descripteur géométrique de fragments* (DGF) défini par sa séquence et ses caractéristiques 3D permettant de représenter n'importe quel fragment 3D de la PDB.

L'algorithme de construction des boucles utilise des DGF composés de deux résidus adjacents (dimères) dont la TUH est formée par la position des atomes

($C_\alpha[i]$,C[i],O[i],$C_\alpha[i+1]$,C[$i+1$],O[$i+1$]). Pour chaque dimère nous avons défini une BDDGF et une discrétisation de son espace de conformations. Un algorithme de pré-sélection est appliqué aux BDDGF afin de limiter la taille de l'arbre de recherche. Chaque boucle se lie à deux ESS qui incluent deux résidus de contact que nous avons appelés *points d'ancrage.*

L'algorithme construit une boucle en ajoutant un résidu à la fois, les uns à la suite des autres afin de relier les points d'ancrage. Le résidu i est placé à partir du résidu $i-1$, de la première vers la deuxième ESS. Le dernier résidu de la boucle se positionne par rapport au résidu suivant afin de satisfaire la *contrainte de fermeture.* La contrainte de fermeture est vérifiée avant la construction de la boucle en multipliant les TUH de chaque DGF ce qui permet d'accélérer la recherche des boucles possibles. Une fois la pré-selection des DGF terminée, un algorithme de construction est appliqué afin de générer l'ensemble des modèles possibles.

Un ensemble de contraintes spatiales, telle que la contrainte de fermeture peuvent être spécifiées. Des contraintes déduites des structures de la PDBselect25 sont utilisées. Le programme LoopBuilder complète les structures noyaux générées par MC-SYM. Les modèles sont sélectionnés à partir de leur score de fermeture de la boucle obtenu avec la métrique MTUH.

Trois méthodes de parcours de l'arbre de recherche sont utilisées: exhaustive, partielle et aléatoire. La méthode est choisie selon la longueur de la boucle à modéliser. Nos résutats indiquent que notre méthode est dans le même ordre de précision que les méthodes existantes mais plus efficace. Il serait intéressant d'appliquer la dynamique moléculaire pour raffiner nos modèles.

Sudarsanam *et al.* proposent d'utiliser des trimères pour réduire l'espace de recherche mais nous avons démontré (expérience *JackKnife* [29]) que l'ensemble PDBSelect25 n'est pas assez grand pour permettre de construire une base de

données de DGF représentative. Un autre prolongement de la méthode serait d'ajouter les chaînes latérales qui permettraient, dans certains cas, d'éliminer des modèles par collisions.

RÉFÉRENCES

[1] C. Branden et J. Tooze, *Introduction to protein structure*, Garland Publishing, Inc., New York, (1991).

[2] T. R. Defay et F. E. Cohen, "Protein Modeling", *Encyclopedia of Molecular Biology and Molecular Medecine* **5**, VCH, 158-169 (1996).

[3] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi et M. Tasumi, "The Protein Data Bank: a computer-based archival file for macromolecular structures", *Journal of Molecular Biology* **112**, 535-542 (1977).

[4] E. E. Abola, J. L. Sussman, J. Prilusky, et N. O. Manning, "Protein Data Bank Archives of Three-Dimensional Macromolecular Structures", *Methods in Enzymology* **277**, Academic Press, San Diego, CA, 556-571 (1997).

[5] E. E. Abola, F. C. Bernstein, S. H. Bryant, T. F. Koetzle, et J. Weng, "Protein Data Bank", *Crystallographic Databases-Information Content, Software Systems, Scientific Applications*, Data Commission of the International Union of Crystallography, Bonn/Cambridge/Chester, 107-132 (1987).

[6] M. Parisien, M. Peitsh et F.Major, "A protein conformational search space defined by secondary structure contacts", *Proceedings of the 1998 Pacific Symposium on Biocomputing*, World Scientific, Singapure, 425-436 (1998).

[7] D. A. Benson, M. S. Boguski, D. J. Lipman, J. Ostell et B. F. Ouellette, "GenBank", *Nucleic Acids Research* **1**, 1-7 (1998).

[8] F. Major, M. Turcotte, D. Gautheret, G. Lapalme, E. Fillion et R. Cedergren, "The Combination of Symbolic and Numerical Computation for Three-Dimensional Modeling of RNA", *Science* **253**, 1255-1260 (1991).

[9] D. Gautheret, F. Major et R. Cedergren, "Modeling the Three-dimensional Structure of RNA Using Discrete Nucleotide Conformational Sets", *Journal of Molecular Biology* **229**, 1049-1064 (1993).

[10] F. Major, D. Gautheret et R. Cedergren, "Reproducing the Three-dimensional structure of a tRNA molecule from structural constraints", *Proceedings of the National Academy of Sciences* (USA) **90**, 9408-9412 (1993).

[11] R. M. Haralick, "Increasing Tree Search Efficiency for Constraint Satisfaction Problems", *Artif. Intell.* **14**, 263-266 (1980)

[12] P. Y. Chou et G. D. Fasman, "Prediction of Protein Conformation", *Biochemistry* **13**, 222-245 (1974).

[13] C. Chothia et A. M. Lesk, "The relation between the divergence of sequence and structure in proteins", *EMBO J.* **5**, 823-826 (1986).

[14] M. O. Dayhoff, R. M. Schwartz et B. C. Orcutt, "A Model of Evolutionary Change in Proteins", *Atlas of Protein Sequence and Structure* **5**, Nat. Biomed. Res. Found., Washington, D. C., 345-352 (1978).

[15] L. E. Donate, S. D. Rufino, L. H.J. Canard et T. L. Blundell, "Conformational analysis and clustering of short and medium size loops connecting regular secondary structures: A database for modeling and prediction", *Protein Science* **5**, 2600 (1996).

[16] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells et J. M. Thorton, "CATH - a hierarchic classification of protein domain structures", *Struture* **5**, 1093-1108 (1997).

[17] C. A. Orengo, F. M. G. Pearl, J. E. Bray, A. E. Todd, A. C. Martin, L. Lo Conte et J. M. Thorton, "The CATH Database provides insights into protein structure/function x relationships", *Nucleic Acids Research* **27**, 275-279 (1999).

[18] J. Greer, "Comparative Modeling Methods: Application to the Family of the Mammalian Serine Proteases", *Proteins* **7**, 317-334 (1990).

[19] R. Lüthy, I. Xenarios et P. Bucher, "Improving the sensitivity of the sequence profile method", *Protein Science* **3**, 139-146 (1994).

[20] Arthur M. Lesk, "Systematic representation of protein folding patterns", *Journal of Molecular Graphics* **13**, 159-164 (1995).

[21] J. F. LesZcZynski et G. D. Rose, "Loops in Globular Proteins: A Novel Category of Secondary Structure", *Science* **2**, 849 (1986).

[22] R. E. Bruccoleri, E. Haber et J. Novotný, "Structure of antibody hypervariable loops reproduced by a conformational search algorithm", *Nature* **335**, 564-568 (1988).

[23] U. Hobohm, M. Scharf, R. Schneider et C. Sander, "Selection of representative protein data sets", *Protein Science* **1**, 409 (1992).

[24] U. Hobohm et C. Sander, "Enlarged representative set of protein structures", *Protein Science* **3**, 522 (1994).

[25] E. I. Shakhnovich, "Theoretical studies of protein-folding thermodynamics and kinetics", *Current Opinion in Structural Biology* **7**, 29-40 (1997).

[26] C. B. Anfinsen, "Principles that govern the folding of protein chains", *Science* **181**, 223-230 (1973).

[27] R. M. Fine, H. Wang, P. S. Shenkin, D. L. Yarmush et C. Levinthal, "Predicting antibody hypervariable loop conformations. II: Minimization and x molecular dynamics studies of MCPC603 from many randomly generated loop x conformations", *Proteins* **1**, 342-362 (1986).

[28] R. Henderson, J. M. Baldwin, T. A. Ceska, F. Zemlin, E. Beckmann et K. H. Downing, "Model for the structure of bacteriorhodopsin based on high-resolution x electron cryo-microscopy", *Journal of Molecular Biology* **213**, 899-929 (1990).

[29] B. Efron, *The Jack Knife, the bootstrp and other resampling plans.* Society for industrial and applied mathematics, Philadelphia, PA, 19982.

[30] Q. Zhen, R. Rosenfeld, S. Vajda et C. DeLisi, "Loop Closure via Bond Scaling and Relaxation", *Journal of Computational Chemistry* **14**, 556-565 (1993).

[31] P. Herzyk et R. E. Hubbard, "Automated method for modeling seven-helix transmembrane receptors from x experimental data", *Biophysical Journal* **69**, 2419-2442 (1995).

[32] M. M. Teeter, S. M. Roe et N. H. Heo, "Atomic resolution (0.83 A) crystal structure of the hydrophobic protein x crambin at 130 K", *Journal of Molecular Biology* **230**, 292 (1993).

[33] S. D. Rufino, L. E. Donate, L. J. Canard et T. L. Blundell, "Predicting the Conformational Class of Short and Medium Size Loops Connecting Regular Secondary Structures: Application to Comparative Modeling", *Journal of Molecular Biology* **267**, 352 (1997).

[34] Z. Sun et B. Jiang, "Patterns and Conformations of Commonly Occuring Supersecondary Structures (Basic Motif) in Protein Data Bank", *Journal of Protein Chemistry* **15**, 675-690 (1996).

[35] S. Sudarsanam, R. F. Dubose, C. J. March et S. Srinivasan, "Modeling protein loops using a ϕ_{i+1}, ψ_i dimer database", *Protein Science* **4**, 1412-1420 (1995).

[36] D. Rosenbach et R. Rosenfeld, "Simultaneous modeling of multiple loops in proteins", *Protein Science* **4**, 496-505 (1995).

[37] L. Carlacci et W. Englander, "The Loop Problem in Proteins: A Monte Carlo Simulated Annealing Approach", *Biopolymers* **33**, 1271-1286 (1993).

