Université de Liège
Faculté des Sciences Appliquées
Année Académique 2010 - 2011

Prédiction de structures de macromolécules par apprentissage automatique

Mémoire de fin d'études réalisé en vue de l'obtention du grade d'Ingénieur Civil Electricien

Promoteur : Louis Wehenkel
Jury : Dominique Dehareng
Pierre Geurts
Quentin Louveaux
Francis Maes

Alejandro Marcos Alvarez

6 juin 2011

Remerciements

Le travail de fin d'études constitue le point culminant, l'apogée, de ces nombreuses années de scolarité que l'étudiant a dû affronter. Son achèvement est à la fois un soulagement et un déchirement. Soulagement car il marque la fin d'un processus de longue haleine dans lequel l'étudiant a dû mettre toute sa conviction, tout son courage et toutes ses connaissances. Déchirement car il marque la fin d'une ère qui, bien que parfois maudite, n'est finalement appréciée à sa juste valeur qu'une fois qu'elle s'achève. On l'oublie souvent, mais derrière un tel labeur se cache bien plus que les efforts d'un seul étudiant. Par ce biais, j'aimerais vivement remercier toutes les personnes qui ont rendu cette tâche possible.

Tout d'abord, je tiens à sincèrement remercier le Pr. Louis Wehenkel pour son suivi assidu, ses conseils avisés, son esprit critique et pour toute son aide aussi bien dans le cadre de la réalisation de ce travail qu'en dehors. Il m'a fait entrevoir le monde passionnant de la recherche et pour tout cela, je l'en remercie.

Je tiens également à remercier Francis Maes sans qui ce travail n'aurait pas été possible. Je tiens à le remercier pour tout le temps qu'il m'a consacré, pour ses idées pertinentes et pour son aide, autant technique que théorique, qui m'ont permis de mener à bien ce projet. J'ai pu profiter de son savoir et j'espère en avoir acquis un peu moi-même.

J'aimerais aussi remercier Dominique Dehareng et Pierre Geurts pour les précisions et les corrections qu'ils ont pu apporter à ce document, ainsi que pour leurs suggestions.

Ensuite, je tiens à remercier François Schnitzler et Alexandre Irrthum pour m'avoir mis le pied à l'étrier. Le premier pour m'avoir permis de dompter NIC3 sans qui mon travail n'aurait pas la même envergure, le second pour m'avoir guidé à travers le long processus d'apprentissage de Rosetta qui, sans son aide, se serait avéré beaucoup plus long.

Finalement, je souhaiterais remercier Julien Becker et Fabien Heuze pour leur disponibilité et leur aide venue à point nommé.

Université de Liège
Faculté des Sciences Appliquées
Année Académique 2010 - 2011

Université de Liège

Prédiction de structures de macromolécules par apprentissage automatique

Mémoire de fin d'études réalisé en vue de l'obtention du grade d'Ingénieur Civil Electricien

Alejandro Marcos Alvarez

Résumé

Les protéines sont un constituant essentiel de la vie cellulaire dont l'essentiel des fonctions et des propriétés est déterminé par leur structure tridimensionnelle. Il n'existe pourtant, à l'heure actuelle, aucune méthode permettant de prédire efficacement la structure tridimensionnelle des protéines à partir de la description de leur contenu en acides aminés. Nous proposons ici une approche *ab initio* basée sur le concept du *learning for search*. La prédiction de structure de protéines est modélisée sous la forme d'un problème d'optimisation dans lequel l'algorithme d'optimisation suit un schéma itératif où un opérateur de modification de structure est sélectionné et appliqué à la structure courante. La qualité de la structure ainsi obtenue est ensuite évaluée à l'aide d'un oracle qui déterminera si celle-ci est acceptée. La répétition de ce schéma a pour objet de trouver, à terme, la structure optimale recherchée. Le point critique de ce raisonnement se situe au niveau du choix de l'opérateur qui doit être réalisé très précisément afin d'éviter les écueils classiques des problèmes d'optimisation. C'est cette étape qui fera l'objet de l'apprentissage automatique justifiant ainsi le qualificatif *learning for search* de l'approche proposée.

Le but de ce travail est de prouver que l'apprentissage automatique permet d'améliorer les résultats obtenus via un algorithme d'optimisation naïf. Les résultats de nos expérimentations montrent que cet objectif est atteint. Néanmoins, avant que cette procédure ne soit réellement utile, de nombreux choix doivent être remis en questions aussi bien au niveau de l'algorithme d'optimisation que des procédés d'apprentissage. Finalement, notons que le domaine d'application de ce travail s'étend au-delà de la prédiction de structures de protéines. Il existe de très nombreux problèmes de la littérature scientifique qui sont, à ce jour, encore très mal résolus et pour lesquels aucun algorithme d'approximation n'existe. De tels problèmes pourraient grandement bénéficier d'une approche *learning for search* telle que celle mise en avant dans ce travail.

Table des matières

1 Introduction

1.1 Contexte

Les protéines sont un constituant essentiel de la vie cellulaire. Ces macromolécules sont omniprésentes dans toute cellule et, de ce fait, une bonne compréhension de leurs différents modes d'action est nécessaire pour pouvoir appréhender le fonctionnement des cellules. L'essentiel des fonctions et des propriétés d'une protéine est déterminé par sa structure tridimensionnelle, elle-même déterminée par la séquence d'acides aminés. Ce constat explique que la prédiction de la structure des protéines soit un problème de première importance. Ainsi, malgré quelques décennies de dur labeur, les chercheurs ne sont, à l'heure actuelle, pas encore en mesure de prédire efficacement et précisément la structure tridimensionnelle d'une protéine inconnue. De ce fait, ce domaine de la biochimie et de la bioinformatique, étudié depuis près de 40 ans est en quelque sorte devenu le Saint Graal des sciences biologiques [48].

La prédiction de la structure tridimensionnelle des protéines est actuellement l'objet de recherches intensives aux quatre coins du globe. En effet, en plus de la compréhension des mécanismes biologiques du corps humain qu'elle peut apporter, la prédiction de structure à partir de la séquence d'acides aminés est utilisée dans de nombreux domaines de la recherche médicale tels que le développement de nouveaux médicaments ou enzymes [71, 40]. Obtenir des prédictions de structure fiables est donc une condition *sine qua non* pour que la recherche médicale dans ces domaines puisse avancer.

Les méthodes permettant de prédire la structure tridimensionnelle des protéines peuvent être divisées en deux grandes classes. La première n'utilise que les propriétés physiques afin de générer un repliement et modélise donc le problème sous forme d'un problème d'optimisation pour lequel il faut chercher l'optimum d'une fonction objectif. La seconde approche consiste à utiliser un ensemble de structures connues afin de déduire, plus ou moins fidèlement, la structure de nouvelles protéines. Cette classification n'est pas rigide et de nombreuses méthodes actuelles font appel à des aspects issus des deux approches essayant ainsi de combiner le meilleur des deux mondes. Cependant, malgré la prolifération de ces méthodes de prédiction, très peu permettent d'obtenir des résultats précis et aucune n'est actuellement en mesure de prédire parfaitement une structure ; la difficulté extrême de ce problème explique très probablement ce constat.

1.2 Description des objectifs

Par ce travail, nous allons à notre tour essayer d'apporter une pierre à l'édifice, tout d'abord, en étudiant le problème de manière théorique et, ensuite, en proposant une nouvelle approche basée sur le concept du *learning for search*. Ce terme générique désigne des algorithmes capables d'apprendre des heuristiques qui permettent de parcourir, de manière efficace, l'espace d'état d'un problème améliorant ainsi l'optimisation d'un tel problème.

En vue de générer la structure tertiaire d'une protéine cible, nous utilisons un procédé d'optimisation itératif. A chaque itération, un opérateur est choisi aléatoirement parmi un ensemble vaste d'opérateurs possibles et appliqué à la structure courante. La qualité de la structure ainsi obtenue est ensuite évaluée à l'aide d'un oracle qui déterminera si celle-ci doit

être acceptée ou rejetée. La répétition de ce schéma conduit, à terme et sous certaines conditions, à la structure optimale recherchée. Le point critique de ce raisonnement se situe au niveau du choix de l'opérateur qui doit être réalisé très précisément afin d'éviter les écueils classiques des problèmes d'optimisation. C'est dans cette optique que l'approche *learning for search* sera utilisée. Notre point de départ est une base de données composées d'un ensemble de structures primaires associées aux structures tertiaires correctes correspondantes (typiquement la PDB [1]). A partir de cet ensemble, nous allons générer un deuxième jeu de données qui contiendra, en plus des structures primaires et tertiaires, un ensemble de structures intermédiaires, susceptibles d'être rencontrées lors d'un processus d'optimisation, chacune associée à un ou plusieurs bons opérateurs. C'est finalement ce dernier ensemble de données qui fera l'objet de l'apprentissage automatique dans le but de créer une fonction capable de fournir un bon opérateur en tenant compte de certaines caractéristiques de la structure courante. La méthodologie décrite ici est très générique puisqu'elle est très tolérante vis-à-vis de l'algorithme d'optimisation utilisé, des classes d'opérateurs considérés, des caractéristiques décrivant la protéine et de la manière dont l'ensemble d'apprentissage est généré. Le but de ce travail est de démontrer, en testant la procédure sur un ensemble de petites protéines, sa pertinence et son efficacité en termes d'amélioration de l'exploration de l'espace des conformations possibles de la protéine.

1.3 Organisation du manuscrit

Ce document comporte plusieurs parties qui permettront de suivre le cheminement que nous avons entrepris afin de réaliser ce travail. Tout d'abord, nous commencerons, dans la section 2, par une brève introduction concernant les protéines afin de bien comprendre les aspects qui régissent le problème que nous nous proposons d'étudier. Ensuite, la section 3 détaillera l'état de l'art dans ce domaine dans le but d'apprécier le travail déjà réalisé et pouvoir ainsi comparer notre procédure aux procédures existant. Nous poursuivrons par la section 4 en détaillant les diverses techniques permettant d'évaluer la qualité d'une structure. Ces méthodes d'évaluation s'avèreront utiles au sein même de la procédure que nous développons et pas seulement au moment du contrôle final. Nous expliquerons ensuite dans le détail la méthode que nous avons mise en oeuvre (cf. section 5). Pour finir, après avoir précisé certains éléments concernant la réalisation pratique (section 6), nous nous attacherons à décrire les résultats obtenus (section 7), les problèmes rencontrés (section 8) et les perspectives découlant de notre travail (section 9). Une conclusion terminera ce document (section 10). Nous proposons, en outre, dans les annexes, quelques détails supplémentaires concernant les protéines et une évaluation sommaire de quelques algorithmes d'alignement de structures.

Nous avons, dans ce document, fait de notre mieux afin de constituer une bibliographie aussi pertinente et complète que possible afin de guider le lecteur intéressé vers les ressources adéquates.

1. voir section 2.4.

2 Introduction aux protéines

Les protéines sont un des composants principaux de la cellule. En effet, les protéines, au même titre que les acides nucléiques, les glucides et les lipides, constituent l'essentiel de l'activité moléculaire de la cellule. Plus particulièrement, les protéines exécutent la grande majorité des fonctions de celle-ci parmi lesquelles, nous pouvons citer la transmission de signaux inter et intracellulaires ou le contrôle du passage de molécules au travers de la membrane. Les protéines sont, en outre, les constituants essentiels de la membrane cellulaire et peuvent également agir en temps qu'anticorps, hormones ou fibres élastiques [1, 48]. Au vu de leurs fonctions avancées, il n'est pas surprenant de constater à quel point ces molécules sont complexes. Cette section a pour but de présenter rapidement quelques éléments clés concernant les protéines.

2.1 Acides aminés

Les protéines sont des hétéropolymères dont les éléments de base sont les acides aminés. Il existe 20 acides aminés différents qui ont une structure commune montrée à la figure 1. Cette structure est composée d'un atome central de carbone, appelé C_α, d'un groupe amine et d'un groupe carboxyle. Les acides aminés se différencient les uns des autres par la chaîne latérale indiquée sur la figure 1 par la lettre R. Le cas de la proline est particulier par le fait que l'extrémité de sa chaîne latérale est attachée au C_α, remplaçant ainsi l'hydrogène habituel. Il s'agit alors d'un acide aminé cyclique. Les acides aminés se divisent en 4 catégories suivant la nature de leur chaîne latérale. Celle-ci peut être non polaire, polaire et non chargée, polaire et chargée négativement (acide ionisée à pH 7) ou polaire et chargée positivement (basique ionisée à pH 7). La table 1 liste les différents acides aminés classés par catégories et l'annexe A, page 83, montre les structures de ceux-ci.

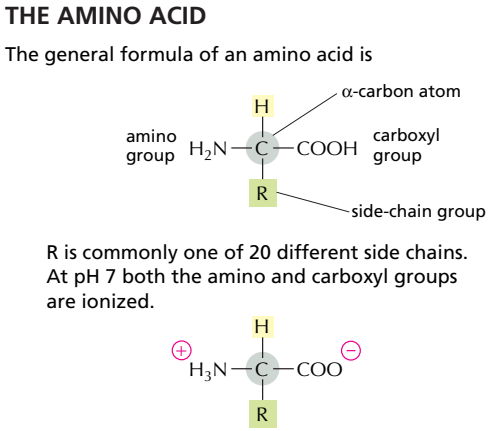


FIGURE 1 – Structure générique des acides aminés [1]

Les acides aminés sont reliés entre eux par des liens peptidiques, voir figure 2. Ces liens se forment lorsque les groupes carboxyle d'un acide aminé et amine d'un autre réagissent ensemble. La réaction libère une molécule d'eau et crée une liaison covalente entre le carbone du premier acide aminé et l'azote du second. La structure ainsi formée porte le nom de

Catégorie	Acide aminé
Acide	Acide aspartique (D), acide glutamique (E)
Basique	Lysine (K), arginine (R), histidine (H)
Polaire non chargé	Sérine (S), thréonine (T), asparagine (N), glutamine (Q), cystéine (C), tyrosine (Y)
Non-polaire	Glycine (G), alanine (A), valine (V), leucine (L), isoleucine (I), proline (P), méthionine (M), phénylalanine (F), tryptophane (W)

TABLE 1 – Classification des acides aminés

dipeptide. La répétition d'une telle réaction à l'une ou l'autre extrémité du dipeptide produira une chaîne d'acides aminés qui, finalement, constituera la protéine. La séquence des acides aminés formant la protéine s'appelle la structure primaire. La nature de la chaîne latérale influence la structure tridimensionnelle finale de la protéine. En effet, les chaînes latérales des acides aminés non-polaires sont hydrophobes et ont tendance à se regrouper en un coeur hydrophobe au sein de la protéine et à laisser les acides aminés hydrophiles à l'extérieur de celle-ci où ils sont accessibles au solvant. Ceci est illustré à la figure 3.

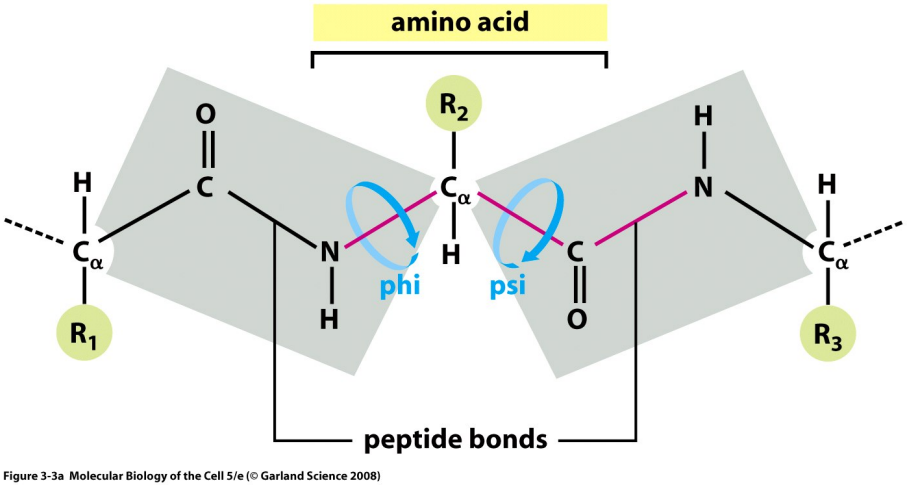


FIGURE 2 – Liens peptides entre acides aminés [1]

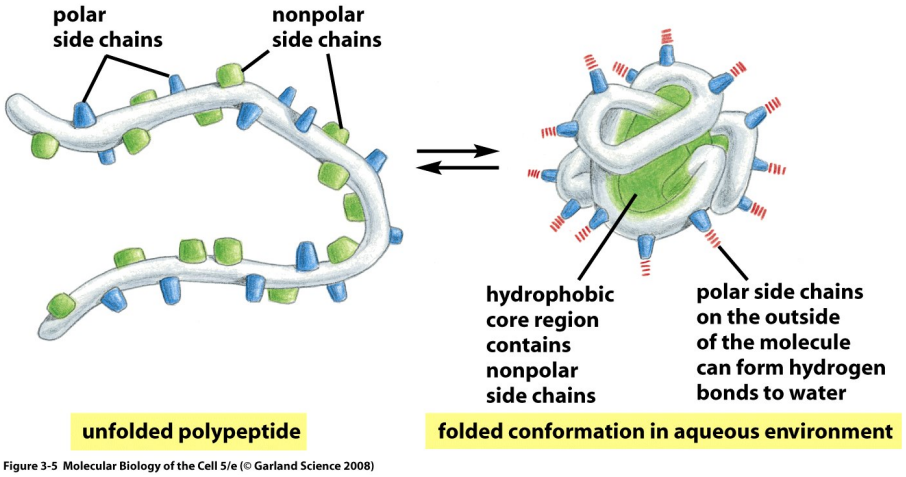


FIGURE 3 – Coeur hydrophobe d'une protéine [1]

2.2 Structures

La structure primaire d'une protéine, c'est-à-dire sa séquence en acides aminés, détermine sa structure tertiaire. Elle contient donc toute l'information nécessaire au repliement de celle-ci [1, 48, 5, 4, 11].

Le deuxième niveau de structure est appelé structure secondaire et définit les différentes structures tridimensionnelles locales que l'on peut observer sur des sous-ensembles d'acides aminés de la structure primaire. Parmi les différentes structures secondaires, les plus connues sont les hélices α et les brins β. Ces deux structures secondaires résultent de liaisons hydrogène entre un atome d'oxygène du groupe carboxyle d'un acide aminé et un atome d'hydrogène du groupe amine d'un autre acide aminé de la chaîne principale. Ces structures ne font intervenir que les atomes de la chaîne principale, ce qui explique leur grande fréquence d'observation. Dans le cas des hélices α, les acides aminés contigus dans la séquence faisant partie de l'hélice adoptent une structure de forme éponyme. Celles-ci se forment lorsque des liaisons hydrogène apparaissent entre deux acides aminés séparés par trois autres. Ces structures sont très stables d'une protéine à l'autre et présentent généralement des pas à droite de 3.6 acides aminés. Une telle hélice α est représentée à la figure 4. L'hélice α est la structure qui est adoptée par environ 25% des résidus présents dans les protéines [48].

Les brins β sont quant à eux des structures plus ou moins planes qui peuvent se regrouper pour former des feuillets très rigides. Les liaisons hydrogène apparaissent ici entre deux brins β au sein d'un même feuillet. Ceux-ci peuvent être soit parallèles soit antiparallèles suivant l'orientation de la chaîne principale à l'emplacement du feuillet. La figure 5 représente un feuillet β détaillé en plusieurs brins tandis que la figure 6 illustre les deux types de feuillets β. Les feuillets β sont également très fréquents au sein des protéines puisque, de manière similaire aux hélices α, environ 25% des résidus des protéines se structurent sous forme de feuillets β.