[38] Y. Wang, H. I. Huq, X. F. de la Cruz et B. Lee, "A new procedure for constructing peptides into a given C_α chain", *Folding & Design* **3**, 1-10 (1997).

REMERCIEMENTS

Je remercie mon directeur de recherche, François Major, pour les discussions scientifiques et la liberté d'action qu'il m'a donnée dans ce projet. Je le remercie également pour son aide très appréciée lors de la correction de ce mémoire. Je remercie Dany Dubé pour toutes nos discussions sur la science et Babylon5. Je remercie également Dany Dubé, Patrick Gendron, Sébastien Lemieux, Daniel St-Arnaud pour toutes nos discussions, parfois farfelues, mais toujours appréciées ainsi que nos soirées de billards au Resto Pub. Je me dois de remercier Marc Parisien, Roberto Portillo, Patrick Gendron, Sébastien Lemieux et Daniel St-Arnaud pour nos parties de Doom endiablées. En terminant, je remercie Hélène Juteau pour son support moral (et parfois financier) dans mes deux maîtrises et son aide lors de la correction des deux mémoires.

ANNEXE A

Ficher PDB de la crambine (1CBN)

```
HEADER    PLANT SEED PROTEIN                      11-OCT-91   1CBN       1CBN   2
COMPND    CRAMBIN                                                        1CBN   3
SOURCE    ABYSSINIAN CABBAGE (CRAMBE ABYSSINICA) SEED                    1CBN   4
AUTHOR    M.M.TEETER,S.M.ROE,N.H.HEO                                     1CBN   5
REVDAT   1   31-JAN-94 1CBN     0                                        1CBN   6
JRNL        AUTH   M.M.TEETER,S.M.ROE,N.H.HEO                          1CBN   7
JRNL        TITL   ATOMIC RESOLUTION (0.83 ANGSTROMS) CRYSTAL           1CBN   8
JRNL        TITL 2 STRUCTURE OF THE HYDROPHOBIC PROTEIN CRAMBIN AT     1CBN   9
JRNL        TITL 3 130 K                                               1CBN  10
JRNL        REF    J.MOL.BIOL.                  V. 230   292 1993      1CBN  11
JRNL        REFN   ASTM JMOBAK  UK ISSN 0022-2836                 070   1CBN  12
REMARK   1                                                             1CBN  13
REMARK   1 REFERENCE 1                                                 1CBN  14
REMARK   1  AUTH   H.HOPE                                              1CBN  15
REMARK   1  TITL   CRYOCRYSTALLOGRAPHY OF BIOLOGICAL MACROMOLECULES:   1CBN  16
REMARK   1  TITL 2 A GENERALLY APPLICABLE METHOD                       1CBN  17
REMARK   1  REF    ACTA CRYSTALLOGR.,SECT.B     V.  44    22 1988      1CBN  18
REMARK   1  REFN   ASTM ASBSDK  DK ISSN 0108-7681                 622   1CBN  19
REMARK   1 REFERENCE 2                                                 1CBN  20
REMARK   1  AUTH   M.WHITLOW,M.M.TEETER                                1CBN  21
REMARK   1  TITL   AN EMPIRICAL EXAMINATION OF POTENTIAL ENERGY        1CBN  22
REMARK   1  TITL 2 MINIMIZATION USING THE WELL-DETERMINED STRUCTURE    1CBN  23
REMARK   1  TITL 3 OF THE PROTEIN CRAMBIN                             1CBN  24
REMARK   1  REF    J.AM.CHEM.SOC.               V. 108  7163 1986      1CBN  25
REMARK   1  REFN   ASTM JACSAT  US ISSN 0002-7863                 004   1CBN  26
REMARK   1 REFERENCE 3                                                 1CBN  27
REMARK   1  AUTH   M.M.TEETER,H.HOPE                                   1CBN  28
REMARK   1  TITL   PROGRESS IN THE WATER STRUCTURE OF THE PROTEIN      1CBN  29
REMARK   1  TITL 2 CRAMBIN BY X-RAY DIFFRACTION AT 140 K               1CBN  30
REMARK   1  REF    ANN.N.Y.ACAD.SCI.            V. 482   163 1986      1CBN  31
REMARK   1  REFN   ASTM ANYAA9  US ISSN 0077-8923                 332   1CBN  32
REMARK   1 REFERENCE 4                                                 1CBN  33
REMARK   1  AUTH   M.M.TEETER                                          1CBN  34
REMARK   1  TITL   WATER STRUCTURE OF A HYDROPHOBIC PROTEIN AT         1CBN  35
REMARK   1  TITL 2 ATOMIC RESOLUTION.  PENTAGON RINGS OF WATER         1CBN  36
REMARK   1  TITL 3 MOLECULES IN CRYSTALS OF CRAMBIN                    1CBN  37
REMARK   1  REF    PROC.NAT.ACAD.SCI.USA        V.  81  6014 1984      1CBN  38
REMARK   1  REFN   ASTM PNASA6  US ISSN 0027-8424                 040   1CBN  39
REMARK   1 REFERENCE 5                                                 1CBN  40
REMARK   1  AUTH   W.A.HENDRICKSON,M.M.TEETER                          1CBN  41
REMARK   1  TITL   STRUCTURE OF THE HYDROPHOBIC PROTEIN CRAMBIN        1CBN  42
REMARK   1  TITL 2 DETERMINED DIRECTLY FROM THE ANOMALOUS SCATTERING   1CBN  43
REMARK   1  TITL 3 OF SULPHUR                                          1CBN  44
REMARK   1  REF    NATURE                       V. 290   107 1981      1CBN  45
REMARK   1  REFN   ASTM NATUAS  UK ISSN 0028-0836                 006   1CBN  46
REMARK   1 REFERENCE 6                                                 1CBN  47
REMARK   1  AUTH   M.M.TEETER,W.A.HENDRICKSON                          1CBN  48
REMARK   1  TITL   HIGHLY ORDERED CRYSTALS OF THE PLANT SEED PROTEIN   1CBN  49
REMARK   1  TITL 2 CRAMBIN                                            1CBN  50
REMARK   1  REF    J.MOL.BIOL.                  V. 127   219 1979      1CBN  51
REMARK   1  REFN   ASTM JMOBAK  UK ISSN 0022-2836                 070   1CBN  52
REMARK   2                                                             1CBN  53
REMARK   2 RESOLUTION. 0.83 ANGSTROMS.                                 1CBN  54
REMARK   3                                                             1CBN  55
REMARK   3 REFINEMENT.                                                 1CBN  56
REMARK   3   PROGRAM                    PROLSQ                         1CBN  57
REMARK   3   AUTHORS                    KONNERT,HENDRICKSON            1CBN  58
REMARK   3   R VALUE                    0.106                          1CBN  59
REMARK   3   RMSD BOND DISTANCES        0.020  ANGSTROMS               1CBN  60
REMARK   3   RMSD BOND ANGLE DISTANCES  0.041  ANGSTROMS               1CBN  61
REMARK   4                                                             1CBN  62
REMARK   4 SEQUENCE ADVISORY NOTICE:                                   1CBN  63
REMARK   4 THERE IS SEQUENCE MICROHETEROGENEITY FOR RESIDUES 22 AND    1CBN  64
REMARK   4 25.  RESIDUE 22 CAN BE PRO OR SER AND RESIDUE 25 CAN BE LEU 1CBN  65
REMARK   4 OR ILE.  THE MOST LIKELY COMPOSITIONS FOR CRAMBIN IN THE    1CBN  66
REMARK   4 CRYSTAL USED IN THIS STUDY IS PRO 22 - LEU 25 AND           1CBN  67
REMARK   4 SER 22 - ILE 25.  BECAUSE OF LIMITATIONS IN PROTEIN DATA    1CBN  68
REMARK   4 FORMAT, ONLY PRO 22 AND LEU 22 ARE SHOWN ON THE SEQRES      1CBN  69
REMARK   4 RECORDS BELOW.  IN ADDITION RESIDUES SER 22 AND ILE 25 ARE  1CBN  70
REMARK   4 PRESENTED AS RESIDUES SER 22B AND ILE 25B ON THE ATOM       1CBN  71
REMARK   4 RECORDS BELOW, IMMEDIATELY FOLLOWING PRO 22 AND LEU 25,     1CBN  72
```

```
REMARK    4 RESPECTIVELY.                                                                1CBN  73
REMARK    5                                                                              1CBN  74
REMARK    5 IN SHEET *S1*, STRAND 3 IS QUESTIONABLY A STRAND.                            1CBN  75
REMARK    6                                                                              1CBN  76
REMARK    6 IN SHEET *S2*, STRAND 1 HAS NO HYDROGEN BONDS.  THE                          1CBN  77
REMARK    6 BACKBONE ATOMS ARE IN BETA CONFORMATION.                                     1CBN  78
SEQRES    1      46  THR THR CYS CYS PRO SER ILE VAL ALA ARG SER ASN PHE                 1CBN  79
SEQRES    2      46  ASN VAL CYS ARG LEU PRO GLY THR PRO GLU ALA LEU CYS                 1CBN  80
SEQRES    3      46  ALA THR TYR THR GLY CYS ILE ILE ILE PRO GLY ALA THR                 1CBN  81
SEQRES    4      46  CYS PRO GLY ASP TYR ALA ASN                                         1CBN  82
HET    EOH        66       5      ETHANOL                                              1CBN  83
FORMUL    2  EOH    C2 H6 O1                                                           1CBN  84
HELIX     1 H1  ILE       7  PRO      19   1 3/10 CONFORMATION RESID 17-19             1CBN  85
HELIX     2 H2  GLU      23  THR      30   1 ALPHA-N DISTORTION AT START               1CBN  86
SHEET     1 S1  3 CYS     32  ILE      35   0                                           1CBN  87
SHEET     2 S1  3 THR      1  CYS       4  -1                                          1CBN  88
SHEET     3 S1  3 ASN     46  ASN      46  -1                                          1CBN  89
SHEET     1 S2  1 THR     39  PRO      41   0                                          1CBN  90
TURN      1 T1  ARG     17  GLY     20                                                 1CBN  91
TURN      2 T2  PRO     41  TYR     44                                                 1CBN  92
SSBOND    1 CYS       3    CYS      40                                                 1CBN  93
SSBOND    2 CYS       4    CYS      32                                                 1CBN  94
SSBOND    3 CYS      16    CYS      26                                                 1CBN  95
CRYST1   40.763   18.492   22.333  90.00  90.61  90.00 P 21           2                1CBN  96
ORIGX1      1.000000  0.000000  0.000000        0.00000                                1CBN  97
ORIGX2      0.000000  1.000000  0.000000        0.00000                                1CBN  98
ORIGX3      0.000000  0.000000  1.000000        0.00000                                1CBN  99
SCALE1      0.024532  0.000000  0.000261        0.00000                                1CBN 100
SCALE2      0.000000  0.054077  0.000000        0.00000                                1CBN 101
SCALE3      0.000000  0.000000  0.044779        0.00000                                1CBN 102
ATOM      1  N  ATHR     1      16.864  14.059   3.442  0.80  6.22                    1CBN 103
ATOM      2  N  BTHR     1      17.633  14.126   4.146  0.20  8.40                    1CBN 104
ATOM      3  CA ATHR     1      16.868  12.814   4.233  0.80  4.45                    1CBN 105
ATOM      4  CA BTHR     1      17.282  12.671   4.355  0.20  7.82                    1CBN 106
ATOM      5  C   THR     1      15.583  12.775   4.990  1.00  4.39                    1CBN 107
ATOM      6  O   THR     1      15.112  13.824   5.431  1.00  7.04                    1CBN 108
ATOM      7  CB ATHR     1      18.060  12.807   5.200  0.80  5.42                    1CBN 109
ATOM      8  CB BTHR     1      18.202  11.709   5.108  0.20 11.07                    1CBN 110
ATOM      9  OG1ATHR     1      19.233  12.892   4.380  0.80  7.87                    1CBN 111
ATOM     10  OG1BTHR     1      17.662  10.381   4.831  0.20 14.39                    1CBN 112
ATOM     11  CG2ATHR     1      18.117  11.578   6.092  0.80  6.88                    1CBN 113
ATOM     12  CG2BTHR     1      17.973  11.955   6.599  0.20 19.74                    1CBN 114
ATOM     13 1H  ATHR     1      16.007  14.241   3.093  0.80  6.17                    1CBN 115
ATOM     14 2H  ATHR     1      17.087  14.800   3.997  0.80  5.69                    1CBN 116
ATOM     15 3H  ATHR     1      17.481  14.013   2.750  0.80  5.69                    1CBN 117
ATOM     16  HA ATHR     1      16.940  12.048   3.605  0.80  4.30                    1CBN 118
ATOM     17  HB ATHR     1      18.008  13.641   5.765  0.80  5.82                    1CBN 119
ATOM     18  HG1ATHR     1      19.761  13.553   4.647  0.80  6.62                    1CBN 120
ATOM     19 1HG2ATHR     1      17.940  10.768   5.540  0.80  6.36                    1CBN 121
ATOM     20 2HG2ATHR     1      18.983  11.488   6.531  0.80  6.13                    1CBN 122
ATOM     21 3HG2ATHR     1      17.391  11.604   6.799  0.80  5.68                    1CBN 123
ATOM     22  N   THR     2      15.050  11.574   5.153  1.00  4.23                    1CBN 124
ATOM     23  CA  THR     2      13.832  11.480   5.936  1.00  4.30                    1CBN 125
ATOM     24  C   THR     2      14.140  10.771   7.259  1.00  3.57                    1CBN 126
ATOM     25  O   THR     2      14.981   9.844   7.352  1.00  4.45                    1CBN 127
ATOM     26  CB ATHR     2      12.704  10.758   5.178  0.80  6.48                    1CBN 128
ATOM     27  CB BTHR     2      12.892  10.382   5.108  0.20  7.31                    1CBN 129
ATOM     28  OG1ATHR     2      13.216   9.504   4.842  0.80  6.62                    1CBN 130
ATOM     29  OG1BTHR     2      12.711  10.718   3.818  0.20  4.34                    1CBN 131
ATOM     30  CG2ATHR     2      12.269  11.573   3.957  0.80  7.05                    1CBN 132
ATOM     31  CG2BTHR     2      11.621   9.921   5.913  0.20  4.84                    1CBN 133
ATOM     32  H   THR     2      15.438  10.820   4.856  1.00  3.41                    1CBN 134
ATOM     33  HA  THR     2      13.491  12.378   6.134  1.00  3.88                    1CBN 135
ATOM     34  HB ATHR     2      11.919  10.668   5.811  0.80  6.19                    1CBN 136
ATOM     35  HG1ATHR     2      13.260   9.363   3.966  0.80  6.05                    1CBN 137
ATOM     36 1HG2ATHR     2      11.950  12.485   4.227  0.80  6.77                    1CBN 138
ATOM     37 2HG2ATHR     2      13.054  11.692   3.362  0.80  6.63                    1CBN 139
ATOM     38 3HG2ATHR     2      11.560  11.107   3.470  0.80  7.49                    1CBN 140
ATOM     39  N   CYS     3      13.415  11.193   8.281  1.00  3.32                    1CBN 141
ATOM     40  CA  CYS     3      13.564  10.664   9.653  1.00  3.69                    1CBN 142
ATOM     41  C   CYS     3      12.179  10.373  10.192  1.00  3.13                    1CBN 143
ATOM     42  O   CYS     3      11.303  11.249   9.999  1.00  3.95                    1CBN 144
ATOM     43  CB  CYS     3      14.253  11.707  10.547  1.00  3.67                    1CBN 145
ATOM     44  SG  CYS     3      15.804  12.398   9.921  1.00  4.70                    1CBN 146
ATOM     45  H   CYS     3      12.845  11.899   8.208  1.00  2.69                    1CBN 147
ATOM     46  HA  CYS     3      14.118   9.887   9.650  1.00  3.08                    1CBN 148
ATOM     47 1HB  CYS     3      13.651  12.471  10.670  1.00  2.92                    1CBN 149
ATOM     48 2HB  CYS     3      14.442  11.331  11.426  1.00  3.39                    1CBN 150
ATOM     49  N   CYS     4      11.998   9.281  10.858  1.00  2.95                    1CBN 151
ATOM     50  CA  CYS     4      10.657   8.927  11.341  1.00  2.76                    1CBN 152
ATOM     51  C   CYS     4      10.648   8.749  12.842  1.00  2.67                    1CBN 153
ATOM     52  O   CYS     4      11.615   8.257  13.440  1.00  3.24                    1CBN 154
ATOM     53  CB  CYS     4      10.137   7.678  10.634  1.00  3.20                    1CBN 155
ATOM     54  SG  CYS     4       9.804   7.928   8.865  1.00  3.15                    1CBN 156
ATOM     55  H   CYS     4      12.658   8.660  10.951  1.00  2.05                    1CBN 157
ATOM     56  HA  CYS     4      10.037   9.642  11.092  1.00  2.84                    1CBN 158
ATOM     57 1HB  CYS     4      10.790   6.963  10.692  1.00  2.94                    1CBN 159
ATOM     58 2HB  CYS     4       9.300   7.373  11.036  1.00  2.33                    1CBN 160
ATOM     59  N   PRO     5       9.529   9.075  13.506  1.00  2.81                    1CBN 161
ATOM     60  CA  PRO     5       9.474   9.025  14.942  1.00  3.11                    1CBN 162
ATOM     61  C   PRO     5       9.335   7.680  15.544  1.00  3.30                    1CBN 163
ATOM     62  O   PRO     5       9.599   7.511  16.761  1.00  4.77                    1CBN 164
ATOM     63  CB  PRO     5       8.247   9.901  15.266  1.00  3.52                    1CBN 165
ATOM     64  CG  PRO     5       7.322   9.733  14.055  1.00  3.44                    1CBN 166
ATOM     65  CD  PRO     5       8.316   9.745  12.912  1.00  2.86                    1CBN 167
```

```
ATOM     66  HA  PRO     5      10.272   9.410  15.343  1.00  3.20       1CBN 168
ATOM     67 1HB  PRO     5       7.828   9.705  16.029  1.00  2.63       1CBN 169
ATOM     68 2HB  PRO     5       8.508  10.814  15.358  1.00  2.54       1CBN 170
ATOM     69 1HG  PRO     5       6.930   8.993  14.102  1.00  2.88       1CBN 171
ATOM     70 2HG  PRO     5       6.681  10.453  13.977  1.00  3.25       1CBN 172
ATOM     71 1HD  PRO     5       7.995   9.225  12.148  1.00  3.05       1CBN 173
ATOM     72 2HD  PRO     5       8.592  10.611  12.555  1.00  2.71       1CBN 174
ATOM     73  N   SER     6       8.903   6.701  14.772  1.00  2.75       1CBN 175
ATOM     74  CA  SER     6       8.706   5.341  15.230  1.00  2.94       1CBN 176
ATOM     75  C   SER     6       8.791   4.422  14.009  1.00  3.08       1CBN 177
ATOM     76  O   SER     6       8.722   4.876  12.867  1.00  3.29       1CBN 178
ATOM     77  CB  SER     6       7.376   5.151  15.947  1.00  3.38       1CBN 179
ATOM     78  OG  SER     6       6.327   5.254  14.943  1.00  3.48       1CBN 180
ATOM     79  H   SER     6       8.760   6.855  13.874  1.00  5.06       1CBN 181
ATOM     80  HA  SER     6       9.380   5.087  15.880  1.00  3.01       1CBN 182
ATOM     81 1HB  SER     6       7.323   4.248  16.309  1.00  2.78       1CBN 183
ATOM     82 2HB  SER     6       7.229   5.831  16.635  1.00  2.84       1CBN 184
ATOM     83  HG  SER     6       5.605   4.995  15.359  1.00  3.13       1CBN 185
ATOM     84  N   ILE     7       8.887   3.129  14.301  1.00  3.60       1CBN 186
ATOM     85  CA  ILE     7       8.941   2.087  13.267  1.00  4.98       1CBN 187
ATOM     86  C   ILE     7       7.653   2.106  12.395  1.00  3.75       1CBN 188
ATOM     87  O   ILE     7       7.752   2.109  11.182  1.00  5.13       1CBN 189
ATOM     88  CB  ILE     7       9.193   0.736  13.880  1.00  7.27       1CBN 190
ATOM     89  CG1 ILE     7      10.659   0.726  14.337  1.00 11.19       1CBN 191
ATOM     90  CG2 ILE     7       8.877  -0.425  12.927  1.00 10.39       1CBN 192
ATOM     91  CD1AILE     7      11.293  -0.460  15.011  0.70 10.78       1CBN 193
ATOM     92  CD1BILE     7      11.538   0.178  13.256  0.30  7.65       1CBN 194
ATOM     93  H   ILE     7       8.885   2.833  15.164  1.00  2.71       1CBN 195
ATOM     94  HA  ILE     7       9.687   2.293  12.658  1.00  5.28       1CBN 196
ATOM     95  HB  ILE     7       8.615   0.637  14.674  1.00  8.32       1CBN 197
ATOM     96 1HG1AILE     7      11.182   0.915  13.495  0.70  8.99       1CBN 198
ATOM     97 1HG1BILE     7      10.893   1.657  14.542  0.30  9.15       1CBN 199
ATOM     98 2HG1AILE     7      10.782   1.519  14.934  0.70  9.14       1CBN 200
ATOM     99 2HG1BILE     7      10.723   0.161  15.142  0.30  9.59       1CBN 201
ATOM    100 1HG2 ILE     7       8.879  -1.259  13.461  1.00  9.84       1CBN 202
ATOM    101 2HG2 ILE     7       9.581  -0.446  12.230  1.00  9.69       1CBN 203
ATOM    102 3HG2 ILE     7       7.993  -0.280  12.525  1.00  9.37       1CBN 204
ATOM    103 1HD1AILE     7      10.589  -0.934  15.540  0.70 12.06       1CBN 205
ATOM    104 1HD1BILE     7      10.967  -0.201  12.521  0.30  9.32       1CBN 206
ATOM    105 2HD1AILE     7      11.690  -1.087  14.327  0.70 12.32       1CBN 207
ATOM    106 2HD1BILE     7      12.061   0.936  12.858  0.30  9.70       1CBN 208
ATOM    107 3HD1AILE     7      11.998  -0.148  15.646  0.70 11.78       1CBN 209
ATOM    108 3HD1BILE     7      12.163  -0.521  13.603  0.30  9.35       1CBN 210

        ICI on saute les residus 8 a 20 pour limiter le nombre de pages...

ATOM    338  N   THR    21       4.065   9.392  -4.234  1.00  2.42       1CBN 440
ATOM    339  CA  THR    21       3.677  10.559  -3.462  1.00  2.82       1CBN 441
ATOM    340  C   THR    21       4.903  11.355  -3.019  1.00  2.63       1CBN 442
ATOM    341  O   THR    21       5.844  10.709  -2.465  1.00  3.41       1CBN 443
ATOM    342  CB  THR    21       2.945  10.108  -2.197  1.00  2.66       1CBN 444