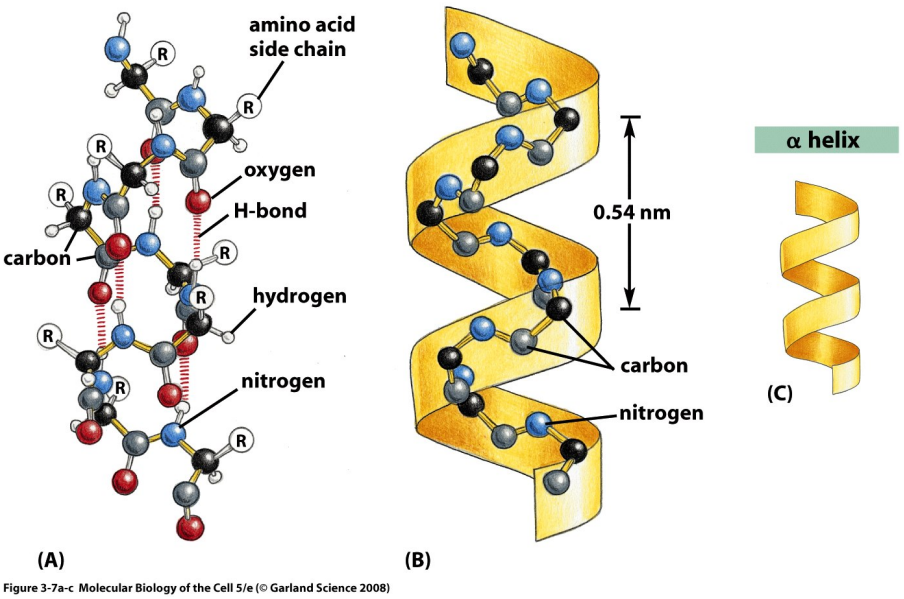


FIGURE 4 – Représentation d'une hélice α [1]

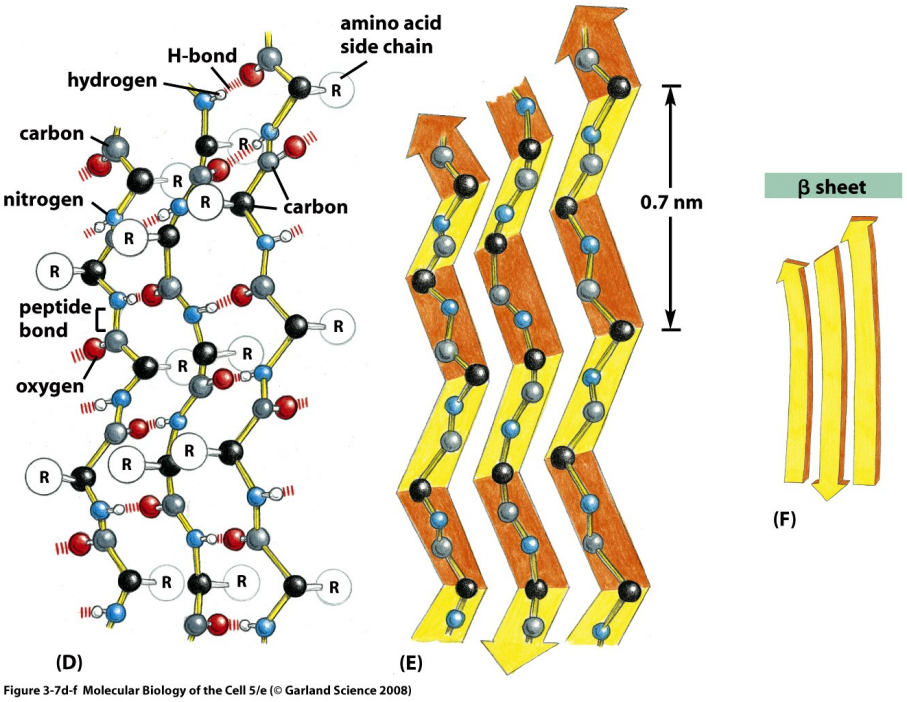


FIGURE 5 – Représentation d'un feuillet β et de ses brins β [1]

Le dernier niveau de structure est appelé la structure tertiaire et représente la structure tridimensionnelle de la protéine repliée. Cette structure est d'une extrême importance puisque les extraordinaires propriétés que possèdent les protéines dépendent de manière non négligeable de leur structure tertiaire [1, 48], c'est la raison pour laquelle la détermination de cette

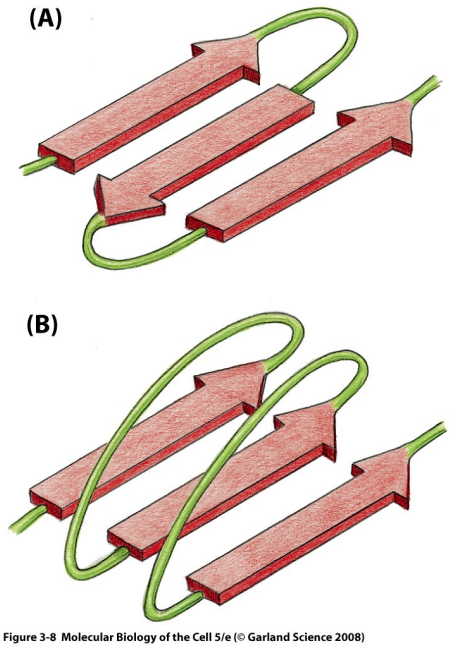


FIGURE 6 – Représentation de deux feuillets β. (A) antiparallèle (B) parallèle [1]

structure est si importante. Les structures tertiaires des protéines peuvent être classifiées en trois grandes catégories : les protéines α qui sont composées en grande partie d'hélices α, les protéines β pour lesquelles les structures secondaires sont essentiellement des feuillets β et les protéines α - β qui possèdent des hélices qui alternent avec des feuillets le long de la séquence [26]. La classe α contient de petites protéines et est la classe la plus petite parmi les trois. Certains des membres de cette classe sont en fait des motifs qui se répètent dans la classe α - β comme, par exemple, la structure globine qui se retrouve dans la myoglobine et dans l'hémoglobine. La classe β contient les feuillets β parallèles et antiparallèles qui peuvent se replier selon différentes structures qui sont observées dans d'autres protéines plus volumineuses. Finalement, la classe α - β contient le plus grand nombre de protéines, ainsi que les plus volumineuses, dans lesquelles alternent hélices et feuillets afin de former des structures encore plus complexes. Un exemple de protéine α - β repliée est illustré à la figure 7. Il s'agit ici du domaine SH2 qui a un rôle important dans la transmission de signal dans les cellules eucaryotes.

Les protéines peuvent se replier de nombreuses manières différentes. Néanmoins, on sait que, comme dans toute molécule, certaines forces très énergétiques limitent les mouvements possibles de la protéine qui ne peut donc se déplacer que de manière contrainte. Ces contraintes sont essentiellement les liaisons covalentes qui relient chaque atome de la protéine afin que celle-ci ne disloque pas. La question qui se pose est donc de savoir quels degrés de liberté permettent à la protéine d'adopter autant de conformations différentes. Ces degrés de liberté sont les angles de torsion entre les liaisons covalentes, dont la variation demande beaucoup moins d'énergie. Les trois angles de torsion principaux relatifs à la conformation de la chaîne principale sont les suivants. L'angle Φ est l'angle de torsion autour du lien qui lie l'atome d'azote au carbone C_α au sein d'un même acide aminé. Le deuxième angle est l'angle Ψ qui

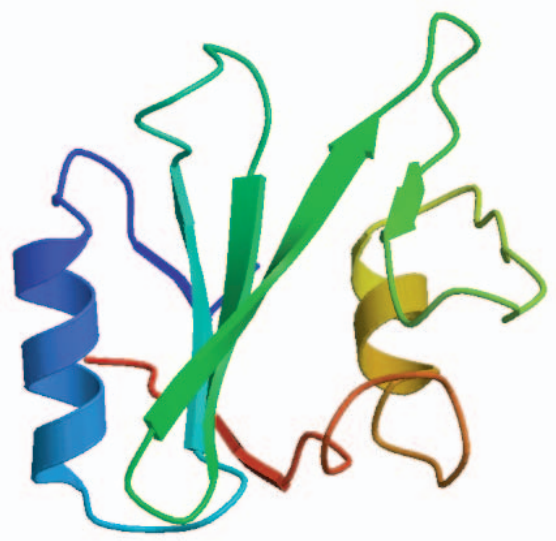


FIGURE 7 – Exemple de la structure tertiaire du domaine SH2 [1]

représente la torsion autour de la liaison établie entre le C_α et le carbone du groupe carboxyle d'un même acide aminé. Finalement, l'angle ω caractérise la rotation de deux résidus adjacents puisqu'il représente la torsion autour du lien peptidique. Ces trois angles permettent à eux seuls d'obtenir un grand nombre de structures tertiaires différentes. Il est évident que d'autres angles de torsion existent, mais ceux-ci sont de moindre importance. Les angles Φ et Ψ peuvent être visualisés sur la figure 2.

2.3 Fonction d'énergie

Les lois de la mécanique stipulent qu'un système évolue toujours vers son état de plus faible énergie afin d'atteindre un équilibre stable. Les mêmes principes s'appliquent aux molécules en général qui, dans leur état le plus stable, se situeront au point de plus faible énergie qu'elles peuvent atteindre. En effet, tout changement de structure tridimensionnelle est associé à une variation d'énergie liée aux variations des interactions entre les différents atomes constitutifs. Cette énergie est fonction de la position relative des atomes. Cette quantité peut donc être utilisée pour caractériser la stabilité d'un polypeptide. En d'autres termes, plus l'état de la protéine sera stable, plus son énergie sera faible. D'un point de vue purement mathématique, cette fonction peut être vue comme l'objectif d'un problème que l'on désire minimiser.

L'état le plus stable d'une protéine est appelé sa conformation native, celle-ci correspond à la conformation présente naturellement dans les organismes vivants et au point de plus faible énergie. Les interactions de Van der Waals, les liaisons hydrogène et les ponts disulfures ne sont que des exemples d'interactions agissant sur la cohésion d'une protéine. La fonction d'énergie possède en fait un grand nombre de termes qui peuvent être résumés comme suit

[48]

$$G = U - TS + pV \tag{1}$$

où T, p et V représentent respectivement la température, la pression et le volume du système. L'énergie G porte le nom d'énergie libre de Gibbs et possède deux composantes principales. La première est U et correspond à l'énergie interne du système, la seconde S représente l'entropie du système, c'est-à-dire le désordre de celui-ci. L'état stable du système est obtenu lorsque la fonction G est minimale, ce qui revient à minimiser U et maximiser S, si on néglige les effets du dernier terme. La première condition est remplie lorsque de nombreuses interactions favorables existent au sein de la protéine, ce qui a pour effet de stabiliser celle-ci et d'augmenter son niveau d'organisation. Inversement, la seconde condition est respectée lorsque le niveau de désordre est maximal, et donc lorsque la protéine a un faible niveau d'organisation [48]. La stabilité résultera donc d'un compromis entre ces deux conditions. Le calcul de l'énergie s'effectue en deux étapes. La première consiste à calculer l'énergie interne de la molécule et la seconde mesure l'entropie en échantillonnant l'espace des conformations possibles.

2.3.1 Energie interne

L'équation (2) représente une formule semi-empirique typique permettant de calculer la fonction d'énergie [48].

$$\begin{aligned} U &= \frac{1}{2}\sum_{b} K_b \left(b - b_0\right)^2 + \frac{1}{2}\sum_{\theta} K_\theta \left(\theta - \theta_0\right)^2 + \sum\sum K_n \left(1 + \cos\left(n\phi - \delta_n\right)\right) \\ &\quad + \sum_{i<j}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{4\pi\varepsilon r_{ij}}\right) \end{aligned} \tag{2}$$

Le premier terme représente l'énergie due aux compressions ou extensions des liaisons chimiques. La liaison est ici représentée comme un ressort et son énergie est évaluée, pour des petites déformations, grâce à la loi de Hooke [48]. Les angles entre tout triplet d'atomes consécutifs au sein d'une molécule stable sont généralement fixés à des valeurs relativement bien connues qui dépendent de la nature des atomes en présence. La valeur de l'angle formé résulte en fait de la minimisation de l'énergie de la molécule. C'est pourquoi, tout triplet d'atomes, dont l'angle dévie de la valeur d'équilibre, contribue à l'augmentation de l'énergie interne de la molécule. Cet effet est modélisé par le deuxième terme de l'équation (2). Contrairement à l'énergie provenant de l'extension ou de la compression des liaisons, il n'existe pas de consensus concernant la forme de l'énergie due aux angles de liaison, la forme utilisée dans l'équation (2) est une forme parmi d'autres utilisées dans la littérature [48]. Le troisième terme correspond à l'énergie due aux variations des angles de torsion. La forme de l'expression modélisant cette énergie est différente des deux précédentes parce que cette énergie est périodique, de période 2π.

Le dernier terme de l'équation (2) tient compte des interactions entre atomes non liés chimiquement. Les deux types d'interactions considérées sont les interactions de Van der Waals et les interactions électrostatiques. Ces dernières sont modélisées simplement en utilisant la

loi de Coulomb entre deux atomes portant chacun une charge nette. L'énergie potentielle due à l'interaction de deux charges est, pour rappel, donné par l'équation (3)

$$U_{\text{electrostatique}} = \frac{q_i q_j}{4\pi\varepsilon r_{ij}} \tag{3}$$

où q_i et q_j représentent la charge portée par chaque atome, ε la permittivité du milieu dans lequel les atomes baignent et r_{ij} la distance entre les deux atomes considérés [48, 22].

Les interactions de Van der Waals, quant à elles, sont modélisées par un potentiel de Lennard-Jones dont la forme est reprise à l'équation (4)

$$U_{\text{Van der Waals}} = \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \tag{4}$$

où A_{ij} et B_{ij} sont des paramètres qui dépendent de la nature des atomes intervenant et r_{ij} est la distance les séparant [48, 22].

2.3.2 Entropie

L'entropie d'une protéine en solution peut être calculée en deux étapes. La première consiste à calculer l'entropie du solvant et la seconde à calculer l'entropie du soluté. L'estimation de l'entropie de ces deux systèmes est généralement obtenue par dynamique moléculaire [48, 62].

2.3.3 Fonction d'énergie utilisée

Au vu de ce qui précède, il apparaît clairement que le calcul de l'énergie d'une protéine est une opération qui est loin d'être triviale. Afin de faciliter la réalisation de ce travail, nous utiliserons donc une fonction d'énergie déjà implémentée, à savoir celle proposée dans la suite logicielle Rosetta [2] [55]. Rosetta calcule l'énergie d'une structure en effectuant une somme pondérée de termes énergétiques (interactions de Van der Waals attractives et répulsives, liaisons hydrogène, énergie électrostatique, ponts disulfures, etc.) et offre, de plus, un grand nombre de pondérations prédéfinies. Dans le cadre de ce travail, nous avons utilisé la pondération *standard*. Malgré nos recherches, il nous a été impossible d'obtenir des explications détaillées sur la manière dont l'énergie était calculée par Rosetta. Nous utiliserons donc la fonction d'énergie comme une boîte noire fournissant la valeur souhaitée.

2.4 Protein Data Bank

Depuis que les protéines sont étudiées, de nombreuses structures déterminées expérimentalement par cristallographie ou par résonance magnétique nucléaire ont été rendues publiques. Afin de centraliser toutes ces informations, une base de données mondiale a été créée Celle-ci regroupe plus de 70000 structures partielles ou complètes. Cette base de données porte le nom de Protein Data Bank ou PDB [9] et contient également des structures de séquences d'acides nucléiques (ADN ou ARN). Les protéines sont regroupées en son sein sous forme de fichiers .pdb qui possèdent une syntaxe spécifique permettant de décrire la structure de la protéine à laquelle le fichier correspond. La figure 8 illustre un extrait du fichier 1VWT.pdb

2. `http://www.rosettacommons.org/`

qui représente l'hémoglobine humaine dans son état T. Cet extrait caractérise les résidus numérotés 16 et 17 dans la séquence primaire. Chaque ligne correspond à un atome pour lequel certaines informations sont données : indice de l'atome, nom de l'atome, type et indice du résidu auquel il appartient, coordonnées cartésiennes de l'atome, etc. Le format .pdb est aujourd'hui largement utilisé dans la communauté scientifique. Nous fournissons, ci-dessous, des explications à propos de certaines colonnes.

- Colonne 1 : indique que la ligne courante décrit un atome ;
- Colonne 2 : indice de l'atome dans la protéine ;
- Colonne 3 : nom de l'atome dans le résidu ;
- Colonne 4 : type de résidu auquel appartient l'atome ;
- Colonne 6 : indice, dans la séquence primaire de la protéine, du résidu auquel appartient l'atome ;
- Colonnes 7 à 9 : coordonnées cartésiennes (x, y, z) de l'atome courant ;
- Colonne 12 : élément chimique de l'atome dans le tableau périodique des éléments.

Les colonnes décrites ci-dessus sont celles qui sont les plus utiles dans notre application, une description complète de ce format est disponible à l'adresse suivante `www.wwpdb.org/docs.html#format`.

1	2	3	4	5	6	7	8	9	10	11	12
ATOM	110	N	LYS	A	16	43.128	10.069	6.644	1.00	36.08	N
ATOM	111	CA	LYS	A	16	43.925	8.851	6.631	1.00	35.68	C
ATOM	112	C	LYS	A	16	43.413	7.879	5.573	1.00	33.98	C
ATOM	113	O	LYS	A	16	44.191	7.121	4.997	1.00	34.17	O
ATOM	114	CB	LYS	A	16	43.897	8.185	8.010	1.00	37.34	C
ATOM	115	CG	LYS	A	16	44.482	6.776	8.039	1.00	42.12	C
ATOM	116	CD	LYS	A	16	45.953	6.752	7.634	1.00	47.48	C
ATOM	117	CE	LYS	A	16	46.792	7.704	8.484	1.00	51.04	C
ATOM	118	NZ	LYS	A	16	48.239	7.338	8.480	1.00	52.09	N
ATOM	119	N	VAL	A	17	42.103	7.893	5.327	1.00	32.83	N
ATOM	120	CA	VAL	A	17	41.519	7.014	4.317	1.00	30.96	C
ATOM	121	C	VAL	A	17	42.221	7.384	3.020	1.00	31.11	C
ATOM	122	O	VAL	A	17	42.668	6.518	2.262	1.00	28.25	O
ATOM	123	CB	VAL	A	17	39.994	7.252	4.156	1.00	31.58	C
ATOM	124	CG1	VAL	A	17	39.499	6.596	2.871	1.00	28.70	C
ATOM	125	CG2	VAL	A	17	39.240	6.689	5.367	1.00	31.25	C

FIGURE 8 – Résidus 16 et 17 de l'hémoglobine dans son état T, 1VWT.pdb

3 Etat de l'art dans le domaine de la prédiction de structure de protéines

Cette section suit la même structure que celles utilisées pour présenter l'état de l'art dans l'article *Protein structure prediction* de P. Koehl [48] ainsi que dans le livre *Structural Bioinformatics* de P. Bourne et H. Weissig [11].

La recherche concernant les structures tridimensionnelles des protéines a mis en évidence le fait que, bien que les protéines se comptent par dizaines de milliers, il n'existe en fait qu'un nombre restreint de repliements. De nombreuses protéines, bien que différentes sur le plan de la composition et des fonctions, ont donc des conformations globalement semblables. Le problème de la prédiction de structure tridimensionnelle d'une protéine se pose donc d'abord en termes de reconnaissance de repliement ou *fold recognition* [11]. Si le repliement est évident, par exemple dû à une forte similarité de la structure primaire, le problème devient alors celui de la détermination précise de la structure en utilisant les informations disponibles pour les protéines dont la structure est connue et le repliement semblable à la protéine cible. C'est ce que l'on appelle la prédiction par homologie ou *comparative modeling* ou encore *homology modeling*. Si, par contre, aucune similarité entre la protéine étudiée et les protéines existant dans les bases de données ne peut être identifiée, le problème devient tout autre puisqu'il ne peut plus être résolu de la même manière. Il porte alors le nom de *de novo modeling*. Bien qu'aucune structure connue similaire n'existe, les informations contenues dans les bases de données peuvent néanmoins être utilisées afin de prédire la structure de la protéine cible. Ces informations seront, par contre, exploitées différemment que dans le cas où une réelle similarité a pu être identifiée (homology modeling). Parmi les méthodes utilisables afin de prédire le repliement d'une protéine qui n'a pu être associée à une protéine dont la structure est déjà connue, une classe notoire de techniques n'utilise que les principes de base de la physique pour modéliser le repliement et obtenir la conformation du polypeptide cible. Ces méthodes sont connues sous le nom de *ab inito modeling*.

La prédiction de structure de protéines est un domaine de recherche unique. En effet, la manière dont les progrès sont évalués est inédite puisque cette évaluation se fait sous forme d'une compétition : le CASP. Ce concours peut ainsi être considéré comme un benchmark qui permet d'évaluer les méthodes proposées par les chercheurs actifs dans le domaine, offrant ainsi une mesure quantitative de l'évolution de la recherche. Le CASP permet également d'orienter les chercheurs dans de nouvelles voies prometteuses en déterminant les limitations et les points forts des méthodes actuelles. Une description plus détaillée du CASP est proposée à la section 4.

3.1 Homology modeling

La prédiction par homologie permet de prédire la structure d'une protéine par inférence à partir de structures de protéines connues. Elle est basée sur deux constats simples [11] :

- la séquence des acides aminés détermine à elle seule la structure de la protéine,
- la structure des protéines est stable au cours de l'évolution et change moins que les séquences d'acides aminés correspondantes. Cela implique que des séquences avec un certain degré de similarité mènent à la même structure tridimensionnelle. Cette observation a bien sûr des limites qui ont été quantifiées. Une courbe qui définit la *safe zone*

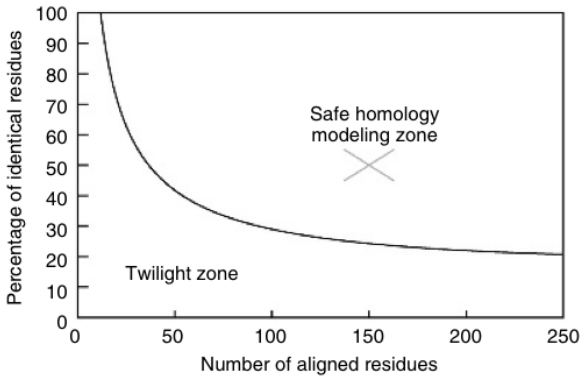


FIGURE 9 – Twilight et safe zone [11]

a ainsi été identifiée [75] et est montrée à la figure 9. Les paires de protéines qui peuvent être placées dans cette zone auront des structures tertiaires similaires. De plus, l'étendue des changements structurels est directement reliée à l'étendue des changements au sein de la séquence [17, 29, 76, 79].

La qualité de cette méthode repose sur trois conditions [48] :

1. l'existence de, au moins, un homologue parmi les protéines dont les structures sont connues,
2. la fiabilité avec laquelle les homologues sont identifiés,
3. la qualité de l'inférence à partir des modèles identifiés.

Les avancées en génomique (découvertes de nouvelles séquences et développement de nouveaux algorithmes de comparaison de séquences) ont largement contribué à l'amélioration des deux premiers critères de qualité [21]. De très nets progrès ont été réalisés dans les techniques permettant de trouver des séquences homologues et le principal facteur limitatif reste, à ce jour, l'alignement de ces séquences [48]. L'inférence à partir des modèles identifiés est encore problématique comme nous le verrons par après et certaines étapes, comme le loop modeling, restent jusqu'à présent relativement mal résolues. Néanmoins, on peut dire que certaines régions sont mieux prédites que d'autres, c'est le cas notamment des régions biologiquement importantes qui sont généralement mieux prédites que la structure dans sa globalité [84]. Finalement, notons que ces méthodes possèdent un défaut inhérent à leur nature. En effet, les méthodes basées sur des données expérimentales sont sujettes aux erreurs de même type présentes dans la PDB [9].

3.1.1 Détails de la procédure

La prédiction par homologie est un procédé itératif qui peut être résumé en quatre étapes [48] : recherche de séquences similaires, alignement des séquences similaires trouvées, création d'un modèle pour la protéine cible et évaluation du modèle produit.