ATOM    343  OG1 THR    21       1.965   9.112  -2.559  1.00  3.02       1CBN 445
ATOM    344  CG2 THR    21       2.287  11.257  -1.482  1.00  3.34       1CBN 446
ATOM    345  H   THR    21       3.688   8.587  -3.997  1.00  1.89       1CBN 447
ATOM    346  HA  THR    21       3.077  11.128  -3.963  1.00  2.61       1CBN 448
ATOM    347  HB  THR    21       3.618   9.655  -1.589  1.00  2.44       1CBN 449
ATOM    348  HG1 THR    21       2.135   8.347  -2.156  1.00  4.77       1CBN 450
ATOM    349 1HG2 THR    21       2.939  12.020  -1.308  1.00  1.75       1CBN 451
ATOM    350 2HG2 THR    21       1.573  11.647  -2.059  1.00  2.95       1CBN 452
ATOM    351 3HG2 THR    21       1.902  10.997  -0.619  1.00  2.63       1CBN 453
ATOM    352  N   PRO    22       4.909  12.659  -3.127  0.60  3.03       1CBN 454
ATOM    353  CA  PRO    22       6.035  13.459  -2.622  0.60  3.04       1CBN 455
ATOM    354  C   PRO    22       6.362  13.139  -1.174  0.60  3.08       1CBN 456
ATOM    355  O   PRO    22       5.473  12.959  -0.323  0.60  3.67       1CBN 457
ATOM    356  CB  PRO    22       5.528  14.895  -2.825  0.60  4.19       1CBN 458
ATOM    357  CG  PRO    22       4.614  14.846  -4.059  0.60  3.91       1CBN 459
ATOM    358  CD  PRO    22       3.904  13.493  -3.885  0.60  3.25       1CBN 460
ATOM    359  HA  PRO    22       6.808  13.304  -3.209  0.60  3.46       1CBN 461
ATOM    360 1HB  PRO    22       5.093  15.252  -2.128  0.60  3.86       1CBN 462
ATOM    361 2HB  PRO    22       6.254  15.492  -2.981  0.60  4.22       1CBN 463
ATOM    362 1HG  PRO    22       4.063  15.462  -4.030  0.60  3.83       1CBN 464
ATOM    363 2HG  PRO    22       5.119  14.843  -4.884  0.60  3.68       1CBN 465
ATOM    364 1HD  PRO    22       3.086  13.595  -3.383  0.60  3.65       1CBN 466
ATOM    365 2HD  PRO    22       3.686  13.019  -4.709  0.60  3.25       1CBN 467
ATOM    366  N   SER   22B       4.909  12.659  -3.127  0.40  3.03       1CBN 468
ATOM    367  CA  SER   22B       6.035  13.459  -2.622  0.40  3.04       1CBN 469
ATOM    368  C   SER   22B       6.362  13.139  -1.174  0.40  3.08       1CBN 470
ATOM    369  O   SER   22B       5.473  12.959  -0.323  0.40  3.67       1CBN 471
ATOM    370  CB  SER   22B       5.644  14.934  -2.679  0.40  3.96       1CBN 472
ATOM    371  OG ASER   22B       4.712  15.250  -1.677  0.20  3.53       1CBN 473
ATOM    372  OG BSER   22B       6.688  15.800  -2.315  0.20  7.09       1CBN 474
ATOM    373  H   SER   22B       4.323  13.093  -3.663  0.40  2.42       1CBN 475
ATOM    374  HA  SER   22B       6.808  13.304  -3.209  0.40  3.46       1CBN 476
ATOM    375 1HB ASER   22B       6.463  15.471  -2.534  0.20  3.82       1CBN 477
ATOM    376 1HB BSER   22B       5.381  15.136  -3.616  0.20  5.09       1CBN 478
ATOM    377 2HB ASER   22B       5.292  15.151  -3.585  0.20  3.90       1CBN 479
ATOM    378 2HB BSER   22B       4.838  15.084  -2.107  0.20  4.66       1CBN 480
ATOM    379  N   GLU    23       7.674  13.184  -0.854  1.00  3.14       1CBN 481
ATOM    380  CA  GLU    23       8.080  12.987   0.542  1.00  3.02       1CBN 482
ATOM    381  C   GLU    23       7.376  13.933   1.480  1.00  2.84       1CBN 483
ATOM    382  O   GLU    23       7.035  13.564   2.601  1.00  3.00       1CBN 484
ATOM    383  CB  GLU    23       9.625  13.151   0.698  1.00  3.53       1CBN 485
ATOM    384  CG AGLU    23      10.387  12.021   0.058  0.80  3.74       1CBN 486
ATOM    385  CG BGLU    23      10.421  12.092   0.015  0.20  5.25       1CBN 487
ATOM    386  CD AGLU    23      11.885  12.048   0.399  0.80  4.38       1CBN 488
```

```
ATOM    387  CD BGLU    23      11.899  12.484   0.296  0.20  7.30      1CBN 489
ATOM    388  OE1AGLU    23      12.428  13.096   0.743  0.80  5.50      1CBN 490
ATOM    389  OE1BGLU    23      12.243  13.657   0.425  0.20  7.49      1CBN 491
ATOM    390  OE2AGLU    23      12.521  10.923   0.293  0.80  6.59      1CBN 492
ATOM    391  OE2BGLU    23      12.751  11.501   0.211  0.20  5.04      1CBN 493
ATOM    392  H    GLU    23       8.294  13.350  -1.476  1.00  1.89      1CBN 494
ATOM    393  HA   GLU    23       7.869  12.057   0.794  1.00  2.16      1CBN 495
ATOM    394 1HB AGLU    23       9.878  14.005   0.271  0.80  2.06      1CBN 496
ATOM    395 2HB AGLU    23       9.787  13.219   1.653  0.80  2.19      1CBN 497
ATOM    396 1HG AGLU    23      10.085  11.130   0.299  0.80  3.21      1CBN 498
ATOM    397 2HG AGLU    23      10.354  12.099  -0.918  0.80  3.94      1CBN 499
ATOM    398  N    ALA    24       7.160  15.194   1.059  1.00  2.99      1CBN 500
ATOM    399  CA   ALA    24       6.592  16.184   1.972  1.00  2.64      1CBN 501
ATOM    400  C    ALA    24       5.175  15.818   2.410  1.00  2.55      1CBN 502
ATOM    401  O    ALA    24       4.777  16.010   3.540  1.00  3.06      1CBN 503
ATOM    402  CB   ALA    24       6.626  17.598   1.373  1.00  3.59      1CBN 504
ATOM    403  H    ALA    24       7.505  15.460   0.262  1.00  2.30      1CBN 505
ATOM    404  HA   ALA    24       7.153  16.232   2.780  1.00  2.94      1CBN 506
ATOM    405 1HB  ALA    24       7.543  17.842   1.137  1.00  2.49      1CBN 507
ATOM    406 2HB  ALA    24       6.076  17.631   0.563  1.00  2.50      1CBN 508
ATOM    407 3HB  ALA    24       6.263  18.240   2.020  1.00  2.15      1CBN 509
ATOM    408  N    LEU    25       4.377  15.265   1.447  0.60  2.82      1CBN 510
ATOM    409  CA   LEU    25       3.033  14.786   1.800  0.60  2.71      1CBN 511
ATOM    410  C    LEU    25       3.077  13.514   2.641  0.60  2.38      1CBN 512
ATOM    411  O    LEU    25       2.310  13.367   3.581  0.60  2.92      1CBN 513
ATOM    412  CB   LEU    25       2.186  14.577   0.518  0.60  2.90      1CBN 514
ATOM    413  CG   LEU    25       1.814  15.965  -0.117  0.60  4.43      1CBN 515
ATOM    414  CD1  LEU    25       1.222  15.717  -1.500  0.60  5.09      1CBN 516
ATOM    415  CD2  LEU    25       0.623  16.572   0.761  0.60  4.11      1CBN 517
ATOM    416  H    LEU    25       4.728  15.070   0.644  0.60  1.70      1CBN 518
ATOM    417  HA   LEU    25       2.573  15.480   2.312  0.60  2.56      1CBN 519
ATOM    418 1HB  LEU    25       2.832  14.186  -0.175  0.60  3.74      1CBN 520
ATOM    419 2HB  LEU    25       1.312  14.093   0.720  0.60  3.27      1CBN 521
ATOM    420  HG   LEU    25       2.617  16.584  -0.248  0.60  2.73      1CBN 522
ATOM    421 1HD1 LEU    25       0.515  15.030  -1.466  0.60  4.69      1CBN 523
ATOM    422 2HD1 LEU    25       0.829  16.587  -1.836  0.60  4.59      1CBN 524
ATOM    423 3HD1 LEU    25       1.946  15.436  -2.128  0.60  4.44      1CBN 525
ATOM    424 1HD2 LEU    25      -0.170  15.974   0.843  0.60  2.83      1CBN 526
ATOM    425 2HD2 LEU    25       0.989  16.789   1.668  0.60  3.14      1CBN 527
ATOM    426 3HD2 LEU    25       0.344  17.402   0.282  0.60  3.79      1CBN 528
ATOM    427  N    ILE   25B      4.377  15.265   1.447  0.40  2.82      1CBN 529
ATOM    428  CA   ILE   25B      3.033  14.786   1.800  0.40  2.71      1CBN 530
ATOM    429  C    ILE   25B      3.077  13.514   2.641  0.40  2.38      1CBN 531
ATOM    430  O    ILE   25B      2.310  13.367   3.581  0.40  2.92      1CBN 532
ATOM    431  CB   ILE   25B      2.186  14.577   0.518  0.40  2.90      1CBN 533
ATOM    432  CG1  ILE   25B      1.658  15.855  -0.086  0.40  2.48      1CBN 534
ATOM    433  CG2  ILE   25B      0.962  13.694   0.784  0.40  4.13      1CBN 535
ATOM    434  CD1AILE   25B      0.623  16.572   0.761  0.20  4.11      1CBN 536
ATOM    435  CD1BILE   25B      1.222  15.717  -1.500  0.20  5.09      1CBN 537
ATOM    436  H    ILE   25B      4.728  15.070   0.644  0.40  1.70      1CBN 538
ATOM    437  HA   ILE   25B      2.573  15.480   2.312  0.40  2.56      1CBN 539
ATOM    438  HB   ILE   25B      2.769  14.106  -0.118  0.40  3.37      1CBN 540
ATOM    439 1HG1AILE   25B      2.432  16.460  -0.225  0.20  2.73      1CBN 541
ATOM    440 1HG1BILE   25B      1.075  16.325   0.437  0.20  3.30      1CBN 542
ATOM    441 2HG1AILE   25B      1.243  15.658  -0.968  0.20  2.96      1CBN 543
ATOM    442 2HG1BILE   25B      2.432  16.460  -0.225  0.20  2.73      1CBN 544
ATOM    443 1HG2 ILE   25B      0.423  14.095   1.506  0.40  3.33      1CBN 545
ATOM    444 2HG2 ILE   25B      0.451  13.622  -0.059  0.40  3.57      1CBN 546
ATOM    445 3HG2 ILE   25B      1.278  12.796   1.059  0.40  3.60      1CBN 547
ATOM    446 1HD1AILE   25B      0.989  16.789   1.668  0.20  3.14      1CBN 548
ATOM    447 1HD1BILE   25B      0.515  15.030  -1.466  0.20  4.69      1CBN 549
ATOM    448 2HD1AILE   25B      0.344  17.402   0.282  0.20  3.79      1CBN 550
ATOM    449 2HD1BILE   25B      0.829  16.587  -1.836  0.20  4.59      1CBN 551
ATOM    450 3HD1AILE   25B     -0.170  15.974   0.843  0.20  2.83      1CBN 552
ATOM    451 3HD1BILE   25B      1.946  15.436  -2.128  0.20  4.44      1CBN 553
ATOM    452  N    CYS    26       4.010  12.607   2.307  1.00  2.74      1CBN 554
ATOM    453  CA   CYS    26       4.116  11.434   3.161  1.00  2.53      1CBN 555
ATOM    454  C    CYS    26       4.590  11.790   4.576  1.00  2.43      1CBN 556
ATOM    455  O    CYS    26       4.222  11.090   5.540  1.00  3.18      1CBN 557
ATOM    456  CB   CYS    26       5.022  10.378   2.542  1.00  3.03      1CBN 558
ATOM    457  SG   CYS    26       4.343   9.677   1.004  1.00  3.12      1CBN 559
ATOM    458  H    CYS    26       4.562  12.745   1.613  1.00  1.80      1CBN 560
ATOM    459  HA   CYS    26       3.236  11.004   3.234  1.00  1.94      1CBN 561
ATOM    460 1HB  CYS    26       5.874  10.780   2.294  1.00  2.78      1CBN 562
ATOM    461 2HB  CYS    26       5.169   9.629   3.135  1.00  2.77      1CBN 563
ATOM    462  N    ALA    27       5.387  12.845   4.701  1.00  2.59      1CBN 564
ATOM    463  CA   ALA    27       5.818  13.340   6.023  1.00  2.75      1CBN 565
ATOM    464  C    ALA    27       4.648  13.792   6.870  1.00  2.87      1CBN 566
ATOM    465  O    ALA    27       4.466  13.329   7.986  1.00  3.23      1CBN 567
ATOM    466  CB   ALA    27       6.882  14.427   5.838  1.00  2.94      1CBN 568
ATOM    467  H    ALA    27       5.625  13.325   3.972  1.00  1.96      1CBN 569
ATOM    468  HA   ALA    27       6.280  12.595   6.477  1.00  2.97      1CBN 570
ATOM    469 1HB  ALA    27       7.628  14.068   5.304  1.00  2.23      1CBN 571
ATOM    470 2HB  ALA    27       6.502  15.193   5.354  1.00  2.32      1CBN 572
ATOM    471 3HB  ALA    27       7.213  14.737   6.702  1.00  2.45      1CBN 573
ATOM    472  N    THR    28       3.829  14.741   6.338  1.00  2.95      1CBN 574
ATOM    473  CA   THR    28       2.708  15.163   7.150  1.00  3.39      1CBN 575
ATOM    474  C    THR    28       1.696  14.058   7.401  1.00  3.24      1CBN 576
ATOM    475  O    THR    28       0.984  14.051   8.406  1.00  4.36      1CBN 577
ATOM    476  CB   THR    28       2.009  16.426   6.567  1.00  3.18      1CBN 578
ATOM    477  OG1  THR    28       1.678  16.122   5.199  1.00  3.69      1CBN 579
ATOM    478  CG2  THR    28       2.878  17.654   6.666  1.00  4.26      1CBN 580
ATOM    479  H    THR    28       3.973  15.030   5.506  1.00  1.94      1CBN 581
ATOM    480  HA   THR    28       3.045  15.474   8.031  1.00  2.87      1CBN 582
ATOM    481  HB   THR    28       1.157  16.545   7.086  1.00  3.39      1CBN 583
```

```
ATOM    482  HG1 THR    28       1.024  16.666   4.925  1.00  3.53       1CBN 584
ATOM    483 1HG2 THR    28       3.741  17.490   6.189  1.00  3.21       1CBN 585
ATOM    484 2HG2 THR    28       2.457  18.446   6.263  1.00  3.51       1CBN 586
ATOM    485 3HG2 THR    28       3.108  17.877   7.623  1.00  3.98       1CBN 587
ATOM    486   N   TYR    29       1.598  13.082   6.455  1.00  2.82       1CBN 588
ATOM    487  CA   TYR    29       0.682  12.007   6.659  1.00  3.10       1CBN 589
ATOM    488   C   TYR    29       1.063  11.059   7.808  1.00  3.15       1CBN 590
ATOM    489   O   TYR    29       0.203  10.507   8.463  1.00  4.59       1CBN 591
ATOM    490  CB ATYR    29       0.567  11.223   5.288  0.60  4.44       1CBN 592
ATOM    491  CB BTYR    29       0.577  11.266   5.295  0.40  3.95       1CBN 593
ATOM    492  CG ATYR    29       0.020   9.814   5.251  0.60  4.59       1CBN 594
ATOM    493  CG BTYR    29      -0.542  10.270   5.306  0.40  4.03       1CBN 595
ATOM    494 CD1ATYR    29      -1.332   9.554   5.514  0.60  5.34       1CBN 596
ATOM    495 CD1BTYR    29      -1.816  10.563   5.826  0.40  4.48       1CBN 597
ATOM    496 CD2ATYR    29       0.845   8.736   4.934  0.60  5.36       1CBN 598
ATOM    497 CD2BTYR    29      -0.265   8.932   5.005  0.40 10.03       1CBN 599
ATOM    498 CE1ATYR    29      -1.821   8.250   5.516  0.60  5.27       1CBN 600
ATOM    499 CE1BTYR    29      -2.831   9.612   5.845  0.40  4.38       1CBN 601
ATOM    500 CE2ATYR    29       0.385   7.429   4.942  0.60  5.82       1CBN 602
ATOM    501 CE2BTYR    29      -1.269   7.949   5.037  0.40  9.90       1CBN 603
ATOM    502  CZ ATYR    29      -0.944   7.198   5.246  0.60  5.21       1CBN 604
ATOM    503  CZ BTYR    29      -2.536   8.310   5.477  0.40  7.71       1CBN 605
ATOM    504  OH ATYR    29      -1.387   5.915   5.236  0.60  6.27       1CBN 606
ATOM    505  OH BTYR    29      -3.511   7.365   5.278  0.40 15.00       1CBN 607
ATOM    506   H   TYR    29       2.114  13.148   5.713  1.00  3.31       1CBN 608
ATOM    507  HA   TYR    29      -0.229  12.330   6.824  1.00  3.36       1CBN 609
ATOM    508 1HB ATYR    29      -0.017  11.808   4.732  0.60  4.12       1CBN 610
ATOM    509 1HB BTYR    29       0.421  11.960   4.614  0.40  3.37       1CBN 611
ATOM    510 2HB ATYR    29       1.469  11.262   4.877  0.60  4.15       1CBN 612
ATOM    511 2HB BTYR    29       1.448  10.843   5.107  0.40  3.43       1CBN 613
ATOM    512 HD1ATYR    29      -1.936  10.295   5.754  0.60  4.54       1CBN 614
ATOM    513 HD1BTYR    29      -2.032  11.508   6.031  0.40  5.23       1CBN 615
ATOM    514 HD2ATYR    29       1.800   8.912   4.709  0.60  5.78       1CBN 616
ATOM    515 HD2BTYR    29       0.573   8.709   4.512  0.40  7.39       1CBN 617
ATOM    516 HE1ATYR    29      -2.761   8.080   5.752  0.60  7.88       1CBN 618
ATOM    517 HE1BTYR    29      -3.769   9.894   5.975  0.40  8.10       1CBN 619
ATOM    518 HE2ATYR    29       0.976   6.676   4.710  0.60  4.05       1CBN 620
ATOM    519 HE2BTYR    29      -1.015   6.997   4.985  0.40  7.98       1CBN 621
ATOM    520   N   THR    30       2.412  10.870   7.986  1.00  2.81       1CBN 622
ATOM    521  CA   THR    30       2.910   9.891   8.934  1.00  3.45       1CBN 623
ATOM    522   C   THR    30       3.530  10.406  10.229  1.00  4.22       1CBN 624
ATOM    523   O   THR    30       3.690   9.603  11.128  1.00  5.38       1CBN 625
ATOM    524  CB   THR    30       4.022   9.025   8.261  1.00  4.10       1CBN 626
ATOM    525 OG1 THR    30       5.076   9.886   7.858  1.00  3.55       1CBN 627
ATOM    526 CG2 THR    30       3.492   8.238   7.078  1.00  5.98       1CBN 628
ATOM    527   H   THR    30       2.981  11.285   7.426  1.00  3.36       1CBN 629
ATOM    528  HA   THR    30       2.207   9.243   9.153  1.00  3.21       1CBN 630
ATOM    529  HB   THR    30       4.357   8.393   8.974  1.00  4.95       1CBN 631
ATOM    530 HG1 THR    30       5.140   9.936   6.972  1.00  5.09       1CBN 632
ATOM    531 1HG2 THR    30       2.677   7.698   7.344  1.00  4.37       1CBN 633
ATOM    532 2HG2 THR    30       3.222   8.851   6.357  1.00  3.77       1CBN 634
ATOM    533 3HG2 THR    30       4.175   7.612   6.737  1.00  4.49       1CBN 635
ATOM    534   N   GLY    31       3.897  11.670  10.231  1.00  3.82       1CBN 636
ATOM    535  CA   GLY    31       4.718  12.221  11.316  1.00  4.22       1CBN 637
ATOM    536   C   GLY    31       6.221  12.143  11.061  1.00  3.59       1CBN 638
ATOM    537   O   GLY    31       6.985  12.680  11.889  1.00  3.95       1CBN 639
ATOM    538   H   GLY    31       3.822  12.190   9.480  1.00  5.94       1CBN 640
ATOM    539 1HA GLY    31       4.472  13.159  11.455  1.00  4.08       1CBN 641
ATOM    540 2HA GLY    31       4.485  11.749  12.170  1.00  4.31       1CBN 642
ATOM    541   N   CYS    32       6.646  11.509   9.971  1.00  2.98       1CBN 643
ATOM    542  CA   CYS    32       8.050  11.589   9.560  1.00  3.06       1CBN 644
ATOM    543   C   CYS    32       8.352  13.008   9.148  1.00  3.34       1CBN 645
ATOM    544   O   CYS    32       7.461  13.875   8.946  1.00  3.62       1CBN 646
ATOM    545  CB   CYS    32       8.361  10.618   8.453  1.00  2.76       1CBN 647
ATOM    546  SG   CYS    32       7.987   8.859   8.836  1.00  3.14       1CBN 648
ATOM    547   H   CYS    32       6.035  11.197   9.377  1.00  1.95       1CBN 649
ATOM    548  HA   CYS    32       8.618  11.329  10.318  1.00  3.22       1CBN 650
ATOM    549 1HB  CYS    32       7.816  10.812   7.661  1.00  2.96       1CBN 651
ATOM    550 2HB  CYS    32       9.294  10.628   8.186  1.00  3.15       1CBN 652

        ICI on saute les residus 33 a 45 pour limiter le nombre de pages...

ATOM    737 3HB  ALA    45      15.947   7.697  17.201  1.00  4.18       1CBN 839
ATOM    738   N  ASN    46      13.925   6.435  13.670  1.00  3.39       1CBN 840
ATOM    739  CA  ASN    46      13.501   5.329  12.797  1.00  3.70       1CBN 841
ATOM    740   C  ASN    46      13.281   5.771  11.349  1.00  3.70       1CBN 842
ATOM    741   O  ASN    46      13.691   6.901  11.001  1.00  4.09       1CBN 843
ATOM    742  CB  ASN    46      12.252   4.651  13.386  1.00  4.58       1CBN 844
ATOM    743  CG  ASN    46      12.526   4.138  14.764  1.00  5.34       1CBN 845
ATOM    744 OD1 ASN    46      13.443   3.218  14.924  1.00  9.98       1CBN 846
ATOM    745 ND2 ASN    46      11.916   4.601  15.746  1.00  5.99       1CBN 847
ATOM    746 OXT ASN    46      12.678   4.980  10.600  1.00  4.99       1CBN 848
ATOM    747   H  ASN    46      13.288   7.025  13.931  1.00  7.58       1CBN 849
ATOM    748  HA  ASN    46      14.199   4.633  12.824  1.00  3.68       1CBN 850
ATOM    749 1HB  ASN    46      11.525   5.317  13.418  1.00  4.22       1CBN 851
ATOM    750 2HB  ASN    46      11.960   3.924  12.790  1.00  4.74       1CBN 852
ATOM    751 1HD2 ASN    46      12.053   4.334  16.647  1.00  7.82       1CBN 853
ATOM    752 2HD2 ASN    46      11.317   5.324  15.663  1.00  7.54       1CBN 854
TER     753      ASN    46                                              1CBN 855
HETATM  754  C1 AEOH   66      14.823  -0.159  13.271  0.70 13.49       1CBN 856
HETATM  755  C1 BEOH   66      15.763  -0.521  12.803  0.30 10.99       1CBN 857
HETATM  756  C2  EOH   66      15.702   0.904  12.771  1.00 17.90       1CBN 858
HETATM  757   O AEOH   66      15.029   2.134  12.501  0.70  7.21       1CBN 859
HETATM  758   O BEOH   66      14.540   1.708  12.931  0.30  6.09       1CBN 860
CONECT   44   43  665                                                   1CBN 861
```

```
CONECT   54   53  546                                                  1CBN 862
CONECT  269  268  457                                                  1CBN 863
CONECT  457  269  456                                                  1CBN 864
CONECT  546   54  545                                                  1CBN 865
CONECT  665   44  664                                                  1CBN 866
CONECT  754  756                                                       1CBN 867
CONECT  755  756                                                       1CBN 868
CONECT  756  755  758                                                  1CBN 869
CONECT  757  756                                                       1CBN 870
CONECT  758  756                                                       1CBN 871
MASTER        66    0    1    2    4    2    0    6  757    1   11    4 1CBN 872
END                                                                    1CBN 873
```