Etape 1 : Recherche de séquences similaires La prédiction par homologie utilise des protéines dont la structure est supposée proche de la structure cible afin de générer une prédiction pour la protéine étudiée. Etant donné que la similarité entre les protéines ne peut être évaluée via leur structure, c'est la séquence d'acides aminés qui servira de mesure entre les protéines. Aux premières heures de la prédiction par homologie, les algorithmes utilisés pour trouver des protéines semblables étaient BLAST [2] et FASTA [70]. Ces méthodes effectuent des comparaisons par paires de résidus entre la séquence cible et les séquences contenues dans la PDB [9] et permettent de trouver des protéines présentant un haut degré de similarité.

La safe zone, représentée sur la figure 9, contient les protéines dont le taux de similarité des séquences est supérieur à 40% [17, 75]. La twilight zone contient, quant à elle, les séquences dont le taux d'identification est compris entre 20 et 35%. Finalement, la midnight zone, non représentée sur la figure, contient les paires de protéines dont l'identification est inférieure à 10%. La corrélation entre les structures tertiaires des paires de protéines classées dans la twilight zone est nettement inférieure à celle des paires contenues dans la safe zone. Plus précisément, plus de 90% des protéines dont la similarité de séquence est supérieure à 30%, au sens des algorithmes précédemment cités, possèdent des structures tertiaires semblables. Ce nombre descend sous la barre des 10% lorsque l'on considère des paires de protéines contenues dans la midnight zone. Malgré ce constat, il existe de nombreuses protéines dont la structure est semblable en dépit d'un faible niveau d'identification [75]. Cet état de fait a conduit au développement de nouveaux algorithmes d'alignement qui permettent de détecter de manière plus sensible la similarité de deux séquences. Les méthodes d'alignement multiple sont les méthodes qui se sont avérées les plus efficaces dans ce domaine. Ces algorithmes procèdent en alignant non pas deux, mais de multiples séquences simultanément. L'idée principale de cette nouvelle approche consiste à comparer une famille plutôt qu'une paire de séquences dans l'espoir que, bien que la paire étudiée soit fortement différente, l'étude de la famille mettra en évidence des relations entre les séquences qui n'étaient pas, de prime abord, triviales. PSIBLAST [3] et HMMER [25] sont des exemples d'algorithmes utilisant ce principe.

Le résultat des algorithmes précédents est un ensemble de protéines dont la séquence est similaire à celle de la protéine cible. Il convient alors de choisir au mieux celle qui sera utilisée comme base pour la prédiction de la nouvelle structure. Le choix se porte généralement sur la protéine dont le niveau de similarité avec le polypeptide étudié est le plus élevé, mais d'autres critères doivent être pris en compte comme la qualité des structures expérimentales disponibles, par exemple. Notons que certaines méthodes, comme MODELLER [77, 78], permettent d'utiliser plusieurs modèles afin d'améliorer la qualité des prédictions.

Etape 2 : Alignement des séquences La plupart des algorithmes de recherche de séquences similaires produisent aussi des alignements. Hélas, ces résultats ne sont pas toujours fiables et sont même parfois complètement erronés [48, 56]. Afin d'assurer la qualité des prédictions, des méthodes spécifiques doivent être utilisées en vue d'effectuer l'alignement des séquences [11]. Pour des séquences de forte similarité, les algorithmes existant produisent essentiellement les résultats corrects. Il en est autrement lorsque le taux d'identification diminue et les résultats des diverses méthodes commencent à différer. Il est alors utile de varier les sources et de considérer, non pas un, mais plusieurs alignements obtenus à partir de divers algorithmes [61]. Le résultat de cet alignement sera la superposition des séquences considérées

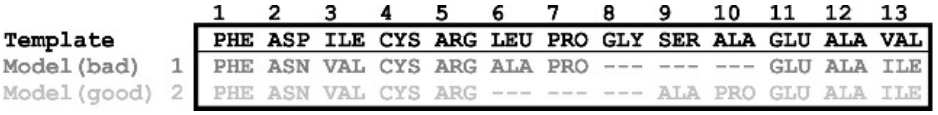


FIGURE 10 – Alignement de séquences [11]

avec, à certains endroits, des trous qui indiquent une position à laquelle les deux séquences ne correspondent pas suite à l'ajout ou au retrait d'un certain nombre de résidus. Ceci est illustré à la figure 10 où l'on voit que, pour produire un bon alignement, un trou doit être inclus dans la séquence de la protéine dont on désire prédire la structure.

Etape 3 : Construction du modèle Lorsqu'un alignement entre les séquences est disponible, la création de la prédiction peut commencer. Celle-ci se déroule généralement en trois étapes : création de la structure aux endroits où les séquences sont alignées et donc où un exemple de structure est disponible, modélisation de la structure aux positions où il n'existe pas d'alignement de séquence (*loop building*) et prédiction de la position des chaînes latérales. Ces différentes étapes sont illustrées à la figure 11.

Création de la chaîne principale Une fois les alignements connus, la création du squelette de la protéine, là où une correspondance de résidus a été identifiée, est très simple. Il suffit de copier les coordonnées des résidus faisant partie à la fois du modèle et de la cible. Si les résidus diffèrent, seules les coordonnées du squelette (N, C_α, C et O) sont copiées, sinon on peut également copier les coordonnées de la chaîne latérale [11].

Loop building L'étape de création du squelette de la protéine laisse presque inévitablement des trous. Ces trous peuvent apparaître soit dans la protéine cible, soit dans le modèle. Dans le premier cas, des résidus du modèle sont manquants dans la séquence cible et on parle dans ce cas de suppression. La solution consiste alors à supprimer les résidus du modèle afin de coller à la séquence désirée. Dans le second cas, c'est le modèle qui possède trop peu de résidus par rapport à la cible et il faut alors introduire les résidus manquant au bon endroit dans le modèle. Dans les deux cas, cela introduit une modification de la conformation du squelette de la protéine. Notons que les structures secondaires régulières, hélices α et feuillets β, ne sont pas affectées par ce problème, c'est pourquoi les suppressions et ajouts apparaissent en dehors de ces structures. A défaut de placer les suppressions et ajouts dans les hélices ou les feuillets, ceux-ci se situeront dans les boucles et les virages. Hélas, les changements de squelette à ces endroits sont extrêmement complexes à modéliser et restent, à ce jour, un des problèmes les plus mal résolus dans le contexte de la prédiction par homologie. Deux approches sont possibles pour modéliser les boucles : la première consiste à chercher dans la PDB les boucles les plus proches et à les copier directement dans notre protéine cible, la deuxième approche, que l'on appelle *ab initio*, sélectionnera la conformation qui minimise l'énergie de la boucle [48, 11].

Prédiction des chaînes secondaires Une fois que la structure de la chaîne principale est fixée, il reste à prédire la conformation des chaînes latérales. Des analyses ayant été réalisées sur la PDB ont mis en évidence le nombre fini de conformations observées pour ces

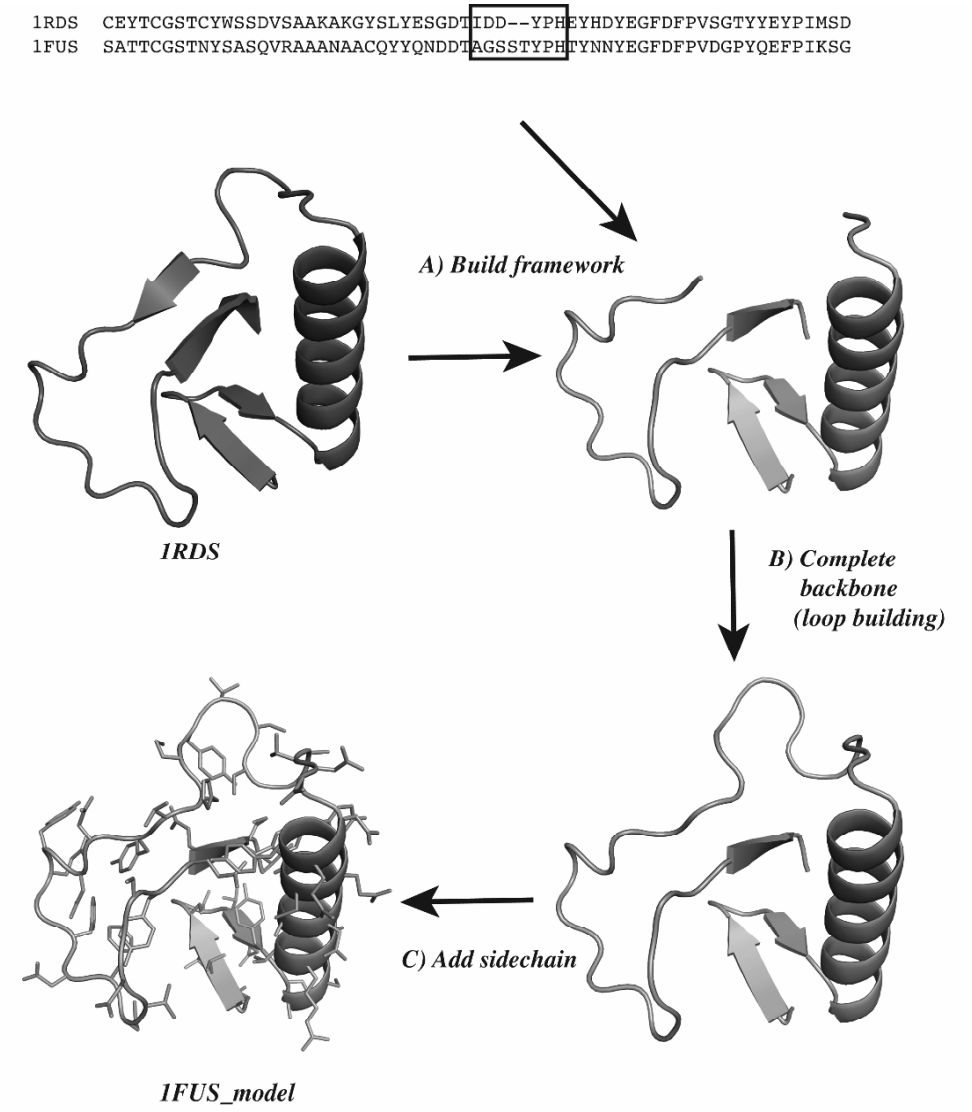


FIGURE 11 – Les trois étapes du homology modeling [48]

chaînes [11, 73, 23]. Celles-ci portent le nom de rotamères. La répétition des rotamères dans la PDB explique que la simple copie de ces chaînes latérales sur la chaîne principale fournit de meilleurs résultats que ceux obtenus lorsque la conformation des chaînes latérales est prédite par d'autres moyens. En pratique, cela n'est valable que pour des séquences qui présentent un haut degré de similitude et dont les chaînes latérales forment donc un réseau interconnecté. Les rotamères extraits de la PDB sont classés dans des bibliothèques qui sont parcourues afin de sélectionner le rotamère qui minimise un ensemble de fonctions d'énergies. Si ce travail doit être réalisé pour chaque résidu, l'espace de recherche devient exponentiellement grand, rendant impossible une exploration simple. Il a été prouvé que ce problème est NP-complet [72] et même qu'il n'existe aucun algorithme d'approximation pour ce problème [15]. L'espace de recherche peut néanmoins être fortement diminué si l'on tient compte de la conformation du squelette qui limite fortement le choix des rotamères possibles. De nombreuses méthodes ont vu le jour afin de résoudre le problème de la prédiction de structure des chaînes latérales [48], l'algorithme le plus connu est SCWRL3 [14] qui est utilisé par de nombreux groupes concourant dans CASP.

Raffinement du modèle Aussi précises que soient les étapes précédentes, le modèle obtenu n'atteindra pas la précision d'une structure déterminée par cristallographie. Afin d'obtenir la prédiction la plus précise possible, une étape de raffinement est nécessaire et ce d'autant plus que la similarité entre les séquences est faible. Cette étape est généralement menée à bien via des méthodes qui ne dépendent plus des structures de références. Une approche typique consiste à minimiser une fonction d'énergie, telle celle donnée dans l'équation (2), soit par des méthodes numériques classiques, soit par simulation de dynamique moléculaire [48]. Afin de garantir la convergence de ces méthodes sur la totalité d'une protéine, la fonction d'énergie doit être d'une très grande précision. Une telle fonction n'est actuellement pas disponible. Typiquement, une étape d'optimisation qui utilise une fonction d'énergie peu précise aura comme conséquence la suppression de quelques grosses erreurs et l'apparition de nombreuses petites qui, en s'accumulant, auront tendance à faire diverger la structure cible de la structure réelle, c'est pourquoi ces étapes sont généralement limitées en nombre afin de ne pas perdre la conformation initialement obtenue. En somme, l'optimisation du modèle est pour l'instant limitée par la précision des champs de force [48, 11]. Les champs de force font actuellement l'objet de recherches intenses afin d'améliorer la précision des fonctions d'énergie. Deux approches semblent actuellement prometteuses :

1. **Méthodes semi-empiriques quantiques [59] :** ces méthodes prennent en compte le nuage électronique de la molécule au niveau quantique. Néanmoins, la résolution des équations appliquées à un très gros système implique d'utiliser des approximations. Dans ce cas-ci, elles sont basées sur des paramétrisations d'un grand nombre de termes, modélisés sur base de calculs précis sur de petits systèmes. Ces méthodes ont l'avantage de prendre en compte les variations d'interaction liées à la déformation du nuage électronique. Elles diffèrent entre elles par le type d'approximations considérées.
2. **Champ de force auto-paramétré :** la précision d'un champ de force dépend, en grande partie de ses paramètres. Or ceux-ci sont généralement déterminés via des calculs de chimie quantique sur de petites molécules et ensuite adaptés pour correspondre aux données disponibles. Ces diverses approximations détériorent la précision des champs de force. La procédure utilisée ici consiste à ajouter les paramètres du champ de force aux variables de l'optimisation. Ainsi, un paramètre du champ de force est changé

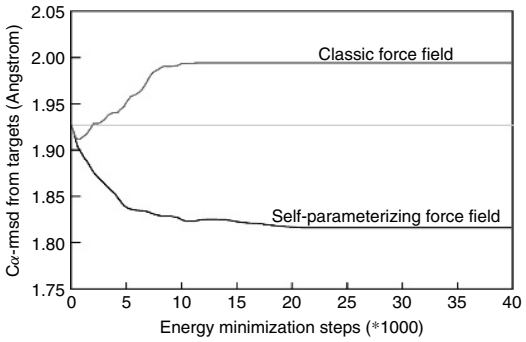


FIGURE 12 – Amélioration du champ de force auto-paramétré [11]

aléatoirement, ensuite, une minimisation d'énergie est appliquée. Si le résultat obtenu est meilleur que le précédent, on garde le nouveau champ de force, sinon on reprend le précédent. Cette approche permet de modifier le champ de force dans le sens d'une amélioration des résultats. Ceci est illustré sur la figure 12 où l'on voit clairement que, contrairement à un champ de force classique, les champs de force auto-paramétrés permettent d'améliorer les résultats. Les deux champs de force améliorent la prédiction pendant les 500 premières étapes. Mais ensuite, l'accumulation de petites erreurs dues à l'imprécision du champ de force classique fait dévier la prédiction de la structure cible. Ce n'est pas le cas pour les champs auto-paramétrés qui évoluent dans le sens d'une diminution du RMSD. Malheureusement, les résultats obtenus grâce à ces méthodes sont encore très éloignés des résultats expérimentaux et ce procédé est très coûteux en termes de temps de calcul [11].

Etape 4 : Validation du modèle La validation du modèle est essentielle avant d'extraire de l'information à partir de la structure cible créée. En effet, il est important de s'assurer que la conformation obtenue soit plausible avant d'essayer d'en extraire la moindre information. Les erreurs obtenues dans les structures prédites par homologie dépendent fortement de la similarité entre la séquence étudiée et les séquences des modèles. Différents moyens existent pour évaluer un modèle. Par exemple, la stéréochimie de la protéine peut être analysée en contrôlant, par exemple, les longueurs et angles de liaisons, la planarité des liens peptidiques, le fait que ces liaisons sont « trans » et non « cis », ou encore en vérifiant que deux atomes ne sont pas trop proches et que la protéine ne viole pas de manière excessive les contraintes géométriques connues. Il existe de nombreux programmes permettant d'analyser la stéréochimie des protéines comme PROCHECK [51] ou encore WHATCHECK [39]. Un autre moyen d'évaluer la qualité d'une structure de protéine prédite consiste à établir les propriétés qu'elle possède et qui sont communément rencontrées chez les protéines. Ces propriétés sont, par exemple, la présence d'un coeur hydrophobe, le positionnement correct des résidus polaires, les liaisons hydrogène ou encore les structures secondaires. Les programmes VERIFY3D [27] et PROSA [86] sont deux exemples de méthodes qui permettent de tester la qualité d'une prédiction sur les critères susmentionnés. D'autres critères peuvent encore bien évidemment être utilisés.

3.2 Fold recognition

Le problème du fold recognition consiste à trouver quelle structure connue est la plus à même de correspondre à la structure de la protéine cible. Ce problème est également connu sous le nom de problème de repliement inverse ou encore de *threading*. Ceci peut être réalisé, par exemple, en comparant la séquence de la protéine inconnue avec la structure des protéines de référence et en calculant, par la suite, des scores qui quantifient la compatibilité d'un acide aminé avec un environnement donné dans la structure [48]. Ce problème est devenu de plus en plus important au fil du temps car, avec l'augmentation du nombre de structures tertiaires connues, il est devenu clair que des protéines, même très éloignées sur le plan de la séquence primaire, pouvaient avoir des repliements similaires. La justification de ces méthodes repose donc sur l'existence d'un nombre relativement restreint de repliements. Il en résulte que l'espace des structures des protéines est sensiblement plus petit que l'espace des séquences. On estime à environ un millier le nombre de repliements différents existant dans les protéines [16]. Ce nombre rend possible la création de bases de données relativement complètes et de tailles limitées qui regroupent des individus de tous les repliements connus. Le procédé de fold recognition est donc un moyen alternatif et complémentaire pour trouver des séquences similaires utilisables lors de la prédiction par homologie pour créer la structure cible.

Il existe plusieurs explications possibles qui justifient le faible nombre de repliements [11]. La plus évidente se base sur les fonctions des protéines. En effet, leur structure est souvent liée à leurs fonctions biologiques, c'est-à-dire que certaines fonctions particulières comme, par exemple, la liaison à un type de substrat, ne sont possibles que si la protéine possède une structure tertiaire spécifique. C'est pourquoi des protéines dont les fonctions sont similaires peuvent avoir des structures tertiaires proches. De plus, la publication abondante de nouvelles séquences a mis en évidence le fait que des protéines de fonctions, et donc de structures, similaires mais provenant d'espèces animales différentes ont des séquences similaires. Cette distance entre les séquences augmente d'autant plus que les espèces sont éloignées dans l'arbre de l'évolution. On peut donc raisonnablement penser que ces protéines ont des ancêtres communs. L'évolution de l'ancêtre à la protéine moderne peut être résumé en une série de mutations, d'insertions ou de suppressions dans la chaîne d'acides aminés. Nous voyons donc ici apparaître la dualité entre la séquence et la structure qui avait déjà été mise en évidence par le homology modeling. Le but du fold recognition est donc de trouver des similarités de structure tertiaire lorsqu'aucune similarité de séquence n'a pu être identifiée. On le voit, le fold recognition et la prédiction par homologie sont étroitement liés et la délimitation entre les deux problèmes évolue au fil des avancées dans le domaine de l'analyse des séquences.

3.2.1 Détails de la procédure

Il existe principalement deux classes de méthodes de fold recognition. Ces classes se distinguent par leur approche du problème. La première approche, que l'on appellera biologique, est basée sur la description, la classification et finalement l'analyse des chemins de mutations et donc de l'évolution de la protéine. Des protéines de même conformation sont classifiées dans des catégories similaires et l'entièreté des individus d'une catégorie est utilisée pour définir les critères d'adhésion à cette catégorie. La deuxième approche, physique quant à elle, repose sur les lois fondamentales de la physique pour établir une correspondance entre les repliements des différentes protéines. Dans ce cas, des protéines de même repliement sont considérées

comme étant deux solutions possibles d'un problème de minimisation. Chaque approche permet de répondre facilement à certaines questions, mais la combinaison des deux perspectives est nécessaire pour pouvoir répondre à la plupart des problèmes que les scientifiques essaient de résoudre.

Approche biologique Le problème se décompose en deux étapes. Premièrement, une similarité doit être établie entre la protéine cible et les protéines connues, ce qui peut, si les protéines sont fort éloignées, être relativement difficile. Ensuite, dans un deuxième temps, l'alignement détaillé entre les résidus doit être construit. L'analyse de la similarité entre les séquences se fait au moyen de méthodes proches de celles utilisées en traitements des signaux. Ces méthodes sont généralement les mêmes que celles utilisées dans l'étape 1 de la prédiction par homologie, voir section 3.1.1. Malgré la performance de ces algorithmes, tous se heurtent au même problème. La similarité entre deux séquences est d'autant plus difficile à déceler que ces séquences sont éloignées. Il est, dans certains cas, très difficile, voire impossible, de déterminer si deux protéines sont réellement homologues ou si le résultat obtenu est un faux positif. Dans cette zone, appelée la *twilight zone*, les algorithmes sont incapables de déceler une vraie similarité d'une analogie fortuite. Cette zone a déjà été rencontrée et est, pour rappel, représentée à la figure 9. Les protéines sont classées en familles. On peut logiquement penser que des mutations qui altèrent fortement la structure ou les fonctions d'une protéine n'existent pas à l'état naturel. L'analyse d'une famille permet donc de détecter les mutations que chaque protéine a subies mettant ainsi en évidence les acides aminés qui jouent un rôle prépondérant dans les fonctions de la protéine. Un exemple simple consiste à imaginer qu'un acide aminé qui mute fréquemment d'une protéine à l'autre peut être considéré comme participant peu à la structure de la protéine. La détermination des résidus réellement importants permet d'améliorer les alignements de séquence même au sein de la *twilight zone*. Ces algorithmes sont connus sous le nom d'algorithmes d'alignement de multiples séquences, voir également section 3.1.1. Il existe de nombreuses implémentations de ce type d'algorithmes, par exemple PSI-BLAST [3], qui diffèrent principalement sur trois points. Le problème d'alignement simultané de multiples séquences est un problème NP-hard [44]. Pour le résoudre, la plupart des algorithmes utilisent donc des heuristiques qui sont différentes suivant l'algorithme que l'on considère : procédures de constructions hiérarchiques [3], création d'un arbre phylogénétique [83], etc. Les algorithmes diffèrent également sur la manière dont ils analysent les alignements obtenus. Certains calculent par exemple la moyenne des compositions des positions alignées tandis que d'autres maximisent l'information contenue à chaque position [3]. Finalement, la troisième différence apparaît dans le calcul de la similarité entre une famille et une séquence. Il est intéressant de noter que bien que les formulations mathématiques puissent paraître fort différentes, les méthodes utilisent des concepts essentiellement identiques et fournissent des résultats sensiblement équivalents. Ces méthodes de reconnaissance de repliement supposent toutes que la similarité de structure provient de la similarité de séquence. Les structures de référence ne sont donc ici que peu utilisées.