ANNEXE B

Les classes du système de lecture PDB

Les interfaces usager des classes correspondantes au système de lecture de fichier PDB (Figure 19) sont présentées.

```
extern const int ATOM_N;
extern const int ATOM_CA;
extern const int ATOM_C;
extern const int ATOM_O;
extern const int ATOM_HC;
class C_AtomPdb {
public:
  // ********* Contenu *************************
  int _Atom;
  char _Res,_Insert,_ChainId,_Altern;
  float _x,_y,_z;
  int _ResPos,_Serial;
  // ************* Methodes ********************
  // Constructeurs ++++
  C_AtomPdb(const char* PDB_buffer) ;
  C_AtomPdb(int atom,int ser_num,char id,char res,
    int respos,float x,float y,float z) ;
  // Modificateurs +++++
  int SetAtom(const char *name);
  void SetAtom(int token) { _Atom = token; };
  void SetResidu(char Res) { _Res = Res; };
  void SetPosition(int Pos) { _ResPos = Pos; };
  void SetInsert(char insert) { _Insert = insert; };
  void SetChainId(char id) { _ChainId = id; };
  // Operateurs +++++++
  C_AtomPdb& operator= (const C_AtomPdb &X) ;
  int operator== (const C_AtomPdb &X) const ;
  int operator< (const C_AtomPdb &X) const ;
  int operator> (const C_AtomPdb &X) const ;
  float operator[] (char T) const ; // X,Y,Z
  int Empty (void) const; //
  int SameKind (const C_AtomPdb &X) const; // tous les champs sauf 3D
  int Same3D (const C_AtomPdb &X) const ; // champs x,y et z
  int SameAtom (const char *name) const;
  int SameAtom (int token) const { return _Atom == token; };
  int SameAtom (const C_AtomPdb &X) const;
  int SameResidu (const C_AtomPdb &X) const;
  int BackboneMember (void) const;
  int Within (const C_AtomPdb * A,float d) const; // distance euclidienne
  int Within (const C_AtomPdb * A,float d1,float d2) const;
  // Acces au contenu  +++++++
  int GetSerial(void) const { return _Serial; };
  int GetAtom(void) const { return _Atom; };
  const char* GetAtomName(void) const { return AtomBank[_Atom]; };
  char GetAlternate(void) const { return _Altern; };
  char GetResidu(void) const { return _Res; };
  int GetChainId(void) const { return _ChainId; };
  int GetPosition(void) const { return _ResPos; };
  char GetInsertion(void) const { return _Insert; };
  float GetX (void) const { return _x; };
  float GetY (void) const { return _y; };
  float GetZ (void) const { return _z; };
  // Impression  ++++++
  int PrintPdb (ostream& out) const;
  int PrintClosure (ostream& out) const; // pour residu terminal
};
```

FIGURE 41. Présentation de l'interface usager de la classe C_AtomPDB.

```
class C_ResiduPdb {
public:
  // ********* Contenu ******************
  char _Res,_Insert,_ChainId;
  int _ResPos;
  vector < C_AtomPdb* > _VAtomPdb;
  // ************* Methodes ********************
  // Constructeurs +++++
  C_ResiduPdb (const char *pdb_line);
  // Modificateurs  +++++
  int Reset (const char *pdb_line); { Reset(); return Add(pdb_line); };
  int Reset (char Res,int Pos,char Insert,char id) ;
  int Add (const char *pdb_line) ;
  int Add (const C_AtomPdb* atom) ;
  int Remove (const char* atom);
  void SetChainId (char id) ;
  void SetResidu (char Res) ;
  void SetPosition (int Pos) ;
  void SetInsert (char insert) ;
  // Operateurs +++++
  C_ResiduPdb& operator= (const C_ResiduPdb& X) ;
  int operator== (const C_ResiduPdb& X) const;
  const C_AtomPdb* operator[] (int index) const;
  const C_AtomPdb* operator[] (const char* atom) const;
  const C_AtomPdb* operator() (int atom) const;
  static int check_flag(const char* buffer) {
    return C_AtomPdb::check_flag(buffer); }; // fonction globale
  static int check_flag(const char* buffer,char id) {
    return C_AtomPdb::check_flag(buffer,id); }; // fonction globale
  int SameResidu(const C_ResiduPdb& X) const; // compare le residu
  int SameAtomPdb(const C_ResiduPdb& X) const; // compare les atomes
  int Empty(void) const ;
  int IsIn(const char* atom,int& index) const;
  int IsNucleotide (void) const;
  int BackboneIn (void) const ;
  int Within (const C_ResiduPdb * X,float d) const;
  int Within (const C_ResiduPdb * X,float d1,float d2) const;
  // ACCES au contenu  +++++
  int Size(void) const { return _VAtomPdb.size(); };
  char GetResidu(void) const { return _Res; };
  int GetPosition(void) const { return _ResPos; };
  char GetChainId(void) const { return _ChainId; };
  char GetInsert(void) const { return _Insert; };
  // Impression ++++
  friend ostream& operator<< (ostream& out,const C_ResiduPdb &X) ;
  int PrintClosure(ostream& out) const ;
};
```

FIGURE 42. Présentation de l'interface usager simplifiée de la classe C_ResiduPDB.

```
class C_ConfigPdb
{
public:
  // ********** Contenu **************
  vector<char> _Seq;
  vector<C_ResiduPdb*> _VResPdb;
  char _ChainId;
  char* _File;
  int _Model;
  // ***********Methodes ************************
  // Constructeurrs ++++
  C_ConfigPdb (char id,int model,const char* file_name) ;
  // Modificateurs +++
  int Reset (void) ;
  int Reset (char id,int model,const char* file_name);
  int Reload (void) ; // relit l'information du fichier _File
  int Add (const C_ResiduPdb *X);
  int SetFile (const char* file_name);
  int SetChainId (char id);
  int SetModel (int model);
  int SetResiduPdb (const vector<C_ResiduPdb*> &VRes);
  // operateurs +++++
  C_ConfigPdb& operator= (const C_ConfigPdb &X);
  int operator== (const C_ConfigPdb &X) const;
  const C_ResiduPdb* operator[] (int index) const;
  const C_ResiduPdb* operator() (int ResPos) const;
  const C_ResiduPdb* operator() (int ResPos,char insert) const;
  int Empty (void) const ;
  int IsIn (int ResPos,char insert,int& index) const;
  int SameResidu (const vector<char> &seq) const;
  int Same3D (const vector<C_ResiduPdb*> &VRes) const;
  int IsModel (void) const ; // PDB contient parfois des modeles 3D
  // detecte les atomes qui sont a une distance de ''d'' angstrom
  int AnyCollisions (int d) const ;
  int AnyNucleotide (void) const ; // detecte la presence de nucleotide
  // ACCES au contenu ++++
  int Size (void) const ;
  int Size (char Res) const ;
  const char* GetFile (void) const ;
  const C_ResiduPdb* GetFirst (void) const ;
  const C_ResiduPdb* GetLast (void) const ;
  char GetResidu (int ResPos) const ;
  char GetChainId (void) const ;
  int GetModel (void) const ;
  int Transcript (vector<C_ResiduPdb*> & C) const;
  int Transcript (vector<const C_ResiduPdb*> & C) const;
  // filtre les r\'esidus a l'interieur de ''d_min''
  int Transcript (vector<C_ResiduPdb*> & C,float d_min) const;
  // Impression ++++
  int PrintPdb (ostream &out) const;
  int PrintPdb3D (ostream &out) const;
};
```

FIGURE 43. Présentation de l'interface usager de la classe C_ConfigPDB correspondant à la configuration d'une structure du fichier PDB. Un fichier peut contenir plusieurs structures qui sont différenciées par un champ "chain id".

```
extern const char SS_HELIX;
extern const char SS_SHEET;
extern const char SS_LOOP;
extern const char SS_COIL;
class C_SSPdb {
public:
  // Un ensemble de fonction globale sont definies pour
  // traiter les differents champs HELIX SHEET COIL LOOP
  // qui representent la structure secondaire de la proteine.
  // Ces fonctions ne sont pas presentees ici.
  // ********* Contentu ************************
  char _Type; // SS_HELIX SS_SHEET SS_LOOP SS_COIL
  int _Pos1,_Pos2,_Serial,_NumberStrand,_Sense,_HClass;
  char _Res1,_Res2,_Insert1,_Insert2,_ChainId1,_ChainId2;
  char _Id[4]; // identificateur du segment dans le fichier PDB
  // Constructeur ++++
  C_SSPdb(const char* pdb_line) ;
  C_SSPdb (char Type, char ChainId1,char Res1, int Pos1,
   char Insert1,char ChainId2, char Res2, int Pos2,
   char Insert2,const char* Id,int HClass,
   int Sense,int NumberStrand,int Serial) ;
  // Modificateurs ++++
  void SetType(char type) ;
  void SetChainId1(char ChainId) ;
  void SetResidu1(char Res) ;
  void SetPosition1(int Pos) ;
  void SetInsert1(char insert) ;
  void SetChainId2(char ChainId) ;
  void SetResidu2(char Res) ;
  void SetPosition2(int Pos) ;
  void SetInsert2(char insert) ;
  void SetId(const char* Id) ;
  void SetHClass(int HClass) ;
  void SetSense(int Sense) ;
  void SetNumberStrand(int NumberStrand) ;
  void SetSerial(int serial) ;
  // Operateurs ++++
  C_SSPdb& operator= (const C_SSPdb& X) ;
  int operator== (const C_SSPdb& X) const;
  int Empty (void) const ;
  int IsIn (int Pos) const ;
  int IsIn (int Pos1,int Pos2) const ;
  // test si [pos1,pos2] recouvre la SS de cette objet
  int IsOverlap (int Pos1,int Pos2) const ;
  // Acces au contenu ++++
  int Size(void) const ;
  char GetType(void) const ;
  char GetChainId1(void) const ;
  char GetResidu1(void) const ;
  int GetPosition1(void) const ;
  char GetInsert1(void) const ;
  char GetChainId2(void) const ;
  char GetResidu2(void) const ;
  int GetPosition2(void) const ;
  char GetInsert2(void) const ;
  const char* GetId(void) const ;
  int GetHClass(void) const ;
  int GetSense(void) const ;
  int GetNumberStrand(void) const ;
  int GetSerial(void) const ;
  // Impression +++++
  friend ostream& operator<< (ostream& out,const C_SSPdb& X);
};
```

FIGURE 44. Présentation de l'interface usager simplifiée de la classe C_SSPDB correspondant à la structure secondaire du fichier de la PDB. Un fichier de la PDB peut contenir plusieurs structures secondaires qui sont différenciées par un champ "chain id".

```
class C_SeqPdb {
private:
  // ******** Contenu a acces limite ***************
  char _ChainId,*_Seq;
  int _Size;
public:
  // ********** Methodes ***************
  // Constructeurs +++++
  C_SeqPdb (char chainid,const char * seq) ;
  C_SeqPdb (const char *pdb_buffer) ;
  // Modificateurs +++++
  int Reset (const char *pdb_file,char id) ;
  int Reset (char chainid,const char * seq);
  int Add (const char *pdb_buffer);
  // Operateurs ++++
  C_SeqPdb& operator= (const C_SeqPdb& X);
  int operator== (const C_SeqPdb& X) const;
  char operator[] (int index) const;
  int Empty (void) const ;
  int IsIn (int index) const;
  int IsIn (char Res) const;
  // Acces au contenu ++++++
  int Size (void) const ;
  char GetChainId (void) const;
  const char* GetSequence (void) const ;
  // Impression +++++
  int PrintPdb(ostream& out) const;
};
```