Approche physique L'approche physique combine, pour sa part, l'utilisation des structures connues de la PDB[9] et de la fonction d'énergie propre à la protéine étudiée. Les principaux désavantages des méthodes *ab initio* sont connus, voir section 3.3. L'approche physique du fold recognition tente de pallier l'inconvénient de la taille de l'espace de recherche en effectuant une recherche par grille [11]. Cette recherche consiste à évaluer l'énergie de la protéine

lorsque celle-ci prend des conformations particulières qui sont basées sur les structures de protéines connues. La méthode force la protéine à prendre la conformation d'une autre et son énergie est ensuite évaluée. Cette procédure est appelée *threading* [13, 34] et a donné naissance à de nombreux algorithmes [13, 33, 41, 53, 43]. La première étape consiste, comme dans l'approche biologique, à aligner les séquences afin de sélectionner les protéines les plus à même de correspondre à la structure de la protéine cible. Etant donné que de grandes bases de données doivent être parcourues, il est important que la fonction d'énergie soit optimisée en termes de vitesse d'évaluation. Afin d'atteindre cet objectif, la structure tridimensionnelle de la protéine est simplifiée, le nombre d'interactions possible est limité et certains degrés de liberté sont moyennés, par exemple les vibrations des liaisons ou la position des molécules du solvant. Cette dernière approximation, bien que répercutée sur la précision de la fonction d'énergie, est en fait très bien adaptée à l'étude de longs processus comme le repliement. Le score, basé sur l'énergie, utilisé par le threading pour comparer deux structures est un score global, qui contrairement au score local (comparaison de résidus position par position) utilisé dans l'approche biologique, mène à un problème NP-complet et à l'impossibilité d'utiliser les algorithmes rapides de programmation dynamique [11, 26]. Pour résoudre ce problème, une première méthode a consisté à utiliser des algorithmes d'alignement qui ne nécessitaient pas de mesure de similarité locale. Cette approche n'a été implémentée que par peu de groupes [13] étant donné la complexité algorithmique énorme du problème et les temps de calcul colossaux que cela implique. Une autre solution consiste à utiliser un algorithme de programmation dynamique à deux niveaux afin d'optimiser les interactions de chaque paire de résidus alignés en ne considérant que celles entre les résidus les plus fortement dépendants [43]. Finalement, une dernière méthode consiste à approximer le calcul de la fonction d'énergie en gelant les partenaires d'interaction aux positions du modèle et ceux-ci ne sont mis à jour qu'une fois l'alignement réalisé [33]. Les différences essentielles entre les algorithmes de threading se répartissent en trois catégories : modèle de la protéine et approximation de la fonction d'énergie, paramétrage de la fonction d'énergie et algorithmes d'alignement utilisés (programmation dynamique, programmation dynamique à deux niveaux, optimisation de Monte Carlo ou encore branch-and-bound). Etant donné les similarités de procédure entre le threading et la prédiction par homologie, les sources d'erreur sont similaires. La plus importante est bien sûr qu'une protéine de structure tertiaire adéquate doit exister dans la PDB pour pouvoir créer une prédiction correcte. Ensuite, la précision sera limitée par la qualité du modèle et par la distance existant entre la cible et le modèle. Finalement, les imprécisions de la fonction d'énergie ainsi que les approximations faites pour faciliter le calcul de l'alignement sont deux sources d'erreur non négligeables. Ces algorithmes, bien que performants lors du CASP, voir section 4.1, n'aident pas à comprendre les mécanismes de repliement des protéines.

3.3 *ab initio* modeling

Les méthodes de prédiction *ab initio* sont utilisées lorsque les méthodes précédentes sont inapplicables parce qu'aucune protéine similaire à la protéine cible ne peut être identifiée dans la PDB. A l'opposé de la prédiction par homologie, les méthodes *ab initio* ne créent pas la nouvelle structure en inférant à partir d'une structure sélectionnée comme modèle. Celles-ci utilisent plutôt les principes fondamentaux de la physique et de la chimie afin de calculer le repliement de la cible [57, 11, 48]. Notons que nombre de méthodes *ab initio* utilisent néanmoins des informations recueillies auprès des structures déjà résolues afin d'améliorer les prédictions obtenues. Les résultats publiés par Christian Boehmer Anfinsen et par de

nombreux chercheurs actuels [5, 4, 11], suggèrent que, en l'absence de grandes barrières de potentiel dans la fonction d'énergie, la structure native de la protéine correspond à son minimum global d'énergie libre. Quelques exceptions à ce principe ont néanmoins été découvertes et une description plus détaillée peut être trouvée dans [6]. Les méthodes *ab initio* reposent sur ce paradigme.

Pour pouvoir exploiter ce constat, les méthodes *ab initio* nécessitent deux composants essentiels : une fonction d'énergie fiable dans un voisinage des minima d'énergie de la structure native et un algorithme de recherche pour explorer l'espace des conformations. Malgré le potentiel de ces méthodes, les fonctions d'énergie disponibles à ce jour sont trop imprécises et les algorithmes d'exploration trop inefficaces que pour pouvoir fournir des prédictions réellement intéressantes sur de grandes protéines. Le nombre de conformations d'une protéine est extrêmement élevé. Le paradoxe de Levinthal, du nom de Cyrus Levinthal qui l'a énoncé en 1969 [58], estime ce nombre à environ 10^{300}, pour une protéine de 150 acides aminés. Un parcours restreint de l'espace des conformations est donc nécessaire à la détermination d'une solution, ne serait-ce qu'approximative. Malgré la complexité de ces méthodes, celles-ci sont relativement importantes dans des domaines tels que la conception de nouveaux médicaments ou d'enzymes non naturelles pour lesquels la prédiction par homologie n'est donc pas utilisable.

3.3.1 Détails de la procédure

Il existe de nombreux niveaux de représentation d'une protéine allant du plus complet au plus simple. Le premier considère l'entièreté des atomes de la protéine ainsi que toutes les molécules du solvant, mais un tel niveau de détail peut être inutile en particulier dans les premières phases du repliement. De nombreuses approximations existent et ont pour but d'accélérer le calcul de la fonction d'énergie. L'influence des molécules du solvant peut, par exemple, être modélisée très simplement [11] ou bien les chaînes latérales peuvent être entièrement remplacées par leur centroïde [82]. Des études ont démontré que les chaînes latérales se trouvent à l'état naturel dans un nombre limité de conformations [23]. Ce constat peut également être utilisé afin de simplifier la représentation de la protéine en fixant les conformations des chaînes latérales aux conformations les plus fréquemment observées.

Il existe essentiellement deux types de fonction d'énergie. La première catégorie est issue de la mécanique moléculaire. Ce type de fonction modélise les interactions en utilisant des termes d'énergie simples qui ont été paramétrés pour de petites molécules ou par calculs quantiques dans le vide [11]. L'autre catégorie de fonctions de potentiels est directement dérivée des structures expérimentalement connues qui sont contenues dans la PDB [9]. Ce type de fonctions est particulièrement utile pour modéliser les relations qui ne sont pas encore bien comprises théoriquement comme les effets hydrophobes. Nous allons maintenant détailler diverses approches *ab initio* qui ont donné lieu à des méthodes de prédiction de protéines.

Modèles en treillis Les modèles en treillis proposent une manière simple de représenter la géométrie d'une protéine. Chaque noeud du treillis correspond à une position possible d'un acide aminé, cette position pouvant être occupée ou non. Le treillis limite ainsi grandement le nombre de conformations possibles et permet d'explorer l'espace des conformations de manière relativement plus efficace. Les longueurs et angles de liaison sont ainsi discrétisés en un nombre fini relativement faible de valeurs qui dépendent de la structure du treillis. Celui-ci peut être

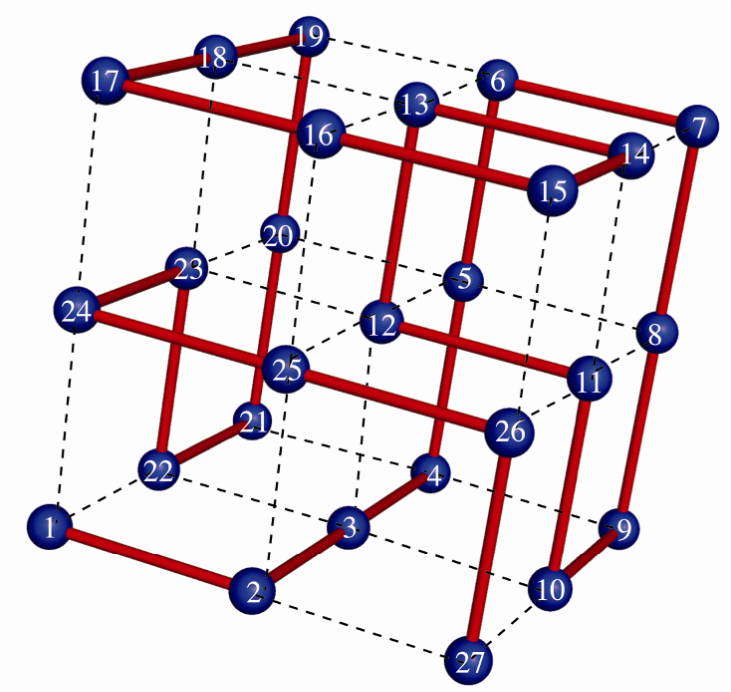


FIGURE 13 – Représentation d'un modèle en treillis pour une protéine de 27 acides aminés [48]

bi ou tridimensionnel. Le treillis tridimensionnel le plus simple est le treillis cubique où les acides aminés ne peuvent être disposés qu'aux sommets des cubes définissant le treillis. Un autre modèle tridimensionnel commun est le treillis cubique centré sur les faces (face-centered) dans lequel les acides aminés peuvent se placer au centre des faces du cube en plus de ses sommets. Ce dernier modèle de treillis permet d'obtenir de meilleures approximations quant à la géométrie des protéines [19, 69]. Les algorithmes qui modélisent les conformations possibles sous forme de treillis peuvent ensuite utiliser différents potentiels pour mesurer l'interaction entre résidus. Le modèle HP [54] par exemple ne modélise les acides aminés que selon leur caractère polaire (P) ou hydrophobe (H). Un contact HH dans le treillis est favorable, tandis que les contacts HP et PP sont neutres. D'autres modèles considèrent les 20 types d'acides aminés et modélisent leurs interactions via des potentiels de Go [32] ou via des potentiels statistiques [66]. L'avantage principal des modèles en treillis est leur simplicité, tandis que leur principal défaut est la perte de résolution engendrée par le treillis. Ces modèles sont amenés à disparaître au profit de modèles plus réalistes en raison de l'augmentation de la puissance de calcul des ordinateurs. Un exemple de structure en treillis cubique pour une protéine de 27 acides aminés est représenté à la figure 13.

Prédiction basée sur les principes fondamentaux Cette approche n'utilise que des principes physiques afin de prédire la structure des protéines cibles. Aucune information provenant des bases de données n'est utilisée dans la méthode, par exemple les fonctions d'énergie statistiques sont exclues. La seule hypothèse effectuée est que la conformation native correspond au minimum global de la fonction d'énergie. L'approche principalement utilisée pour

résoudre ce problème est la dynamique moléculaire qui intègre numériquement les équations de mouvement de Newton afin de simuler l'évolution dynamique de la protéine. La dynamique moléculaire permet d'explorer l'espace des conformations de la protéine en modifiant la structure tridimensionnelle de sorte que celle-ci converge vers un minimum d'énergie. Le grand nombre d'atomes et la taille des pas d'intégration, qui doivent être choisis relativement petits afin d'assurer la stabilité numérique, rendent compliquée l'utilisation de telles techniques dans le cas des protéines. Ces méthodes sont principalement limitées par la précision des champs de force et par la quantité colossale de puissance de calcul nécessaire à la simulation. C'est pourquoi, dans le but d'accélérer le procédé de recherche, des méthodes permettant d'explorer l'espace des conformations de manière plus grossière sont parfois utilisées. Un exemple de ce type de modification consiste à modifier les angles de torsion ϕ et/ou ψ d'un ou plusieurs résidus, ce qui va avoir pour effet de modifier rapidement la structure afin d'explorer l'espace de recherche de manière plus rapide. Un désavantage de cette approche est que les modifications peuvent s'avérer trop importantes et amener de trop grands changements de structure dans le reste de la protéine.

Approche bioinformatique Par opposition au modèle précédent, de l'information est ici extraite des bases de données afin d'assister la fonction d'énergie dans le repliement de la protéine. Ceci est généralement réalisé en trois étapes. La première consiste à prédire les structures secondaires tout le long de la protéine. La deuxième s'attarde à générer un ensemble de conformations supposées correctes pour la protéine, celles-ci portent le nom de *decoys* en anglais. La dernière étape consiste à sélectionner, parmi les *decoys* créés, la conformation la plus à même de correspondre à la conformation native. Les trois étapes sont brièvement décrites ci-dessous.

Génération de la structure secondaire La première étape du repliement naturel des protéines consiste généralement en une agglomération du coeur hydrophobe [48]. La première étape réalisée ici n'est donc pas la même que l'étape naturelle. Néanmoins, ce choix est justifié par le gain en information que les structures secondaires peuvent apporter pour la suite. Les premières méthodes permettant de déterminer à quel type de structure secondaire appartient un résidu sont connues sous le nom de méthode Chou et Fasman [18] et de méthode GOR [31]. Ces méthodes sont basées sur des calculs de probabilités, à partir desquelles des propensions sont calculées pour chaque acide aminé afin de déterminer si celui-ci est plus à même d'appartenir à l'une des trois structures secondaires considérées : les hélices α, les feuillets β ou les virages (turns). Soit $p_S(i)$ la probabilité de trouver l'acide aminé i d'un type donné dans la structure secondaire S, et soit $p(i)$ la proportion d'acides aminés i parmi les protéines connues, alors la propension est donnée par

$$P_S(i) = \frac{p_S(i)}{p(i)}. \tag{5}$$

Si cette valeur est supérieure à 1, alors l'acide aminé de type i a une préférence pour la structure secondaire S. Si la valeur est inférieure à 1, alors l'acide aminé i a tendance à éviter la structure S. Si, par contre, la valeur est proche de 1, l'acide aminé n'a pas de préférence. Un petit traitement, qui n'est pas détaillé ici, est ensuite appliqué aux propensions calculées afin de prédire la structure secondaire de la protéine. Plus de détails peuvent être trouvés dans [48]. Bien que ces méthodes aient fortement évolué au fil du temps, elles ont été dépassées, en

termes de qualité de prédiction, par des méthodes de recherche de multiple séquence. A l'instar de l'alignement, la prédiction de structure secondaire est améliorée lorsque l'on considère le regroupement de protéines semblables en familles.

Echantillonnage des conformations Une grande partie des structures secondaires d'une protéine est généralement regroupée au sein du coeur de la protéine. Beaucoup de méthodes font usage de cette hypothèse afin de générer les *decoys*. Les structures secondaires prédites précédemment sont utilisées par les algorithmes qui les agencent de diverses manières en un coeur compact. Plusieurs méthodes effectuant cette tâche existent. Les agencements des structures secondaires peuvent être créés à partir de ceux qui sont le plus couramment observés dans les bases de données [28, 12], ils peuvent également être choisis sur base d'un critère d'énergie qui sélectionne les agencements dans lesquels les interactions entre structures secondaires sont les plus énergétiquement favorables [87] ou encore générés par échantillonnage de Monte-Carlo ou par des méthodes de recuit simulé. Ces méthodes reposent sur une prédiction préalable de la structure secondaire qui n'est pas toujours fiable et n'imposent aucune contrainte sur les boucles rendant ainsi peu fiable la prédiction de celles-ci. C'est pourquoi de nouvelles méthodes d'échantillonnage de conformations ont vu le jour. C'est le cas notamment des approches à base de fragments. Ces méthodes consistent à prédire la structure de la protéine, segment de résidus par segment de résidus. Différentes conformations pour chaque segment sont essayées et la structure ainsi produite est évaluée grâce à un score qui prend en compte les caractéristiques globales de cette structure telles que la compacité, l'agglomération hydrophobique ou encore les ponts disulfures. Les conformations des segments sont sélectionnées selon une loi probabiliste inférée à partir des conformations observées pour ces mêmes segments dans les bases de données. Un algorithme de recherche de type Monte-Carlo est ensuite utilisé avec cette loi de probabilité afin de générer les structures candidates. Cette méthode est utilisée avec succès dans la suite logicielle Rosetta [74] et a permis de prédire avec une grande résolution les protéines de petites tailles (inférieures à 85 résidus) [48].

Evaluation des *decoys* Afin de ne sélectionner, parmi les *decoys* générés, que les meilleurs, ceux-ci sont évalués grâce à des fonctions de score. Les deux classes principales de fonction de score sont celles basées sur la physique des interactions moléculaires et celles basées sur la connaissance (*knowledge based*). La première catégorie fut déjà discutée dans la section 2.3 et évalue simplement l'énergie du système. Les interactions intramoléculaires sont, de nos jours, relativement bien modélisées. La compréhension que nous avons des interactions extérieures, avec le solvant ou avec des ions, est par contre beaucoup moins bonne. Afin de gérer ces interactions de manière efficace pour un grand nombre de *decoys*, des modèles sont généralement nécessaires, introduisant, de ce fait, des imprécisions dans la fonction de score [48]. La seconde classe de fonction de score est dérivée des bases de données des structures connues et est représentée sous forme de fonctions de probabilités qui déterminent si les caractéristiques de la structure cible sont plausibles ou non. Le score $E_{\mathrm{C}}(x, S)$ caractérisant une propriété C prenant une valeur x dans un sous-ensemble S de la protéine peut alors être exprimé comme suit

$$E_{\mathrm{C}}(x, S) = -k \ln \left(\frac{P_{\mathrm{C}}(x|S)}{P_{\mathrm{C}}(x)} \right). \tag{6}$$

Dans l'équation (6), $P_{\mathrm{C}}(x)$ représente la probabilité, évaluée sur l'entièreté de la base de données, que la propriété C prenne une valeur x. De façon similaire, $P_{\mathrm{C}}(x|S)$ représente la probabilité que la propriété C prenne une valeur x dans un sous-ensemble S des protéines. Ces probabilités sont estimées grâce aux structures connues que l'on peut trouver dans les bases de données de structures de protéines. Ces fonctions de score sont ensuite évaluées pour les différents *decoys* candidats afin de sélectionner la ou les meilleures conformations qui ont été générées par l'algorithme.

4 Méthodes d'évaluation de structures prédites

Le but de ce travail est de développer des algorithmes qui prédisent, de manière aussi précise que possible, les structures tertiaires des protéines. Une comparaison entre les méthodes développées et les méthodes existantes est donc nécessaire pour déterminer si la nouvelle approche améliore les prédictions obtenues. De nombreuses méthodes existent à cette fin et toutes ont des avantages et des inconvénients. Nous présentons ici un certain nombre de ces méthodes en précisant celles que nous utiliserons dans le cadre de ce travail.

4.1 Critical Assessment of Techniques for Protein Structure Prediction

Le CASP qui est l'acronyme de *Critical Assessment of Techniques for Protein Structure Prediction*[3] est un concours qui a lieu tous les deux ans et qui définit un cadre strict pour l'évaluation de différentes méthodes de prédiction. Le concours se divise en fait en plusieurs catégories dont les deux principales sont le *template based modeling* et le *template free modeling*. La première catégorie contient les structures à prédire pour lesquelles un ou plusieurs homologues sont présents dans les bases de données, cette catégorie utilisera typiquement le homology modeling pour effectuer ses prédictions. La seconde catégorie ne contient, par contre, que des protéines pour lesquelles aucun homologue n'est disponible et permet ainsi d'évaluer des méthodes autres que le homology modeling. Evidemment, les prédictions de cette seconde catégorie seront entachées de plus d'erreurs, ce qui se reflète dans CASP par des méthodes d'évaluation différentes.

L'évaluation des structures prédites qui ont été soumises au CASP se divise en deux phases distinctes [20, 46, 7]. Premièrement, une évaluation automatique est réalisée. Cette première étape permet en fait de décanter les prédictions et de ne soumettre à la deuxième phase que les structures qui sont réellement intéressantes. La deuxième phase consiste en l'évaluation par un expert de la qualité de la prédiction. Cette évaluation se base aussi bien sur des calculs effectués par l'expert que sur son appréciation visuelle de la qualité de la structure.

Pour déterminer la meilleure façon dont nous pouvons évaluer nos structures, nous allons prendre comme point de départ les systèmes d'évaluation utilisés lors du CASP et les modifier légèrement afin qu'ils soient aisément utilisables par nos soins. En effet, au cours du processus d'évaluation, de nombreux algorithmes, allant du plus simple au plus complexe, sont utilisés afin de sélectionner les meilleures prédictions. Néanmoins, les algorithmes utilisés ne sont pas tous disponibles en libre accès. C'est pourquoi, afin d'éviter une réimplémentation totale de ces méthodes, les procédures définies par le CASP seront adaptées à nos besoins.