FIGURE 45. Présentation de l'interface usager simplifiée de la classe C_SeqPDB correspondant à la structure secondaire du fichier de la PDB. Cette classe permet d'extraire du fichier PDB la séquence de résidus.

```
class C_ParserPdb {
private:
  // ************** Contenu a acces limite **********
  char * _File;
public:
  // ************** Contenu ********************
  vector < C_SeqPdb* > _VSeqPdb;
  vector < C_SSPdb* > _VSSPdb;
  vector < C_ConfigPdb* > _VConfigPdb;
  vector < char > _VChainId;
  vector < int > _VModel;
  // *************** Methodes *************************
  // Constructeurs ++++++
  C_ParserPdb (const char* file) ;
  // Modificateurs +++++
  int Reset (const char* file,int model,char chainid) ;
  int Load3D (int model,char chainid);
  int TreatInsertion (int index);
  int TreatCollisions (int index,float d);
  void SetFile (const char* name) ;
  void SetVSeqPdb (const vector<C_SeqPdb*> &X) ;
  void SetVSSPdb (const vector<C_SSPdb*> &X) ;
  void SetVConfigPdb (const vector<C_ConfigPdb*> &X) ;
  int Add (C_ConfigPdb *X);
  int Add (C_SSPdb *X);
  int Add (C_SeqPdb *X);
  int Replace (C_ConfigPdb *X);
  int Replace (C_SSPdb *X);
  int Replace (C_SeqPdb *X);
  // operateurs  ++++++
  C_ParserPdb& operator= (const C_ParserPdb &X);
  int operator== (const C_ParserPdb &X) const;
  const C_ConfigPdb* operator[] (int index) const;
  const C_ConfigPdb* operator[] (char chain_id) const;
  const C_ConfigPdb* operator() (int model,char chain_id) const;
  // Acces au contenu +++++++
  int Size (void) const ;
  int ConfigPdbSize (void) const ;
  int ModelSize (void) const ;
  int ChainIdSize (void) const ;
  int SeqPdbSize (void) const ;
  int SSPdbSize (void) const ;
  const char * GetFile (void) const ;
  int Transcript (int index,vector<const C_ResiduPdb*> & VResPdb,
  vector<char> & VSS,float d_min) const;
  int Match (int seq_index,int config_index,
     vector<int> & map_seq2Config) const;
  int Empty(void) const ;
  int IsInConfigPdb (int model,char chain_id,int &index) const;
  const C_ConfigPdb* GetConfigPdb (int index) const;
  int IsInSeqPdb (char chain_id,int &index) const;
  const C_SeqPdb* GetSeqPdb (int index) const;
  int IsInSSPdb (char chain_id,char type,int pos,int &index) const;
  const C_SSPdb* GetSSPdb (int index) const;
  int IsInModel (int model,int &index) const;
  int GetModel (int index) const;
  int IsInChainId (char chain_id,int &index) const;
  char GetChainId (int index) const;
  // Impression ++++
  int PrintPdb(ostream &out) const;
  int Print(ostream &out) const;
};
```

FIGURE 46. Présentation de l'interface usager simplifiée de la classe C_ParserPDB correspondant aux structures 3D contenues dans un fichier de la PDB. Cette classe permet d'extraire du fichier PDB toute l'information 3D.

ANNEXE C

La comparaison de séquences d'acides aminés

Le principe de la phylogénie est que deux séquences quelconques d'une famille proviennent d'une même séquence originale. La Figure 47 présente un

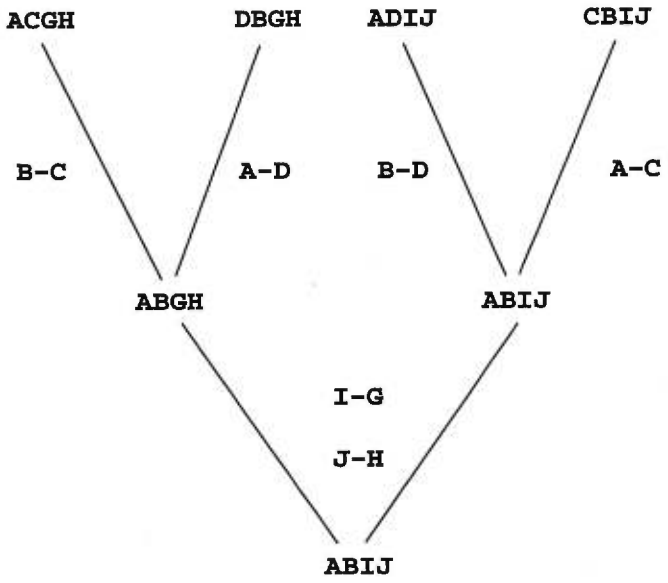

FIGURE 47. Arbre phylogénique simplifié. Quatre séquences différentes sont placées en haut. Des séquences communes inférées (ancêtres) forment les noeuds de l'arbre. Les mutations sont indiquées par les arêtes: X-Y implique que Y est remplacé par X. [14].

exemple d'arbre phylogénique. L'évolution provoque des modifications dans les séquences d'acides aminés. Ces modifications sont appelées *mutations* et peuvent être de trois types : modification d'un acide aminé en un autre, insertion d'un nouvel acide aminé ou délétion d'un acide aminé. En prenant des vraies séquences,

TABLEAU XVI. Matrice des points de mutation acceptés à partir de l'arbre de la Figure 47. [14]

	A	B	C	D	G	H	I	J
A			1	1				
B			1	1				
C	1	1						
D	1	1						
G							1	
H								1
I					1			
J						1		

Dayhoff a établi les fréquences de mutations naturelles. On utilise ces fréquences pour mesurer la distance entre deux séquences quelconques [14]. On établit d'abord la matrice des points de mutation (tableau XVI). On établit aussi la mutabilité d'un résidu avec l'équation C.1

$$m_i = \frac{\text{nombre de mutation}}{\text{fréquence d'apparition dans les séquences}} \quad \text{où } i \text{ représente l'acide aminé} \tag{C.1}$$

La matrice des probabilités de mutation est ensuite définie par l'équation C.2

$$\begin{aligned} M_{ij} &= \frac{\lambda m_j A_{ij}}{\sum_i A_{ij}} \\ M_{jj} &= 1 - \lambda m_j \\ A_{ij} &= \text{élément de la matrice des points de mutation} \\ \lambda &= \text{constante de proportionnalité} \end{aligned} \tag{C.2}$$

Les éléments M_{ij} indique la probabilité que l'acide aminé i soit muté en l'acide aminé j après un certain temps dans l'évolution dénoté λ PAM. L'unité PAM couvre 10^8 années et permet de décrire un pourcentage acceptable de mutations

ponctuelles pour 10^8 années. La matrice M obtenue est appelée matrice de Dayhoff et ses dérivées sont appelées 1 PAM, 2 PAM.

Les matrices de probabilité représentent différentes théories particulières sur l'évolution. Les matrices 1 PAM et 2 PAM n'en sont qu'un exemple. Les matrices les plus près de la "réalité" actuellement sont basées sur l'observation de substitutions dans les séquences disponibles.

Une fois la matrice de probabilité choisie, le problème suivant est de comparer les séquences. Pour ce faire, on calcule la distance qui existe entre une nouvelle séquence et les familles de séquences déjà connues. La distance entre deux séquences est calculée en comparant leur alignement (superposition) optimal qui est trouvé grâce à une métrique qui donne un score à un alignement et un algorithme qui modifie la superposition des séquences. L'alignement optimal est l'alignement qui donne le meilleur score (le plus faible) de la métrique qui tient compte des trois types de mutations. L'algorithme peut insérer des gaps (acide aminé vide) qui représent la délétion ou l'insertion dépendamment de la séquence qu'on étudie, du parent ou de l'enfant dans l'arbre phylogénique. Insérer des gaps dans une séquence induit une pénalité et accroît la distance entre les séquences en augmentant le score de la métrique. Un algorithme d'alignement simple génère toutes les alignements possibles et sélectionne celui qui a le score le plus faible. Vu la taille des séquences, de 50 à des milliers d'acides aminés, il nous faut des algorithmes plus efficaces. L'algorithme de Smith Waterman utilise la technique de la programmation dynamique pour aligner deux séquences.

$$
\begin{aligned}
H_{i,j} &= \max \left\{ \begin{array}{l} H_{i-1,j-1} + w(a_i, b_j), \\ H_{i,j-1} + w(a_i, \Delta), \\ H_{i-1,j} + w(\Delta, b_j) \end{array} \right\} \qquad \text{(C.3)} \\
w(a_i, b_j) &= \text{calcul de la similarité (ex: 1 PAM)} \\
w(a_i, \Delta) &= \text{coût d'insertion d'un gap} \\
n &= \text{nombre d'acides aminés de la première séquence}
\end{aligned}
$$

$$
\begin{aligned}
m &= \text{nombre d'acides aminés de la deuxième séquence} \\
\Delta &= \text{symbole pour un gap} \\
&\text{où } 0 \leq i < n \\
&\quad\ 0 \leq j < m
\end{aligned}
$$

L'équation C.3 présente la relation permettant de construire dynamiquement la matrice H pour obtenir le meilleur score pour un alignement complet de deux séquences qui se trouvent à l'élément $H_{m-1,n-1}$. Cet algorithme modifie une séquence en introduisant des gaps pour décaler un bout de séquence, ce qui entraîne néanmoins une pénalité évaluée par les fonctions $w(a_i, \Delta)$ et $w(\Delta, b_j)$ de la fonction de score. Une substitution fréquemment observée reçoit une pénalité moindre qu'une substitution plus rare [2, 18]. L'algorithme de Smith Waterman est utilisé par la modélisation par homologie pour déterminer à quelle famille de protéines une nouvelle séquence appartient.