De nombreuses mesures de similarité entre structures tridimensionnelles imposent une superposition préalable des structures, c'est pourquoi, avant de discuter le choix des mesures utilisées (voir section 4.3), nous allons discuter le choix de l'algorithme de superposition.

3. `http://predictioncenter.org/`

4.2 Algorithmes de superposition de structures tridimensionnelles

4.2.1 Généralités et algorithmes envisagés

Les algorithmes de superposition sont divers et les méthodes utilisées par chacun diffèrent plus ou moins suivant l'algorithme considéré. L'efficacité de ces algorithmes, ainsi que leur temps de calcul peut varier très fortement de l'un à l'autre. Pourtant, il est difficile de réellement les classifier. En effet, certaines méthodes sont plus aptes à superposer les structures secondaires comme les hélices α ou les feuillets β [63], tandis que d'autres sont plus performants à d'autres niveaux de la superposition. Le choix de l'algorithme dépendra aussi bien de l'application visée que de la simplicité de mise en oeuvre de la procédure.

De nombreux algorithmes ont été envisagés comme candidats potentiels : LGA [91], MAMMOTH [68], Dali [38], TM-align [93], SCALI [90], FATCAT [88] et MATT [63]. Néanmoins, bien que les articles présentant toutes ces méthodes soient accessibles, seul un certain nombre d'entre eux peut être aisément incorporé dans un code C++. En effet, une fraction non négligeable de ces algorithmes sont implémentés sous la forme de serveurs accessibles via internet et permettant d'effectuer des prédictions, ou encore sous la forme de logiciels standalone ne donnant pas directement accès à toutes les fonctionnalités offertes. Nous avons donc dû éliminer d'emblée Dali et LGA qui n'étaient disponibles que sous la forme de serveurs. Parmi les algorithmes restant, seul MATT donne un accès libre et aisé à ses sources. MATT est discuté plus en détails dans l'annexe B. Hélas, bien que l'algorithme MATT se soit avéré, après analyse, fortement intéressant pour notre application, nous avons dû nous résoudre, pour des raisons d'interface entre différents codes sources, à utiliser une autre méthode de superposition décrite dans la section 4.2.2.

4.2.2 Algorithme de superposition utilisé

La méthode de superposition utilisée est une analyse procustéenne (Procrustes analysis) qui permet de comparer deux ensembles de points en effectuant uniquement des translations, des rotations, des mises à l'échelle et des combinaisons de ces trois transformations. Cet algorithme a l'avantage d'être relativement simple et bien compris. La superposition produite minimise la moyenne des carrés des distances entre résidus (RMSD). Biologiquement parlant, ce critère ne permet pas d'effectuer les meilleures superpositions. Néanmoins, la qualité de la superposition n'est pas d'une importance primordiale dans ce travail étant donné les critères de comparaison de structures que nous utiliserons (cf. section 4.3.4).

L'algorithme procustéen est assez simple, il fonctionne en deux étapes dans notre cas [4] : faire coïncider les barycentres des deux ensembles de points, ici les positions dans l'espace des carbones α, et ensuite appliquer une rotation afin de minimiser le RMSD entre les deux structures. Le calcul de la matrice de rotation idéale est obtenu au moyen de la méthode de Kabsch [45] qui permet de minimiser le RMSD de manière très simple. Cette méthode effectue la décomposition en valeurs singulières (SVD) de la matrice de covariance des points des deux ensembles afin d'en extraire la matrice de rotation optimale. La matrice de covariance

$$A = P^T Q \tag{7}$$

4. la transformation de mise à l'échelle n'est pas utilisée.

est obtenue à partir d'un produit matriciel où P et Q sont des matrices $(N \times 3)$ qui représentent les deux ensembles de N points à superposer [5]. Tout comme l'analyse procustéenne, les algorithmes de PyMol superposent les structures en minimisant le RMSD, ce n'est pas le cas de MATT qui utilise d'autres critères. L'appendice B compare, de manière plus détaillée, MATT avec les algorithmes de PyMol. La superposition de structures sera utilisée, dans ce travail, afin de calculer le score AL0 entre deux structures et donc afin d'analyser les performances des diverses méthodes d'optimisation que nous avons implémentées.

4.3 Evaluation des structures prédites et choix des mesures utilisées

L'évaluation de la qualité d'une structure prédite n'est pas chose aisée, principalement parce que celle-ci est subjective étant donné qu'il n'existe pas de critère universel permettant de classer parfaitement les structures comme bonnes ou mauvaises. Lors du CASP, des méthodes numériques sont utilisées pour sélectionner les meilleures prédictions qui seront finalement analysées par des experts humains pour en déterminer la qualité. Les critères automatiques d'évaluation ont continuellement évolué et se sont améliorés au fil du temps de sorte que des procédures standards ont pu voir le jour et qu'elles sont aujourd'hui largement acceptées par la communauté. Dans CASP, la qualité moyenne des structures prédites varie suivant la catégorie à laquelle la protéine appartient, ceci explique les approches, différentes pour chaque catégorie, utilisées pour évaluer les prédictions. Ainsi, l'évaluation des protéines appartenant au template based modeling devra être beaucoup plus précise et donc utilisera des critères d'évaluation plus fins que la catégorie template free. Nous allons ici détailler brièvement les différentes méthodes d'évaluation et expliquer les critères que nous retiendrons pour ce travail. Dans la littérature concernant CASP, la cible fait référence à la structure réelle de la protéine que doivent deviner les groupes participant au concours, tandis qu'un modèle fait référence à une prédiction soumise par un groupe pour une cible donnée.

4.3.1 Principaux critères d'évaluation de structures

Le critère le plus communément utilisé est le RMSD (root mean square deviation), c'est-à-dire la racine carrée de la moyenne des carrés des différences de position entre les atomes du modèle et de la cible. Ce critère est extrêmement simple, très rapide à calculer, facilement compréhensible et permet une bonne comparaison de structures proches. Néanmoins, il est très mal adapté à la comparaison de structures relativement éloignées. Il souffre, de plus, de certains défauts : il pénalise très fort les mauvais alignements, il dépend du nombre d'atomes dans la protéine et donc tend à augmenter avec la taille de celle-ci et finalement, il ne fournit aucune indication quant à l'endroit dans la structure où les plus grosses erreurs apparaissent.

La mesure GDT-TS (global distance test) est une méthode de comparaison de structures basée sur des seuils qui permet de pallier certains inconvénients du RMSD. Tout d'abord, le modèle et la cible sont alignés grâce à l'algorithme LGA [91]. Le GDT-TS est ensuite calculé comme étant la moyenne du nombre maximum de paires de résidus correspondants, un appartenant au modèle, l'autre à la cible, dont les C_α sont séparés par une distance inférieure à un certain seuil, pour des valeurs de seuil de 1, 2, 4 et 8 Å. Cette mesure fut d'abord développée pour CASP4 [92] et est depuis largement utilisée. Etant donné que GDT-TS effectue des calculs moyens, cette mesure favorise les modèles qui ont un bon repliement

5. Dans notre cas, N correspond au nombre de résidus de la protéine.

global et donc dont la chaîne principale est sensiblement correcte. Une version haute résolution existe (GDT-HA) qui est basée sur le même principe mais qui permet d'évaluer des structures beaucoup plus précisément puisque les distances de seuil sont 0.5, 1, 2 et 4 Å. La qualité de l'évaluation par GDT-TS dépend de la manière dont les structures sont alignées. Une petite perturbation locale peut entraîner une déviation assez importante de l'alignement et donc des scores plus faibles, cette mesure n'est donc pas parfaitement adaptée à l'évaluation de structures fortement différentes.

AL0 est une mesure qui se rapproche fortement du GDT-TS puisque celle-ci correspond au pourcentage de résidus dont le C_α est séparé par moins de 3.8 Å du C_α correspondant de la structure cible après avoir aligné celles-ci grâce à l'algorithme LGA.

Les méthodes présentées ci-dessus nécessitent une superposition tridimensionnelle préalable entre le modèle et la cible. Le QScore est un moyen élégant de s'affranchir de l'alignement [7, 35, 24]. Le QScore développé pour CASP8 utilise les matrices de distances calculées pour le modèle et la cible. r_{ij} représente donc la distance séparant le C_α du résidu i du C_α du résidu j dans le modèle. r^0_{ij} représente la même distance dans la structure de la cible. Un QScore est ensuite calculé pour chaque paire de résidus suivant la formule (8).

$$Q_{ij} = \exp\left(-\left(r_{ij} - r^0_{ij}\right)^2\right) \tag{8}$$

Il faut ensuite définir un score global, celui-ci est obtenu en effectuant une moyenne. Plus précisément, les Q_{ij} sont triés par ordre décroissant pour obtenir une séquence décroissante de QScores : $Q_r \ \forall r = 1 \ldots \frac{N(N-1)}{2}$. Une valeur moyenne $< Q_{ij}(M) >$ est ensuite calculée $\forall M = 1 \ldots \frac{N(N-1)}{2}$ par la formule (9).

$$< Q_{ij}(M) >= \frac{1}{M} \sum_{r=1}^{M} Q_r \tag{9}$$

Ceci permet d'établir un graphique comme celui présenté à la figure 14. Comme illustré sur cette figure, une prédiction parfaite aura un $< Q_{ij}(M) >$ égal à 1 au fur et à mesure que M augmente. Plus la prédiction sera mauvaise, plus la courbe s'approchera rapidement vers 0. Le QScore est une mesure très flexible puisque, moyennant certains arrangements, elle permet de se concentrer plutôt sur la structure globale ou locale de la prédiction. Une mesure globale (Q_{long}) est réalisée en ne considérant que les Q_{ij} pour lesquels $|i - j| \geq a$ où a correspond à une distance en nombre de résidus et vaut typiquement 20. Une mesure locale (Q_{short}) est par contre obtenue en évaluant les QScores pour lesquels $|i - j| < a$.

D'autres critères moins intuitifs existent et sont utilisés, c'est le cas, par exemple, de l'algorithme MAMMOTH [68], qui, en plus de fournir une superposition des structures, calcule un score qui permet d'évaluer à quel point ces structures sont proches.

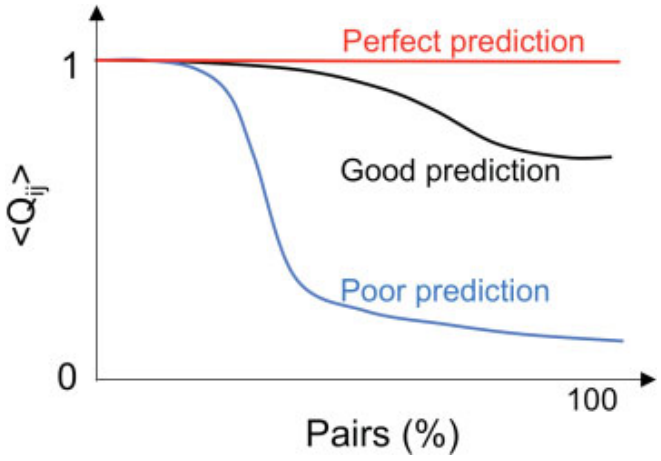


FIGURE 14 – Evolution du QScore en fonction de M. 100% correspond à $M = \frac{N(N-1)}{2}$. [7]

4.3.2 Evaluation des modèles template based

La stratégie d'évaluation des modèles template based se divise en 2 étapes : évaluation des modèles un à un suivant différents critères et ensuite calcul d'un score pour chaque modèle en fonction de la distribution des résultats obtenus sur la totalité des modèles soumis pour la même cible.

La première étape de l'évaluation consiste à calculer pour chaque modèle un certain nombre de valeurs numériques décrivant la similarité du modèle avec la cible. Parmi ces mesures, on trouve, par exemple, les mesures RMSD, GDT-TS, GDT-HA et AL0. Les valeurs ainsi obtenues grâce à ces mesures pourraient être directement utilisées pour classer les modèles. Néanmoins, une telle approche n'est pas égalitaire puisque la difficulté de la prédiction n'est pas la même en fonction des cibles. C'est pourquoi le concept de ZScore [84] qui fut introduit lors du CASP4 est préféré lors de l'évaluation des structures. Ce score tient compte de la totalité des prédictions soumises pour une même cible, rendant ainsi le classement plus juste. En effet, une méthode qui prédit correctement la structure d'une protéine alors que toutes les autres ont échoué est certainement digne d'intérêt même si ses prédictions sont dans la moyenne pour les autres cibles. Le ZScore permet donc d'évaluer la qualité d'une prédiction, non seulement en utilisant des comparaisons entre la structure prédite et la structure cible, mais également en fonction des résultats des autres groupes. Diverses questions pratiques se posent quant à l'utilisation des ZScores et ne sont pas explicitées ici ; plus de détails peuvent être trouvés dans l'article originel [84] ou dans l'article décrivant l'évaluation des modèles template based pour le CASP8 [20].

La procédure complète d'évaluation des modèles est explicitée en détails dans [20]. Nous reprenons ici les points principaux :

1. La mesure GDT-TS est utilisée pour calculer les scores décrivant la similarité entre les modèles proposés et les cibles. Une matrice $n \times m$ est ainsi créée où n est le nombre

de cibles proposées par CASP et m le nombre de modèles soumis pour chaque cible [6]. Chaque élément X_{ij} de cette matrice correspond donc au score GDT-TS évalué entre le modèle j et la cible i. La moyenne $\overline{\mathrm{X}}_i$ et l'écart type σ_i de ces scores sont ensuite calculés pour chaque ligne $i = 1 \dots n$ [7]. Afin de purifier les valeurs obtenues, les *outliers* sont retirés. Ainsi, la moyenne $\overline{\mathrm{X}}_i^{\mathrm{p}}$ et l'écart type σ_i^{p} sont recalculés pour chaque ligne i en négligeant les scores inférieurs à $\overline{\mathrm{X}}_i - 2\sigma_i$.

2. Les ZScores Z_{ij} sont ensuite calculés pour chaque ligne i et chaque modèle j . Le ZScore est calculé par la formule (10).

$$Z_{ij} = \left(\frac{\mathrm{X}_{ij} - \overline{X}_i^{\mathrm{p}}}{0.5\sigma_i^{\mathrm{p}}} \right) f \tag{10}$$

où f représente le pourcentage de la structure qui est présent dans la prédiction. On le voit ici un ZScore positif indique qu'une prédiction est meilleure, en moyenne, que les autres pour une même cible.

3. Les ZScores négatifs sont mis à 0.
4. Les différentes prédictions sont classées suivant les ZScores calculés.

Cette procédure permet d'obtenir un ZScore par modèle soumis (donc par groupe en compétition dans CASP) et par cible. Afin de déterminer un score global par groupe, les ZScores obtenus pour toutes les prédictions d'un même groupe sont sommés ou moyennés. Le classement final des groupes sera établi grâce à cette valeur globale. Notons que des tests statistiques sont effectués afin de déterminer si les différences de scores observées entre les différents groupes sont statistiquement significatives ou pas.

4.3.3 Evaluation des modèles template free

Le GDT-TS est relativement peu approprié pour l'évaluation des méthodes *ab initio*, ceci est dû au fait que peu de prédictions ont un RMSD sous la barre des 10 Å [7]. C'est pourquoi des mesures supplémentaires doivent être utilisées et en particulier le QScore. Le OK_rank a été utilisé lors du CASP8 [7] pour classer les prédictions obtenues. Ce score est une combinaison de QScores globaux et locaux, de GDT-TS et des scores obtenus grâce à MAMMOTH. Les QScores et le GDT-TS sont discrétisés par pas de 1 %. Ceci permet d'obtenir un classement, représenté sous forme de valeurs entières allant de 0 à 99, pour les prédictions en fonction de chacun des scores. Le classement du score de MAMMOTH est obtenu directement, c'est-à-dire que le 10^{e} meilleur score sera classé à la 10^{e} position. Le classement final du modèle est obtenu en effectuant une moyenne de ces quatre scores entiers.

4.3.4 Choix de la méthode d'évaluation de structure

Au vu de la relative complexité des méthodes présentées dans les sections 4.3.1, 4.3.2 et 4.3.3, nous avons décidé de n'utiliser qu'une partie de celles-ci étant donné que nos prédictions ne doivent pas être évaluées avec une grande précision. De plus, celles-ci seront légèrement modifiées afin qu'elles soient encore plus adaptées à nos besoins. Le choix final a également été influencé par la facilité de mise en oeuvre des méthodes présentées dans les proceedings du

6. Dans cette explication, on suppose qu'il y a le même nombre de modèles soumis pour chaque cible.
7. donc sur les scores GDT-TS des modèles soumis pour une même cible.

CASP. Au vu de ces éléments, nous avons donc utilisé les QScores pour évaluer la similarité de deux structures. Néanmoins, une seule valeur était suffisante pour nos besoins, aussi la totalité des $< Q_{ij}(M) >$ n'était pas nécessaire. Nous n'avons donc utilisé que la valeur $< Q_{ij}(M) >$ pour $M = \frac{N(N-1)}{2}$, c'est-à-dire la moyenne de tous les QScores. Cependant, nous avons quand même considéré les mesures globales et locales : Q_{long} et Q_{short}.

5 Description de la méthode *ab initio* proposée

L'approche *learning for search*, comme son nom l'indique, consiste à apprendre, à partir de données observées, le meilleur chemin qu'un algorithme de recherche puisse parcourir dans le but de trouver de manière efficace une solution optimale pour un problème donné. Etant donné la complexité mise en avant précédemment et le nombre élevé de solutions disponibles, cette approche semble tout indiquée pour tenter de résoudre le problème de prédiction de structures de protéines.

Nous proposons ici une approche pour apprendre, à partir d'une base de données D de structures tridimensionnelles connues, une procédure d'optimisation de structures de protéines quelconques définies par leur séquence de résidus p. La base de données D se présente sous la forme d'un ensemble de paires $D = (p_i, s_i^*)_{i=1}^A$, où p désigne une séquence de résidus, s^* une structure tridimensionnelle considérée comme correcte [8] et A le nombre de protéines de la base de données. Outre la base de données, nous exploitons un oracle [9] qui permet de calculer l'énergie d'une structure candidate s pour une protéine p (éventuellement de façon approchée). Nous ferons l'hypothèse, valable pour la grande majorité des protéines, qu'une bonne structure est une structure qui minimise cette énergie.

Nous souhaitons créer une procédure itérative d'optimisation de structures de protéines qui, à chaque itération, sélectionne et applique à la structure un opérateur la modifiant. Après un certain nombre d'itérations, le procédé s'arrête et renvoie la meilleure structure, en termes d'énergie, qu'il a générée. Notons que la qualité d'une telle stratégie dépend fortement de l'ensemble d'actions candidates disponibles à chaque itération et de la manière d'en choisir une. En effet, dans le cas des protéines le nombre d'actions permettant de modifier une structure est très grand et toutes celles-ci ne sont pas égales en termes d'amélioration de structure. Ainsi, plus les opérateurs seront choisis de manière pertinente, plus l'algorithme sera efficace. C'est donc ce point qui fera l'objet de l'apprentissage automatique et qui permettra d'améliorer l'optimisation des structures.

5.1 Description générale de la méthode proposée

Notre procédure se base sur un algorithme d'optimisation $\mathcal{A}$ qui effectue les tâches précédemment citées. C'est ce même algorithme que nous utiliserons tout au long de notre méthode mais qui diffèrera, suivant les étapes, par la manière de générer les opérateurs à chaque itération.

La logique de ce travail se divise en quatre temps. La première étape consiste à générer, pour chaque protéine p_i de la base de données D, un ensemble de B structures intermédiaires susceptibles d'être rencontrées lors d'un processus d'optimisation. Le but ici est d'étoffer notre base de données en ajoutant, pour chaque protéine, diverses structures plus ou moins repliées. Cela est réalisé au moyen de l'algorithme $\mathcal{A}$ qui génèrera des opérateurs de manière aléatoire. Nous disposerons ainsi d'une nouvelle base de données E de structures de la forme

$$E = (p_i, s_{i,j}, s_i^*)_{(i,j)=(1,1)}^{(A,B)}. \tag{11}$$

8. Dans le cadre de notre travail, cette structure correcte sera obtenue à partir de la PDB.
9. L'oracle utilisé ici sera la fonction d'énergie standard de la suite logicielle Rosetta.

L'étape suivante consiste à générer, pour chaque couple (i, j) de E, un ensemble de C opérateurs o, considérés comme optimaux ou quasi-optimaux, qui seront ajoutés au jeu de données E. Cette étape sera réalisée au moyen d'un algorithme de type EDA. Le résultat sera une base de données F qui contiendra un ensemble de protéines, chacune associée à sa structure optimale ainsi que, pour chacun de ces couples, un ensemble de repliements intermédiaires. La base de données offre, de plus, pour chaque structure intermédiaire, un ensemble d'opérateurs permettant de la modifier dans le sens d'une amélioration de la structure. Cette base de données F est de la forme

$$F = (p_i, s_{i,j}, o_{i,j,k}, s_i^*)_{(i,j,k)=(1,1,1)}^{(A,B,C)}. \tag{12}$$

La base de données F, ainsi constituée, est ensuite utilisée afin de créer un modèle génératif $P_\theta[o|s]$ permettant d'échantillonner un opérateur o, conditionnellement à la structure courante s de la protéine décrite par un ensemble de caractéristiques ou *features*. La dernière étape consistera en l'introduction de ce modèle génératif dans l'algorithme d'optimisation $\mathcal{A}$ afin de tester son efficacité et de valider notre approche.

Ainsi, l'algorithme final d'optimisation que nous allons créer ne diffère de l'algorithme simple utilisé pour générer les structures intermédiaires (cf. section 5.2 et 5.4) que par la politique de sélection des opérateurs. Dans la procédure finale, ceux-ci seront sélectionnés conditionnellement à la structure courante s. Le modèle $P_\theta[o|s]$ doit permettre une exploration aussi efficace que possible de l'espace d'état des conformations et doit donc être capable d'identifier les actions les plus intéressantes en vue d'optimiser la structure s.

Dans la suite, nous commençons, dans la section 5.2, par décrire l'algorithme d'optimisation itératif $\mathcal{A}$ que nous utiliserons. Ensuite, nous présentons, dans la section 5.3, les divers opérateurs de modification de structures que nous considérons dans ce travail. Nous décrivons, par après, la manière de générer la base de données E de structures intermédiaires dans la section 5.4. Ensuite, la section 5.5 explique, quant à elle, le procédé permettant de déduire, à partir de E, les opérateurs de modification optimaux. La section 5.6 poursuit en décrivant les détails concernant la définition du modèle génératif ainsi que sa création. Finalement, la procédure finale d'optimisation est présentée dans la section 5.7.

5.2 Description de l'algorithme d'optimisation

L'algorithme d'optimisation $\mathcal{A}$ envisagé dans ce travail est itératif et basé sur une recherche heuristique stochastique de type *recuit simulé* [47]. Cet algorithme n'est pas le plus sophistiqué qui soit mais il permettra de se faire une idée quant à la pertinence de notre approche. Les diverses étapes de l'optimisation sont décrites ci-dessous et résumées dans l'algorithme 1. Dans la suite de ce document, nous utiliserons activement la fonction d'énergie, c'est pourquoi nous lui réservons une notation : $\mathcal{E}$.

1. La procédure dispose en entrée de la séquence de résidus p de la protéine dont on veut prédire la structure tridimensionnelle d'énergie minimale. Il faut aussi fournir à l'algorithme une fonction de décroissance de la température (cf. plus loin).
2. La structure s est d'abord initialisée. Nous avons pris le parti d'initialiser la structure sous forme d'une hélice en fixant les angles de torsion à certaines valeurs bien précises.

L'énergie de cette structure est ensuite évaluée afin de fixer la première valeur de comparaison qu'utilisera l'algorithme : $e_{min} = \mathcal{E}(s)$.

3. A chaque étape i, la procédure choisit un opérateur o parmi l'ensemble $\mathcal{O}$ des opérateurs disponibles.
4. L'opérateur o est ensuite appliqué à la structure courante s pour fournir une structure candidate $s_{candidat} = s(o)$ dont l'énergie est ensuite évaluée par l'oracle.
5. Le critère de Metropolis [64] est utilisé afin d'accepter ou de rejeter la structure candidate. Ce critère se présente sous la forme de l'équation (13).

$$C = \exp\left(\frac{-\Delta E}{kT}\right) \tag{13}$$

où ΔE, k et T représentent respectivement la différence de l'énergie de la structure candidate et de l'énergie courante, la constante de Boltzmann et la température absolue du système. L'acceptation ou le rejet de la structure se fait alors suivant les critères suivants. Si $\Delta E < 0$, alors le système évolue vers une énergie plus favorable (plus faible) et on accepte la structure candidate. Sinon, un nombre aléatoire ξ est tiré d'une distribution uniforme sur $[0, 1]$ et est comparé à la valeur C. Si $\xi < C$, alors le candidat est accepté, sinon il est rejeté. Ce procédé permet de toujours accepter les structures qui évoluent vers un minimum de l'énergie et d'accepter parfois des structures qui évoluent vers des énergies plus grandes, ceci dans le but d'explorer l'espace des conformations et donc d'essayer de ne pas rester piégés dans des minima locaux. Dans l'algorithme du recuit simulé [47], la température T décroît au fil des itérations, ceci dans le but de permettre à des structures relativement mauvaises d'être acceptées en début d'optimisation mais pas dans la partie finale de celle-ci.

6. Si la structure candidate $s_{candidat}$ est acceptée, alors $s = s_{candidat}$ et l'énergie minimale est mise à jour, sinon on recommence la procédure à partir du point 3.
7. Le processus se termine après avoir effectué un certain nombre, fixé *a priori*, d'itérations. La procédure renvoie à ce moment la meilleure structure rencontrée (comme évalué par l'oracle) au cours du processus de recherche.

Notons que la qualité d'une telle stratégie dépend fortement de l'ensemble d'actions candidates disponibles à chaque itération et de la manière d'en choisir une. En effet, ce type de méthodes est susceptible de trouver des minima locaux, malgré l'application du critère de Metropolis. Nous l'utilisons néanmoins car il est simple d'implémentation et de compréhension et qu'il permet une comparaison aisée entre les différentes politiques de sélection des opérateurs que nous allons implémenter. On peut raisonnablement penser que, quelle que soit la complexité du système, si il existe des opérateurs qui peuvent, à chaque itération, modifier de manière importante et fondamentale le système, alors celui-ci est moins susceptible d'être entraîné par l'algorithme vers des minima locaux de la fonction objectif. Hélas, de ce constat, qui privilégie un ensemble d'opérateurs le plus complet possible, en découle un autre : plus le nombre et la complexité des opérateurs disponibles à chaque itération seront grands, plus lente sera la convergence de l'algorithme sans garantie aucune que l'augmentation de la complexité fournira un résultat meilleur que celui qui aurait pu être obtenu avec des opérateurs plus simples. On voit apparaître ici le paradoxe entre l'exhaustivité et l'efficacité de l'exploration de l'espace de recherche. L'exhaustivité est nécessaire si le problème possède un grand nombre

Algorithm 1 Recuit simulé appliqué à l'optimisation de la structure de protéines

Input: s la protéine dont la séquence d'acides aminés p est donnée
Input: N le nombre maximum d'itérations
Input: $M(.,.)$ une fonction implémentant le critère de Metropolis
Input: $\mathcal{O}$ un ensemble d'opérateurs o
Input: $\mathcal{E}(.)$ un oracle fournissant l'énergie de la protéine

```
initialiser les positions des résidus de s
e_min = E(s)
for i = 1...N do
    sélectionner un bon opérateur o ∈ O
    s_candidat = s(o)
    e_candidat = E(s_candidat)
    if M(e_candidat, e_min) then
        s = s_candidat
        e_min = e_candidat
    end if
end for
return la meilleure structure tertiaire observée
```

de minima locaux et que la solution désirée se situe au minimum global de la fonction objectif. L'efficacité de l'exploration caractérise par contre la possibilité de trouver une solution en un temps raisonnable. Ces deux caractéristiques sont évidemment cruciales puisqu'elles influencent respectivement la qualité de la solution trouvée et la probabilité de trouver une solution rapidement. Dans le cadre des problèmes de petite taille, l'exhaustivité et l'efficacité ne sont pas antinomiques. Ce n'est pas le cas pour les problèmes de plus grande taille, c'est pourquoi des aménagements doivent être tentés afin de concilier les deux objectifs.

Dans le cas des protéines, la nature du problème d'optimisation (contraintes et caractéristiques de la fonction d'énergie) est telle qu'une grande complexité des opérateurs est désirée afin d'obtenir l'exploration la plus exhaustive possible d'un espace d'état qui possède un nombre colossal de minima locaux. Il importe également que l'exploration de cet espace s'effectue de la manière la plus efficace qui soit. Nous réaliserons ceci en modélisant le choix des opérateurs par une distribution stochastique. Les bons opérateurs auront, dans cette distribution, une probabilité plus grande augmentant ainsi leurs chances d'être sélectionnés et donc augmentant ainsi la probabilité que l'effet produit sur la structure soit positif. Réduire ainsi la taille de l'ensemble des opérateurs est un aspect non trivial. Notre stratégie scientifique a pour objectif de mettre en oeuvre l'apprentissage automatique afin de définir une bonne politique de choix parmi un nombre relativement élevé d'actions candidates dans le cadre du recuit simulé [10].

5.3 Opérateurs de modification de structures

Cette section décrit les opérateurs que nous utiliserons dans ce travail afin de modifier les structures des protéines étudiées. Comme mentionné précédemment, chaque protéine a un grand nombre de degrés de liberté. En effet, chaque atome peut être déplacé indépendamment

10. Cette approche est bien évidemment généralisable à d'autres méthodes que le recuit simulé

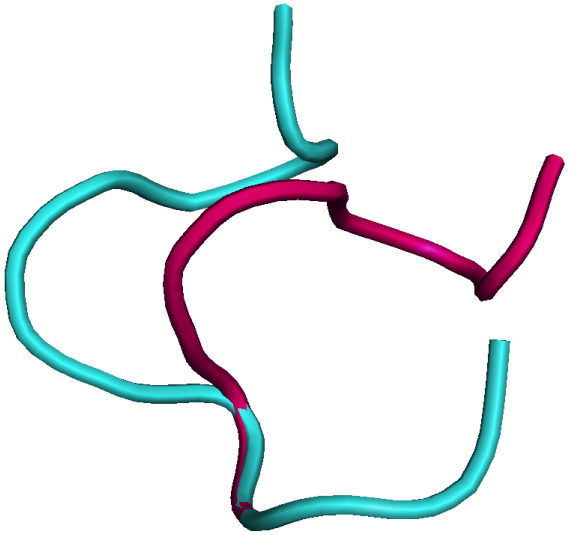


FIGURE 15 – Illustration d'un PhiPsiMover sur une structure jouet. Image générée grâce à PyMol.

des autres dans le but de générer une nouvelle structure. Néanmoins, cette approche est quelque peu irréaliste puisque généralement la structure d'un acide aminé est plus ou moins rigide. Nous allons donc plutôt utiliser un ensemble d'opérateurs de plus haut niveau afin de modifier la structure de la protéine. Implémenter ces opérateurs de manière efficace est relativement complexe, c'est pourquoi nous nous contenterons de ceux qui sont disponibles dans la suite logicielle Rosetta [55] et portent le nom de *movers*. Rosetta offre une grande bibliothèque de tels opérateurs, néanmoins, dans le cadre d'une première approche, seul un nombre restreint de movers nous intéresse. Nous avons choisi de nous concentrer essentiellement sur trois movers différents : le PhiPsiMover, le ShearMover et le RigidBodyMover.

Le PhiPsiMover est un mover qui nécessite trois paramètres : R l'indice du résidu dans la séquence d'acides aminés, $\Delta\phi$ la modification de l'angle de torsion ϕ de ce résidu et $\Delta\psi$ la modification de l'angle ψ de ce même résidu. Pour rappel, les angles ϕ et ψ sont illustrés sur la figure 2 page 4. L'effet de ce mover est de déplacer, par une rotation, l'entièreté de la structure se trouvant de part et d'autre du résidu [11]. Les modifications résultant de l'application d'un PhiPsiMover peuvent être non négligeables pour les atomes relativement éloignés du résidu R. En conséquence, un tel mover est très utile pour effectuer des modifications importantes de la structure en particulier au début de l'optimisation. L'effet de l'application d'un PhiPsiMover est illustré à la figure 15.

A l'instar du PhiPsiMover, le ShearMover nécessite trois paramètres qui sont, de plus, sensiblement équivalents à ceux du PhiPsiMover : R l'indice du résidu dans la séquence d'acides aminés, $\Delta\phi$ la modification de l'angle de torsion ϕ de ce résidu et $\Delta\psi$ la modification de l'angle ψ du résidu précédent R dans la chaîne. Le mouvement résultant est un mouvement de cisaillement qui minimise les modifications de structure éloignées du résidu R. Ce type de

11. Remarquons que si $\Delta\phi$ ou $\Delta\psi$ est nul, seule la structure située d'un côté du résidu sera modifiée. Si les deux incréments sont nuls, il n'y aura alors aucune modification.

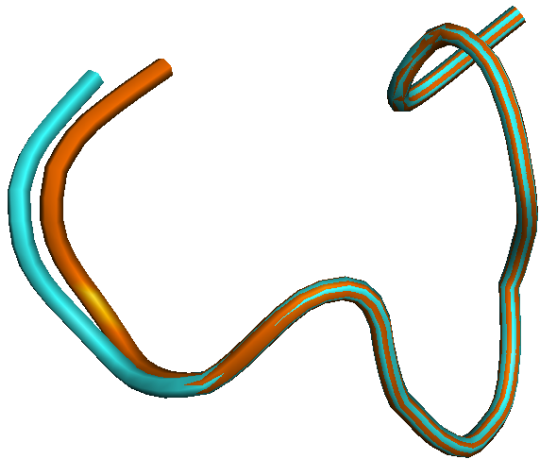


FIGURE 16 – Illustration d'un ShearMover sur une structure jouet. Image générée grâce à PyMol.

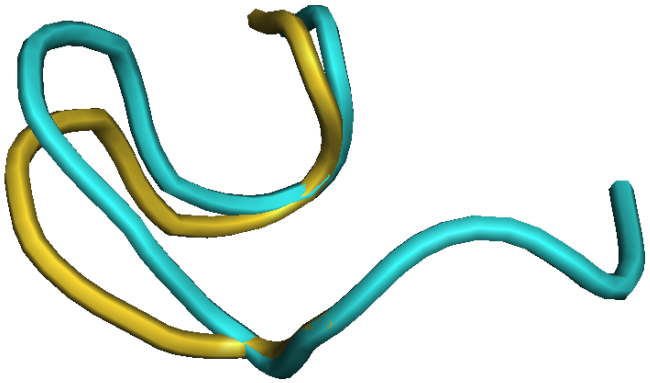
mover est donc un mover dont l'effet est plus local, en comparaison avec le PhiPsiMover, et sera utile en fin d'optimisation afin d'ajuster finement les positions des résidus. Nous avons pu remarquer, au cours de nos expériences, qu'il est également très intéressant en début d'optimisation. La figure 16 illustre l'application d'un ShearMover sur une structure jouet.

Les deux premiers movers présentés se focalisent sur un résidu et modifient ses angles de torsion. Il en résulte généralement des modifications des résidus se situant loin au-delà de la position où a réellement été effectuée la modification. Le RigidBodyMover agit, quant à lui, sur deux ensembles de résidus qu'il va déplacer l'un par rapport à l'autre. Le positionnement relatif des deux ensembles de résidus sera affecté par une translation et/ou par une rotation. Afin d'être appliqué, ce mover nécessite quatre paramètres. R_1 correspond à l'indice, dans la séquence d'acides aminés, du premier résidu participant à la modification de structure tandis que R_2 correspond à l'indice du second résidu. Ces deux résidus définissent les deux ensembles de résidus dont la position sera affectée par le mover. Le premier ensemble contient les résidus numérotés de 1 à R_1, alors que le deuxième ensemble sera constitué des résidus indicés de R_2 à n où n représente la longueur de la protéine. Les deux paramètres supplémentaires, nécessaires au RigidBodyMover, sont m la longueur de la translation effectuée et a l'amplitude de la rotation appliquée. Ces deux transformations sont définies par m et a respectivement ainsi que par l'axe formé par les deux C_α des résidus R_1 et R_2. L'application d'un RigidBodyMover s'effectue en trois temps. D'abord, l'axe entre les C_α est créé :

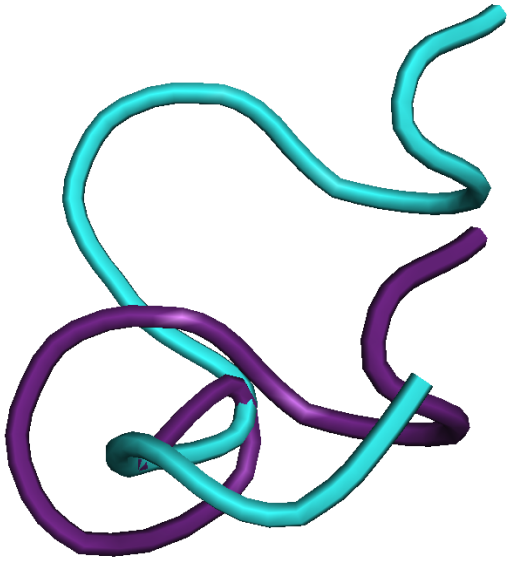
$$t = \begin{pmatrix} x_{C_{\alpha 1}} - x_{C_{\alpha 2}} \\ y_{C_{\alpha 1}} - y_{C_{\alpha 2}} \\ z_{C_{\alpha 1}} - z_{C_{\alpha 2}} \end{pmatrix}. \tag{14}$$

Ensuite, le résidu R_1 étant fixé, une translation de longueur m et d'axe t, ainsi qu'une rotation, d'amplitude a et de même axe, sont appliquées au deuxième ensemble de résidus. Les structures du premier et du deuxième ensemble restent fixées, seule leur position relative est modifiée. Cependant, bien que les structures des deux ensembles restent inchangées, il n'en

est pas de même de la partie de la protéine se situant entre ces deux ensembles. La troisième étape consiste en la reconstruction automatique par Rosetta de cette troisième partie de la protéine. La reconstruction se base sur différents critères afin de sélectionner les nouvelles positions intermédiaires de ces résidus dont un critère de minimisation d'énergie. Ce mover permet de modifier la structure de manière importante aussi bien que de manière très précise. Néanmoins, il possède plus de paramètres que les autres movers et donc sélectionner un RigidBodyMover performant sera plus difficile. Il possède, en outre, le grand avantage de pouvoir modifier les positions relatives de deux parties de la structure sans changer les conformations internes de celles-ci. La figure 17 illustre l'application d'un tel mover.



(a) RigidBodyMover qui effectue uniquement une rotation



(b) RigidBodyMover qui effectue uniquement une translation

FIGURE 17 – Illustration de deux RigidBodyMovers sur une structure jouet. Image générée grâce à PyMol.

5.4 Génération des structures intermédiaires

Un ensemble d'apprentissage E, tel que mentionné précédemment, n'est pas disponible *a priori*, c'est pourquoi nous devons en constituer un. Pour cela, nous proposons d'utiliser l'algorithme d'optimisation itératif, présenté à la section 5.2, avec des opérateurs sélectionnés aléatoirement et un très grand nombre d'itérations, afin de simuler le repliement de la protéine p_i. Des structures intermédiaires $s_{i,j}$ rencontrées pendant le repliement seront ainsi récoltées au cours de l'optimisation à intervalles réguliers. Nous n'avons utilisé ici qu'un seul type d'opérateur tout au long de l'optimisation. Les degrés de liberté se situent au niveau du choix des paramètres du mover. Ainsi, les résidus ont été, à chaque itération, sélectionnés suivant une distribution uniforme, tandis que les paramètres continus ont été sélectionnés via des distributions normales. Les résultats issus des optimisations réalisées dans le cadre de la génération de la base de données F sont décrits à la section 7.1.

5.5 Génération des opérateurs optimaux

Nous sommes, à présent, en possession d'une base de données E contenant des protéines, leur structure optimale et un certain nombre de structures intermédiaires. La prochaine étape de notre procédure consiste à sélectionner, pour chaque structure intermédiaire $s_{i,j}$, un ensemble d'opérateurs $o_{i,j,k}$ qui, appliqués à la structure $s_{i,j}$ fournissent une nouvelle structure plus proche de la structure optimale.

Dans le but de résoudre ce problème, il nous faut une fonction F_{proxy} permettant de décrire l'amélioration apportée par un opérateur o. Cette fonction est décrite à la section 5.5.1 de ce document. Cette fonction permet alors d'écrire mathématiquement le problème qui nous occupe :

$$\left\{ o_{i,j,k} : o_{i,j,k} \approx \arg \max_{o \in \mathcal{O}(s_{i,j})} F_{\text{proxy}}(s_{i,j}, o, s_i^*) \right\}, \forall s_{i,j} \in E. \tag{15}$$

Afin de générer des actions optimales en vue de constituer la base de données F, nous proposons d'appliquer à chaque entrée de la base de données E un algorithme évolutionnaire visant à déterminer une ou plusieurs actions $o_{i,j,k} \in \mathcal{O}(s_{i,j})$ optimales du point de vue de F_{proxy}.

5.5.1 Définition d'une fonction décrivant l'amélioration de structures

Afin de sélectionner, les opérateurs $o_{i,j,k}$ optimaux pour une structure $s_{i,j}$, il nous faut un critère permettant d'évaluer la différence de proximité entre la structure cible s_i^* d'une part et la structure $s_{i,j}(o)$ après application de l'opérateur d'autre part. C'est-à-dire qu'il nous faut un moyen permettant de dire si un opérateur o, sélectionné d'une quelconque manière, permet, en l'appliquant à la structure $s_{i,j}$, d'obtenir une structure qui est plus proche de la structure cible s_i^* par rapport à la structure précédente $s_{i,j}$. Ce critère doit permettre, en plus, de classer les opérateurs sélectionnés en fonction des améliorations qu'ils apportent. Pour ce faire, nous utiliserons une somme pondérée de deux termes, un dépendant de l'énergie et l'autre dépendant de la structure. Cette fonction est reprise ci-dessous.

$$F_{\text{proxy}} = \mu T_E + (1 - \mu) T_S \tag{16}$$

Le premier terme de la fonction proxy est un terme d'énergie. Dans ce problème, c'est la variation de l'énergie qui nous intéresse, non seulement le signe, mais également l'amplitude de la différence. Néanmoins, étant donné que le critère que nous utiliserons est une somme pondérée de deux termes, il importe que ceux-ci aient des valeurs bornées (typiquement entre 0 et 1). C'est pourquoi nous ne pouvons utiliser directement ΔE qui est la variation d'énergie entre les deux structures. Nous utilisons plutôt la forme suivante

$$T_E = \frac{2}{1 + \exp\left\{-0.0005\left(\mathcal{E}(s_i^*) - \mathcal{E}(s_{i,j}(o))\right)\right\}} \tag{17}$$

où, pour rappel, la fonction d'énergie est représentée par $\mathcal{E}$. L'équation (17) a la forme d'une sigmoïde pour limiter les valeurs entre 0 et 1. Le facteur 2 au numérateur permet d'obtenir cette gamme de variation puisque l'énergie de la référence $\mathcal{E}(s_i^*)$ est supposée être l'énergie minimale atteignable, le terme $\Delta E = \mathcal{E}(s_i^*) - \mathcal{E}(s_{i,j}(o))$ ne sera donc jamais positif et donc une sigmoïde normale (facteur 1 au numérateur) aurait des valeurs comprises entre 0 et 0.5. Le facteur 2 permet d'étendre la gamme jusque 1. Il importe que la fonction (17) soit suffisamment sensible pour une large gamme d'énergies. Au vu de notre expérience, les énergies fournies par Rosetta pour des protéines relativement petites peuvent s'élever jusqu'à quelques dizaines de milliers de $kCal/mol$ et descendre jusque dans les valeurs négatives. La valeur du paramètre multiplicatif au sein de l'exponentielle a été choisie dans le but d'atteindre une sensibilité correcte dans des énergies s'étendant sur une gamme de quelques dizaines de milliers d'unités de part et d'autre de 0.

Le deuxième terme de l'équation (16) est un terme qui évalue la similarité entre deux structures. Nous utiliserons le Q_{long} afin d'évaluer cette ressemblance. Le QScore est, par construction, borné entre 0 et 1, aucun aménagement particulier n'est donc nécessaire en vue de mettre celui-ci à l'échelle. Le Q_{long} est donc utilisé directement dans l'équation (16).

$$T_S = Q_{\text{long}} \tag{18}$$

Les deux fonctions (17) et (18) évoluent toutes deux de la même manière. Plus la structure candidate se rapproche de la structure cible, plus leurs valeurs se rapprochent de 1. La somme pondérée, par μ, de ces deux termes, fournit l'équation (16) qui permet d'assigner, à l'opérateur o candidat, un score compris entre 0 et 1 d'autant plus élevé que la structure obtenue est proche de la structure cible. Cette fonction F_{proxy} permet donc d'évaluer la structure et d'effectuer un classement entre les différents opérateurs candidats.

5.5.2 Algorithme de génération des opérateurs optimaux

La construction de la base d'apprentissage F suppose que nous sommes en mesure de déterminer, pour une structure courante s rencontrée au cours de l'optimisation, une ou plusieurs actions $o \in \mathcal{O}(s)$ qui représentent de bonnes approximations de

$$\arg\max_{o \in \mathcal{O}(s)} F_{\text{proxy}}(s, o, s^*). \tag{19}$$

L'algorithme proposé à ce niveau est un algorithme de type *Estimation of Distribution Algorithms* ou EDA [50]. Soient une structure s donnée, la structure cible correspondante s^* et un ensemble $\mathcal{P} = \{p_\gamma : \gamma \in \Gamma\}$ [12] de lois de probabilités p_γ de tirage de o dans $\mathcal{O}(s)$ où γ désigne les paramètres de p et Γ l'ensemble des paramètres possibles γ, l'algorithme EDA fonctionne de la manière suivante :

1. Le processus commence avec une valeur de γ pour laquelle $p_\gamma(\cdot)$ couvre bien $\mathcal{O}(s)$, c'est-à-dire qu'il n'existe pas de valeur $o \in \mathcal{O}(s)$ telle que $p_\gamma(o)$ est très faible, voire nulle.
2. Un ensemble $O(s)$ d'opérateurs $o \in \mathcal{O}(s)$ de cardinalité fixée à N est créé par N tirages aléatoires, successifs et indépendants selon la loi de probabilités $p_\gamma(\cdot)$.
3. On évalue la valeur de $F_{\text{proxy}}(s, s(o_j), s^*)$, $\forall o_j \in O(s)$, et on ajoute à une liste O^x les opérateurs o_j pour lesquels $F_{\text{proxy}}(s, s(o_j), s^*) > F_{\text{proxy}}(s, \mathbb{I}, s^*)$ où $\mathbb{I}$ représente l'opérateur identité, c'est-à-dire qui ne modifie pas la structure. Les x meilleurs éléments de O^x sont gardés tandis que les autres sont supprimés de la liste. Notons que la liste O^x n'est pas vide et qu'elle se complète au fil des itérations, cela veut dire que la liste O^x garde en mémoire les x meilleurs opérateurs jamais observés.
4. La valeur du paramètre $\gamma_+ \in \Gamma$ qui maximise la vraisemblance de O^x est ensuite estimée (cf. section 5.5.3). Le procédé recommence alors à partir de l'étape 2 en posant $\gamma = \gamma_+$.
5. Le processus se termine soit après un nombre donné, fixé *a priori*, d'itérations, soit lorsqu'il y a convergence des paramètres γ, c'est-à-dire que $\gamma_+ \approx \gamma$.
6. La ou les meilleures valeurs de o (au sens de F_{proxy}) rencontrées lors de la procédure sont finalement renvoyées comme approximations de l'optimum.

L'algorithme 2 résume les étapes principales de cette méthode. La procédure décrite ici est une variante des algorithmes EDA traditionnels. Ces modifications ont été réalisées afin d'obtenir les meilleurs résultats sous des contraintes de temps de calcul. Les algorithmes classiques ont été envisagés, ainsi que d'autres variantes, mais ceux-ci généraient des solutions de moindre qualité dans notre cadre de travail (à savoir des expériences de durée volontairement limitée). Parmi les variantes de l'algorithme que nous avons considérées, nous pouvons citer :

- Apprentissage du point 4 réalisé sur les x meilleurs opérateurs obtenus à l'itération courante plutôt que sur les x meilleurs opérateurs jamais observés.
- Combinaison des deux approches précédentes : garder dans O^x les $\frac{x}{2}$ meilleurs opérateurs générés à l'itération courante ainsi que les $\frac{x}{2}$ meilleurs opérateurs jamais observés.
- Variation des valeurs de sorties : soit l'algorithme renvoie les y meilleurs opérateurs jamais observés, soit les y meilleurs opérateurs observés à la dernière itération.

En plus de ces variantes dans le fond de la méthode, diverses distributions p_γ ainsi que divers algorithmes d'apprentissage ont été testés pour l'apprentissage des meilleurs o pour une structure s donnée. Les distributions choisies sont explicitées dans la section 5.5.3. Les résultats de ces expérimentations sont détaillés dans la section 7. L'algorithme 2 schématise la procédure utilisée afin de générer les opérateurs optimaux pour une structure courante $s_{i,j}$.

12. $\mathcal{P}$ dépend de s notamment au travers de $\mathcal{O}(s)$ mais nous laissons tomber cette dépendance qui n'est pas fondamentale à ce niveau

Algorithm 2 EDA pour la détermination des actions optimales pour une structure

Input: s la protéine dont la séquence d'acides aminés p est donnée
Input: $\mathcal{O}$ un ensemble d'opérateurs o
Input: $F_{\text{proxy}}(.)$ une fonction quantifiant l'optimalité d'un opérateur
Input: $\mathcal{P} = \{p_\gamma : \gamma \in \Gamma\}$ un ensemble de distributions permettant de générer des opérateurs $o \in \mathcal{O}$, γ les paramètres de cette distribution
Input: N le nombre d'opérateurs tirés aléatoirement de p_γ
Input: M le nombre maximum d'itérations
Input: x le nombre de meilleurs opérateurs utilisés pour l'apprentissage

```
initialiser la valeur de γ pour que p_γ couvre bien O
initialiser la liste vide O^x = {}
for i = 1...M do
    for i = 1...N do
        échantillonner o ∈ O à partir de p_γ
        if F_proxy(s, s(o), s*) > F_proxy(s, 𝕀, s*) then
            ajouter l'opérateur o à la liste O^x
        end if
    end for
    purger O^x afin de ne garder que les x meilleurs opérateurs
    apprendre les paramètres γ+ qui maximisent la vraisemblance de O^x
    γ=γ+
end for
return le meilleur opérateur o (au sens de F_proxy)
```

5.5.3 Description des distributions utilisées

Dans le cadre de la génération des opérateurs optimaux, l'algorithme EDA que nous utilisons nécessite des distributions permettant d'échantillonner des opérateurs. Celles-ci doivent disposer d'un algorithme d'apprentissage relativement simple et efficace puisque les distributions doivent être mises à jour à chaque itération. Notons que, généralement, les algorithmes EDA sont utilisés soit sur des distributions discrètes, soit sur des distributions continues. Nous devons l'appliquer ici à des éléments structurés (les movers) ce qui complique de manière non-négligeable les distributions. La densité de probabilités de sélection d'un mover est ainsi donnée par

$$p_\gamma[o] = p_{\gamma^p}[m] \prod_{\forall i:\lambda_i \in \Lambda} p_{\gamma_i^s}[\lambda_i|m] \tag{20}$$

où Λ représente l'ensemble des paramètres d'un mover. Le premier facteur représente la distribution permettant de sélectionner le type de mover, tandis que les autres facteurs modélisent les distributions des paramètres des movers. Nous modélisons donc la probabilité de sélectionner un mover comme le produit de la probabilité de sélection du type de mover par la probabilité de sélection des paramètres. Les paramètres sont ici considérés comme étant indépendants les uns des autres mais dépendants du type de mover bien qu'une telle dépendance se justifie intuitivement [13]. Nous présentons d'abord les distributions qui ont été

13. Par exemple, il semble logique de sélectionner la valeur d'un angle de torsion en fonction du résidu qui doit être modifié.

considérées. Ensuite, nous décrirons comment celles-ci ont été utilisées afin de générer les movers et leurs paramètres.

Distributions discrètes

1. Le premier modèle probabiliste de distribution discrète est le plus simple qui soit. Il consiste simplement en un vecteur de n probabilités, où chaque élément correspond à la probabilité de sélectionner une valeur parmi les n que peut prendre la variable aléatoire discrète. Ce type de modèle est très simple et permet d'échantillonner très facilement la distribution. Son apprentissage est, de même, aisé puisqu'il suffit de calculer les fréquences empiriques d'apparition des différentes valeurs de la variable aléatoire dans la base d'apprentissage et d'attribuer ces fréquences au modèle probabiliste $p_\gamma[o]$. Attention cependant, ce type de distributions est susceptible de converger très, voire trop, rapidement sur des ensembles d'apprentissage trop petits et exclut complètement une valeur de la variable aléatoire si elle n'est pas observée dans l'ensemble d'apprentissage.
2. La seconde méthode dérive de la première mais a été modifiée afin de tenir compte de ses limitations. Elle consiste également en un vecteur de probabilités auquel deux méta-paramètres ont été ajoutés : un taux d'apprentissage τ ou learning rate et une probabilité minimale pour chaque élément $p_{\min}$. Le paramètre τ est utilisé afin de ralentir la convergence de l'apprentissage du modèle. Le second paramètre est, quant à lui, utile en vue de garder une certaine diversité au niveau de la population des valeurs échantillonnées. Le paramètre $p_{\min}$ est donc la probabilité minimale d'échantillonnage d'une valeur de la variable discrète.

Distributions continues

1. Le premier type de distribution continue utilisée est une distribution gaussienne. Cette densité de probabilités a la forme suivante

$$\mathcal{G}_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-0.5\left(\frac{x-\mu}{\sigma}\right)^2\right\} \tag{21}$$

   où μ et σ représentent respectivement la moyenne et l'écart-type de la gaussienne. L'apprentissage d'une telle distribution est direct puisque les valeurs optimales de μ et σ qui maximisent la vraisemblance des données d'apprentissage correspondent simplement à la moyenne et à l'écart-type calculés sur ces mêmes données. Le problème principal qui affecte la distribution gaussienne est la perte de résolution que son apprentissage peut entraîner sur son domaine de définition. Ainsi, la gaussienne effectue une sorte de moyenne sur la totalité des valeurs de l'ensemble d'apprentissage et génère une courbe qui peut être très lisse si les valeurs d'apprentissage sont très différentes. Par exemple, si des valeurs positives et négatives sont présentes en nombre plus ou moins égal dans l'ensemble d'apprentissage, alors la moyenne sera proche de 0 et l'écart-type grand. La forme de la distribution réelle sera donc perdue. Evidemment, il serait préférable d'avoir deux moyennes se situant une dans les valeurs positives et l'autre dans les valeurs négatives et des écart-types plus petits. Une telle distribution représenterait mieux l'allure de l'ensemble d'apprentissage.
2. Pour pallier l'inconvénient des gaussiennes, nous avons décidé de considérer, non pas une seule gaussienne, mais une mixture de gaussiennes permettant ainsi d'apprendre

des distributions plus générales. La mixture de gaussienne se présente sous la forme suivante

$$GMM(x) = \sum_{i=1}^{N} p_i \mathcal{G}_{\mu_i,\sigma_i}(x) \tag{22}$$

qui correspond à une somme de N gaussiennes (21) où μ_i et σ_i sont les paramètres de la i^{e} gaussienne et p_i le poids de celle-ci dans le modèle. Une telle fonction permettra d'apprendre N gaussiennes qui peuvent avoir des moyennes et des écart-types différents augmentant de ce fait le pouvoir génératif de la distribution. La difficulté de cette approche réside essentiellement dans l'apprentissage qui est réalisé au moyen de l'algorithme *expectation-maximization* ou EM [10, 37]. Cet algorithme maximise la log-vraisemblance des données d'apprentissage, il est itératif, fonctionne en deux phases et converge toujours vers un minimum qui peut cependant être local. La première phase, appelée expectation-step, ou E-step, consiste à calculer les probabilités des données d'apprentissage conditionnellement à chaque modèle de la mixture et à les normaliser par la suite. La seconde étape, appelée maximization-step, ou M-step, consiste à mettre à jour les paramètres des modèles en fonction des probabilités calculées lors de la E-step. L'exécution d'une E-step suivie d'une M-step constitue une itération de l'algorithme. A la fin de chaque itération, la log-vraisemblance est calculée et l'algorithme s'arrête lorsque celle-ci a convergé ou lorsque l'algorithme a itéré un certain nombre de fois.

3. Afin de modéliser l'échantillonnage simultané de deux variables aléatoires, nous avons utilisé des gaussiennes multidimensionnelles. La densité de probabilités de telles distributions est donnée par l'équation (23). Dans cette expression, μ représente le vecteur des moyennes des différentes variables, Σ représente la matrice de covariance, $|\Sigma|$ le déterminant de cette matrice et N la dimension de l'espace.

$$\mathcal{G}_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left\{-0.5\,(x-\mu)^T \Sigma^{-1} (x-\mu)\right\} \tag{23}$$

De même que pour les gaussiennes univariées, l'apprentissage est très simple et consiste simplement à calculer le vecteur des moyennes et la matrice de covariance sur l'ensemble d'apprentissage. Les gaussiennes multidimensionnelles souffrent des mêmes défauts que les gaussiennes univariées, c'est pourquoi nous avons également considéré des mixtures de ces gaussiennes en plus de la version simple. Tout comme pour les gaussiennes à une variable, l'algorithme EM est utilisé afin de réaliser l'apprentissage de telles distributions.

Distributions des types de movers La première étape dans l'échantillonnage d'un opérateur consiste à déterminer le type de mover. A cette fin, nous avons utilisé les deux distributions discrètes que nous avons présentées : le vecteur de probabilités et le vecteur de probabilités avec taux d'apprentissage. L'influence du choix de distribution sera discuté dans la section 7.

Distributions des paramètres de movers L'étape suivante de l'échantillonnage consiste à déterminer les paramètres du mover. Afin de tenir compte du type de mover choisi, chaque mover possède des distributions différentes qui lui sont propres garantissant ainsi l'indépendance entre les différents types de movers. Les paramètres qui restent à échantillonner sont

les indices des résidus et les paramètres continus. Les distributions utilisées dans les deux cas sont décrites ci-dessous.

1. **Distributions des indices de résidus** L'échantillonnage des résidus est un procédé discret. Les distributions vecteur de probabilités (avec et sans taux d'apprentissage) utilisées pour les movers sont donc ici aussi parfaitement adaptées. Néanmoins, celles-ci ont le désavantage de rendre les résidus indépendants les uns des autres alors que, intuitivement, il ne semble pas impossible que, si la sélection d'un résidu permet d'obtenir une bonne amélioration de structure, la sélection d'un résidu proche ait une influence similaire. Ceci nous a poussé à considérer une distribution quelque peu différente des premières évoquées. Cette deuxième approche consiste à modéliser la densité de probabilités de choix d'un résidu par une fonction continue et non plus discrète, la discrétisation, nécessaire pour déterminer l'indice du résidu, n'est réalisée que dans un second temps, après avoir échantillonné un nombre continu de la distribution choisie. Ce type de fonction a l'avantage de rendre moins abruptes les transitions entre résidus et donc de tenir compte de l'hypothèse formulée. Ceci a été réalisé au moyen d'une simple gaussienne et d'une mixture de gaussiennes. Un des problèmes de la distribution gaussienne est que nous perdons totalement l'indépendance des résidus. Ainsi, alors que les distributions discrètes permettaient de mettre en évidence plusieurs résidus, ou groupe de résidus, intéressants, la gaussienne a une résolution trop faible pour permettre une telle mise en évidence. Pour pallier cet inconvénient, les mixtures de gaussiennes ont été utilisées permettant ainsi d'avoir une certaine dépendance entre les résidus proches tout en conservant une relative indépendance des résidus éloignés. Le meilleur des deux méthodes est ainsi regroupé en une seule. Les distributions décrites précédemment sont valables lorsque l'on désire échantillonner indépendamment des résidus, donc lors de la génération des paramètres des PhiPsiMover ou des ShearMover. Concernant les RigidBodyMovers, il serait intéressant de conserver une certaine dépendance entre les deux résidus issus de la distribution. Afin de modéliser ce comportement, nous avons utilisé des gaussiennes bidimensionnelles. Dans le but de conserver le même compromis de dépendance que pour les résidus simples, une mixture de gaussiennes est également utilisée afin de modéliser la densité de probabilités des paires de résidus. Les distributions continues permettent de tenir compte d'une dépendance intéressante entre résidus proches. Cependant, ces distributions nécessitent des ajustements après tirage de la valeur. En effet, une première étape de discrétisation est d'abord nécessaire afin de créer une valeur qui puisse être utilisée comme indice. Ensuite, il faut s'assurer que cet indice soit compris dans le bon intervalle, à savoir $[1; n]$ où n est le nombre de résidus de la protéine.
2. **Distributions des paramètres continus** La première distribution considérée est, tout naturellement, une distribution gaussienne telle que celle de l'équation (21). L'apprentissage étant réalisé de façon identique sur les données observées. De même, les mixtures de gaussiennes ont été utilisées afin d'augmenter la résolution de la distribution.

Dans le cadre de la génération de F, la complexité des distributions a volontairement été maintenue à un seuil relativement faible. Le postulat est que, plus la distribution est complexe, plus sa capacité à générer un large nombre de bons movers est grand, mais, a contrario, son apprentissage s'avère plus difficile. Dans le problème de la génération de la base de données F, la distribution apprise ne doit pas modéliser avec précision la distribution des bons movers,

son seul rôle consiste à converger suffisamment afin de générer un faible nombre (quelques uns tout au plus) de très bons movers. C'est pourquoi des distributions plus simples que dans le cadre de l'apprentissage de F ont été utilisées (cf. section 5.6.3).

5.6 Apprentissage d'un modèle génératif

La particularité du problème de prédiction de la structure de protéines est que le nombre de paramètres sur lesquels on peut agir pour espérer optimiser globalement la structure est très grand. En effet, le nombre d'actions exploratoires à envisager croît au moins de façon quadratique avec le nombre de résidus et l'évaluation de la fonction d'énergie est elle aussi au moins de complexité quadratique en fonction du nombre de résidus, ce qui rend pratiquement impossible toute approche d'exploration exhaustive pour des protéines intéressantes (quelques centaines de résidus au moins). C'est pourquoi un choix intelligent de l'opérateur utilisé à chaque itération de la procédure d'optimisation permet d'améliorer l'efficacité de celle-ci.

5.6.1 Description générale du modèle génératif

Dans notre travail, nous voulons construire par apprentissage automatique un modèle génératif qui fournit, à chaque itération de la procédure d'optimisation de la section 5.2, un opérateur dont la probabilité de sélection est une fonction croissante de l'amélioration de la structure courante qu'il induit. Les divers aspects du choix de l'opérateur sont explicités ci-dessous.

1. Désignons par $\mathcal{O}(s)$ l'ensemble de toutes les modifications candidates de la structure s d'une protéine p, défini à partir d'un ensemble $\mathcal{M}$ de *movers*, c'est-à-dire de classes d'actions pouvant être considérées de même type [14] (voir section 5.3).
2. Ainsi, de façon générique, un opérateur de modification $o \in \mathcal{O}(s)$ est déterminé par un triplet (m, R, λ), où $m \in \mathcal{M}$ désigne un des movers, $R \in \mathcal{R}(m, s)$ désigne un sous-ensemble des résidus de s affectés par le mover (en pratique, un singleton ou un couple), et $\lambda \in \Lambda(m, R)$, un vecteur de paramètres associé à m et R (e.g. les angles de torsion (ϕ, ψ) d'un résidu).
3. La politique d'exploration doit alors générer efficacement, à partir de la description d'une structure s d'une protéine p, un opérateur o de $\mathcal{O}(s)$ qui, avec une probabilité élevée, donnera, après application sur s, une structure $s' = s(o)$ intéressante pour la suite du processus d'optimisation ($s(o)$ désigne ici le résultat de l'application de l'opérateur o à la structure s).
4. Nous proposons de modéliser l'intérêt des actions o dans le contexte d'une structure candidate s par un modèle probabiliste $P_\theta[o|s]$ où nous interprétons $P_\theta[o|s]$ comme la probabilité que o est la meilleure action (dans $\mathcal{O}(s)$) à appliquer à s pour l'optimisation de la structure. Ici, θ représente les paramètres du modèle probabiliste issus d'un ensemble Θ de paramètres possibles. Le modèle $P_\theta[o|s]$ est représenté par l'équation (24). Dans un premier temps et dans un souci de simplicité, nous avons considéré que la distribution des probabilités des résidus $R \in \mathcal{R}(m, s)$ était uniforme, celle-ci peut donc être omise de l'équation (24).

$$P_\theta[o|s] = P_{\theta^p}[m|s] \prod_{\forall i:\lambda_i \in \Lambda} P_{\theta^s_i}[\lambda_i|m, s]. \tag{24}$$

14. Nous utiliserons ceux de Rosetta qui ont l'avantage de déjà exister et d'être efficacement implémentés.

L'équation (24) est décomposée en facteurs. Le premier correspond au choix du type m de mover que l'on désire appliquer or, tout élément $o \in \mathcal{O}$ possède en plus un certain nombre de paramètres continus λ dont il faut également tenir compte pour l'évaluation de la qualité d'un opérateur. Les facteurs suivants de l'équation (24) modélisent donc les probabilités de ces paramètres. Les paramètres θ du modèle s'expriment donc ainsi : $\theta = (\theta^p; \theta^s)$ où $\theta^s = \bigcup_{\forall i:\lambda_i \in \Lambda} \theta^s_i$.

5. Pour être intéressant dans notre contexte, le modèle $P_\theta[o|s]$ doit être facile à évaluer. Dans notre cas, nous nous contenterons de générer des opérateurs o qui suivent la distribution $P_\theta[o|s]$ ce qui aura pour effet de générer des bons opérateurs avec une plus grande probabilité que si ils avaient été choisis complètement aléatoirement.

L'apprentissage automatique visera à construire une stratégie $P_\theta[o|s]$ qui est bonne en moyenne sur un ensemble de structures candidates de protéines susceptibles d'être rencontrées dans la procédure d'optimisation. L'apprentissage de la distribution $P_\theta[o|s]$ est basé sur :

1. le choix d'un vecteur de features, ou caractéristiques, $\Phi(s)$ décrivant les structures et les protéines,
2. un ensemble de distributions cibles et d'algorithmes d'apprentissage de ces distributions,
3. un ensemble d'apprentissage F représentatif de l'espace des structures susceptibles d'être rencontrées par l'algorithme d'optimisation et des objectifs visés par celui-ci.

Pour un choix de features $\Phi(s)$ et un ensemble F donnés, l'algorithme d'apprentissage aura pour objectif de choisir θ de façon à ce que la vraisemblance de F associée à (24), i.e.

$$L(F, \theta) = \prod_{(i,j,k) \in F} P_\theta[o_{i,j,k}|s_{i,j}], \tag{25}$$

soit maximale. Cette stratégie, qui porte le nom d'estimation du maximum de vraisemblance, contribue à ce que, en moyenne, les décisions les plus probables prédites par P_θ correspondent aussi à celles observées dans l'ensemble d'apprentissage F.

L'ensemble F est déjà disponible. Il nous faut maintenant définir un ensemble de features décrivant les structures des protéines et un ensemble de distributions et d'algorithmes d'apprentissage. Ces deux aspects sont décrits dans les sections 5.6.2 et 5.6.3 respectivement.

5.6.2 Features utilisées pour décrire les protéines

La manière la plus simple de décrire l'état courant d'une protéine consiste à concaténer au sein d'un vecteur toutes les positions des atomes de la protéine. Cette approche a l'avantage de décrire très précisément la protéine à un instant donné, mais construit un descripteur possédant un très grand nombre de paramètres (le nombre d'atomes de la protéine multiplié par 3). Or, nous n'avons pas besoin ici d'une description de la structure au nanomètre près, mais bien d'une estimation de quelques caractéristiques de la structure globale afin de guider l'algorithme d'optimisation qui effectue le repliement. De plus, cette approche produit des descripteurs dont la taille varie en fonction de la taille de la protéine, ce qui rend difficile la comparaison directe entre deux états courants de protéines différentes.

Nous utiliserons donc un vecteur $\Phi(s)$ de features, c'est-à-dire de caractéristiques, calculé à partir de toutes les informations dont nous disposons à propos de la protéine. Ce vecteur sera typiquement de dimension finie et fixée et ce pour toute protéine, quelle que soit sa taille. Les caractéristiques seront calculées aussi bien à partir d'informations statiques de la protéine comme sa taille ou sa séquence d'acides aminés qu'à partir de données dépendant de la structure courante de celle-ci comme l'énergie, la compacité de la protéine ou un quelconque autre facteur décrivant une structure tridimensionnelle. Le nombre de composantes du vecteur $\Phi(s)$ dépendra des features que nous souhaitons intégrer dedans, mais sera, quoi qu'il arrive, indépendant de la protéine. Dans le cadre d'une première analyse, nous avons considéré quatre features possibles : le nombre d'acides aminés, l'énergie de la structure, l'histogramme des acides aminés et un critère de compacité de la structure courante. Ces features sont relativement basiques et nécessitent d'être complétées et améliorées. La méthode présentée dans ce document a tout intérêt à se baser sur des features décrivant au mieux les caractéristiques de la protéine qui conditionnent le repliement. Cependant, il n'est pas trivial de sélectionner de telles caractéristiques, et une réflexion poussée sur le choix des features permettrait certainement d'augmenter de manière significative la performance de notre méthode. Toutes les features que nous utiliserons sont des nombres compris entre 0 et 1 et ce afin de garantir la convergence des algorithmes d'apprentissage utilisés (cf. section 5.6.3).

Les premières features que nous avons considérées sont calculées à partir de la longueur de la protéine, i.e. le nombre de ses résidus. Afin d'obtenir un vecteur de nombres compris entre 0 et 1 à partir d'un seul nombre entier, nous utilisons un principe simple qui consiste à discrétiser un intervalle $[a, b]$ [15] en un nombre n de valeurs discrètes. Le vecteur de features ainsi créé contiendra donc n éléments et peut être représenté par

$$I = (t_0, t_1, \ldots, t_{n-1}) \tag{26}$$

$$\Phi^{\text{res}} = \left(\phi_0^{\text{res}}, \phi_1^{\text{res}}, \ldots, \phi_{n-1}^{\text{res}}\right) \tag{27}$$

où $t_i - t_{i-1} = \frac{b-a}{n-1}$, $t_0 = a$, $t_{n-1} = b$ et $\sum_i \phi_i^{\text{res}} = 1$. Ensuite, lorsqu'on désire créer un vecteur de features pour un nombre $x \in [a, b]$, il suffit de considérer initialement un vecteur nul ($\phi_i^{\text{res}} = 0$ $\forall i$) et d'attribuer des poids non-nuls aux éléments ϕ_j^{res} et ϕ_{j+1}^{res} tels que $t_j \leq x \leq t_{j+1}$. Les poids ϕ_j^{res} et ϕ_{j+1}^{res} reflètent l'éloignement de la valeur x par rapport aux bornes t_j et t_{j+1}, c'est-à-dire que le ϕ_j sera d'autant plus élevé que x est proche de t_j. Par exemple, si on discrétise l'intervalle $[10, 100]$ en trois valeurs cela donnera lieu au vecteur $I_{[10,100]} = (10, 55, 100)$. Si on désire créer les features pour la valeur 10 par exemple, le résultat sera le vecteur $\Phi^{\text{res}}_{[10,100]} = (1, 0, 0)$, pour la valeur 32.5 on aura $\Phi^{\text{res}}_{[10,100]} = (0.5, 0.5, 0)$, etc. Ce procédé est ainsi appliqué à la longueur de la protéine et fournit un vecteur [16] de features qui caractérisent la protéine en fonction de son nombre de résidus. Le vecteur $\Phi^{\text{res}}(s)$ ainsi généré contient deux features actives [17] au maximum [18].

Nous appliquons le même principe afin de créer un vecteur de features pour l'énergie $\mathcal{E}(s)$. Néanmoins, une transformation est au préalable appliquée à $\mathcal{E}(s)$ afin de ramener celle-ci

15. L'intervalle considéré dans notre cas est $[10, 100]$. Celui-ci est relativement petit car nous n'avons travaillé que sur des protéines contenant moins de 100 acides aminés.
16. de taille 10 dans notre cas.
17. c'est-à-dire non nulles.
18. Une seule feature sera non-nulle si la valeur d'entrée x est égale à un t_i.

entre 0 et 1. Cette transformation consiste simplement en une sigmoïde

$$e_{\text{norm}} = \frac{1}{1 + \exp\left(-0.0005\ \mathcal{E}(s)\right)} \tag{28}$$

Un procédé similaire à celui utilisé pour le nombre de résidus est ensuite appliqué à e_{norm} afin de générer un vecteur $\Phi^{\text{en}}(s)$ possédant les mêmes propriétés que (27).

La troisième caractéristique que nous avons extraite de la protéine est son histogramme d'acides aminés. On sait qu'il existe 20 acides aminés chez les humains, le vecteur de features correspondant sera donc de taille 20. Chaque i^{e} élément du vecteur $\Phi^{\text{hist}}(s)$, noté ϕ_i^{hist}, contiendra donc la fréquence d'apparition de cet acide aminé dans la séquence primaire de la protéine à laquelle correspond la structure s.

Pour finir, la dernière feature que nous avons utilisée est un critère de compacité. Certaines études récentes ont en effet montré que la compacité d'une protéine était liée à son repliement [30], le choix de la compacité comme descripteur de la protéine n'est donc pas anodin. Les critères classiques de compacité nécessitent, pour la plupart, le calcul de la surface de l'ensemble convexe d'un nuage de points où chaque point représente la position du C_α d'un résidu. Le calcul de cette surface n'est pas trivial et nous avons décidé de nous concentrer sur un critère de notre composition, plus simple à mettre en oeuvre et évidemment moins précis mais qui permet également d'obtenir des informations à propos de la compacité de la protéine. Le critère consiste à calculer la moyenne des distances d séparant les différents résidus et à la diviser ensuite par une valeur Z. Cette valeur Z fait office de facteur de normalisation de sorte que le résultat est finalement compris entre 0 et 1. Une fois de plus la procédure standard décrite précédemment est appliquée afin de générer un vecteur $\Phi^{\text{comp}}(s)$ contenant au maximum deux features actives qui décrit la compacité de la structure s. Le facteur Z est obtenu en calculant la plus longue distance séparant les résidus dans la structure initiale. Cette structure est initialisée en forme d'hélice qui, en première approximation, possède les mêmes caractéristiques que l'hélice α. Etant donné que le tour de cette hélice est de 3.6 acides aminés et que son pas est de 5.4 Å, nous pouvons aisément déduire la longueur maximale séparant les deux résidus extrêmes d'une protéine de n acides aminés :

$$Z = d_{\max} = \frac{5.4}{3.6} n. \tag{29}$$

Le critère de compacité est finalement donné par

$$c = \frac{d}{Z}. \tag{30}$$

La valeur de c est utilisée comme entrée de la procédure de génération de features qui produira un vecteur Φ^{comp} décrivant la compacité de la protéine.

Il est maintenant possible de regrouper ces features afin de décrire, de la manière la plus complète qui soit, la protéine au moyen d'un vecteur $\Phi(s) = \Phi^{\text{res}}(s) \cup \Phi^{\text{en}}(s) \cup \Phi^{\text{hist}}(s) \cup \Phi^{\text{comp}}(s)$. La pertinence des différentes features $\Phi^{\text{res}}(s)$, $\Phi^{\text{en}}(s)$, $\Phi^{\text{hist}}(s)$, $\Phi^{\text{comp}}(s)$ ainsi que $\Phi(s)$ a été évaluée. Ces résultats sont décrits plus loin dans la section 7.

5.6.3 Description des distributions utilisées

Nous décrivons ici les algorithmes qui sont utilisés pour calculer les paramètres $\theta \in \Theta$ qui maximisent la vraisemblance $L(F, \theta)$, soit

$$\arg\max_{\theta \in \Theta} L(F, \theta). \tag{31}$$

Notre approche consiste à apprendre directement le modèle probabiliste $P_\theta[o|s]$. Cette approche est simple à mettre en oeuvre au vu de la forme de $P_\theta[o|s]$ (cf. équation (24)). En effet, nous avons modélisé notre modèle probabiliste comme le produit de plusieurs densités de probabilités. Il est donc aisé de séparer l'apprentissage en deux parties : apprentissage du choix des opérateurs et apprentissage des paramètres des opérateurs. Les lois de probabilités que nous allons estimer peuvent être discrètes (choix du type de mover, choix du résidu) ou bien continues (choix des angles ϕ et ψ, etc.). Nous commencerons par introduire deux nouvelles densités de probabilités qui, contrairement à celles présentées dans la section 5.5.3, sont conditionnelles et permettent donc de tenir compte de l'état courant de la structure afin de générer le meilleur opérateur possible.

Distribution discrète La dernière densité de probabilités discrète que nous proposons est une distribution conditionnelle qui permettra de générer une valeur discrète en fonction de features. Le modèle utilisé est un modèle notamment connu sous le nom de modèle à maximum d'entropie [36, 8]. Un tel modèle représente les probabilités de variables aléatoires Y conditionnellement à une variable aléatoire X. Ce modèle s'exprime comme suit

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} \exp\left(\sum_i \lambda_i f_i(x, y)\right) \tag{32}$$

où

$$Z_\lambda(x) = \sum_y \exp\left(\sum_i \lambda_i f_i(x, y)\right) \tag{33}$$

où x, y, λ_i et f_i représentent respectivement des réalisations des variables aléatoires X et Y, les paramètres du modèle et des features calculés entre X et Y. On peut montrer [8] que maximiser l'entropie équivaut à maximiser la log-vraisemblance de l'échantillon d'apprentissage du modèle probabiliste (32). L'algorithme d'apprentissage utilisé pour trouver les valeurs des paramètres λ_i est un algorithme itératif qui est décrit dans [8]. Cette dernière distribution a l'avantage de pouvoir générer une valeur conditionnellement à une autre.

Distribution continue Finalement, la dernière densité de probabilités continue que nous avons utilisée consiste en une distribution gaussienne dont les paramètres μ et σ sont décrits sous forme de produits scalaires. Ces produits scalaires permettent de créer une loi de probabilités qui est conditionnée par un ensemble de features Φ_c. Cette loi est donnée par

$$P_{\theta^c}[x|\Phi_c] = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-0.5\left(\frac{x-\mu}{\sigma}\right)^2\right\} \tag{34}$$

où

$$\mu = \langle \theta_\mu^c; \Phi_c \rangle \tag{35}$$
$$\sigma = \log \{1 + \exp(-\langle \theta_\sigma^c; \Phi_c \rangle)\} \tag{36}$$
$$\theta^c = \theta_\mu^c \cup \theta_\sigma^c \tag{37}$$

où Φ_c est un vecteur de *features* décrivant les caractéristiques de la structure et θ^c représente les paramètres de la distribution. Ceux-ci seront déterminés par apprentissage automatique. Cet apprentissage est réalisé via une descente de gradient stochastique qui maximise la vraisemblance des données d'apprentissage par rapport au modèle (34).

Distributions des types de movers Tout comme dans l'étape de génération des opérateurs optimaux, nous avons d'abord utilisé un vecteur de probabilités afin d'apprendre la distribution des types de movers à partir de l'ensemble d'apprentissage F. L'apprentissage est ici direct mais la distribution créée perd la notion de conditionnalité, c'est-à-dire que la distribution devient $P_{\theta_0}[m]$ indépendante de s. De plus, cette méthode peut attribuer, dans le modèle $P_{\theta_0}[m|s]$ final, une probabilité de 0 à un des types de mover si elle est utilisée sur des bases de données peu représentatives ou trop petites où un type de mover n'est pas observé. Ceci constitue évidemment une grave limitation puisque ce comportement est, de toute évidence, non désiré dans une politique d'exploration. Si un mover est moins représenté que les autres, le modèle probabiliste idéal le sélectionnera moins, voire beaucoup moins, fréquemment que les autres movers, mais se devra de générer le mover au moins quelques fois, sur un nombre suffisamment grand d'échantillonnages (ce qui n'est pas le cas avec une probabilité nulle). Cette approche doit donc être utilisée avec des bases de données F de tailles suffisamment grandes. La distribution vecteur de probabilités est très simple et une distribution plus évoluée, notamment permettant de générer un opérateur conditionnellement à la structure courante, est une perspective intéressante. Dans cette optique, le classifieur à maximum d'entropie décrit ci-dessus a été utilisé. Nous avons ainsi pu créer un modèle génératif de types de movers qui fournit, en moyenne et en fonction de la structure courante, des opérateurs que la densité de probabilités estime être intéressants étant donné la structure courante.

Distributions des paramètres de movers

1. **Distributions des indices de résidus** Le but de cet apprentissage est de créer un modèle génératif permettant d'échantillonner des valeurs optimales des paramètres étant donné le type de mover et les features décrivant la structure courante. Hélas, dans le cas des résidus, la chose n'est pas si simple puisque certaines conditions doivent être respectées lors de l'échantillonnage qui dépendent de la protéine. Typiquement, il faut que le résidu soit compris dans l'intervalle $[1; n]$ où n est la longueur de la protéine. La modélisation de ces contraintes n'est pas triviale et nous avons décidé, dans le cadre d'une première approche, de ne pas inclure le choix des résidus dans le modèle génératif. Ainsi, nous avons simplement utilisé un vecteur de probabilités générant, de manière uniforme des indices de résidus.
2. **Distributions des paramètres continus** La première distribution utilisée est bien évidemment la gaussienne. Cependant, nous ne l'avons pas utilisée en termes d'apprentissage car cela n'aurait eu que peu de sens. Effectivement, comme expliqué précédemment, cette distribution perd son utilité lorsqu'un ensemble d'apprentissage trop complexe lui

est présenté. Donc, nous avons utilisé une distribution gaussienne afin de générer des paramètres continus, mais de manière indépendante de l'état courant de la protéine. D'autre part, nous avons utilisé la gaussienne conditionnelle décrite par l'équation (34) afin de modéliser cette dépendance. Ainsi, nous avons deux politiques qui permettent de générer des paramètres continus, l'une de manière indépendente de la structure courante et l'autre de manière dépendante.

De nombreuses modélisations ont ainsi été envisagées et testées afin de déterminer la, ou les, meilleures. Nous pouvons remarquer que nous avons utilisé des distributions différentes suivant que nous étions à l'étape d'apprentissage de la base de données F ou à l'étape de génération de F (cf. section 5.5) et ceci étant donné les aspects qui différencient les deux problèmes. Dans le cas de l'apprentissage de F, les données d'observation sont disponibles en grand nombre et, vu l'utilisation qui sera faite des distributions apprises (optimisation de nouvelles protéines), on désire pouvoir extraire de l'information des structures courantes et ainsi pouvoir générer des movers adaptés à la situation. Pour ce faire, des distributions conditionnelles et complexes sont souhaitées. Le caractère conditionnel est destiné à tenir compte de la structure courante en vue de générer le mover plus adapté qui soit. En outre, plus la complexité de la distribution est grande, plus le nombre de cas que peut modéliser la distribution est grand. Ces deux propriétés sont donc bien souhaitables dans le cadre de l'apprentissage F en vue de générer une politique d'exploration complète et adaptative. Néanmoins, dans le cadre d'une première approche, nous ne chercherons pas à apprendre un modèle de complexité maximale, mais nous cherchons à apprécier les améliorations qu'apporte notre méthode. Nous pensons que l'augmentation de la complexité des modèles permettra, dans un futur travail, d'améliorer encore les performances de notre procédure.

5.7 Procédure finale d'optimisation

L'algorithme final d'optimisation dont nous allons évaluer l'efficacité ne diffère de l'algorithme utilisé pour générer les structures intermédiaires (cf. section 5.2 et 5.4) que par la politique de sélection des opérateurs. Dans notre procédure finale, les opérateurs échantillonnés, lors d'une itération de $\mathcal{A}$, sont issus de la densité de probabilités $P_\theta[o|s]$ qui dépend de la structure courante s.

Cet algorithme qui est, pour rappel, le recuit simulé, n'est peut-être pas la méthode la plus adaptée à la résolution de ce problème, mais il permet de mettre en évidence, de manière simple, les qualités et défauts de la procédure mise en place. Les résultats que cette procédure fournit sont détaillés dans la section 7.

6 Réalisation pratique

Cette section détaille divers aspects pratiques de la réalisation de la méthode proposée dans la partie 5 de ce document. L'ensemble des codes sources C++ écrits pour ce travail peuvent être trouvés à l'adresse suivante `www.d-labs.fr/cralgo-trac/browser/trunk/projects/Proteins/Rosetta`. Le dossier vers lequel pointe le lien contient l'essentiel de notre code bien que certaines fonctions présentes dans ce dossier aient été écrites par Francis Maes. Notre code a été inclus dans une suite logicielle destinée principalement à l'apprentissage inductif. Cette suite porte le nom de LBCpp et est développée par Francis Maes et Julien Becker. D'autres classes et fonctions que nous avons écrites ont été disséminées dans le reste du code de LBCpp afin d'être réutilisées dans d'autres projets.

6.1 Bases de données utilisées

Afin de mener à bien nos expériences, nous avons dû utiliser une base de données de protéines. Etant donné le temps limité dont nous disposions, nous n'avons sélectionné qu'un petit nombre de protéines sur lesquelles effectuer nos expériences. Le nombre de ces protéines s'élevait à environ 130. Rappelons que le temps nécessaire pour calculer l'énergie d'une protéine est au moins proportionnel au carré du nombre de résidus de celle-ci et que l'application d'un mover est d'autant plus coûteuse en termes de puissance de calcul que la protéine est grande. C'est pourquoi toutes les protéines sélectionnées possédaient moins de 100 acides aminés afin de rendre leur manipulation aussi rapide que possible. Ces protéines sont issues de la base de données `PDB30` utilisée dans le service *Systems and Modeling* de l'Université de Liège. Cette base de données contient des protéines dont les séquences présentent, entre elles, des taux de similarité inférieurs à 30%. Elle est elle-même subdivisée en plusieurs catégories [19] qui contiennent des nombres grandissant de protéines dont la réunion forme `PDB30` [60]. Nous avons donc choisi nos protéines parmi les plus petites de ces bases de données afin d'en avoir environ 130. La première utilisation faite de notre base fraîchement constituée a été d'appliquer à chaque protéine le recuit simulé avec une sélection aléatoire des opérateurs. Ces optimisations n'ont pas toutes pu être menées à bien et ce pour plusieurs raisons qui seront détaillées dans la section 8. Au final, seules 88 protéines ont pu être utilisées. Bien que ce nombre soit relativement faible, il s'est révélé adapté étant donné les temps de calcul moyens nécessaires à l'exécution complète de nos méthodes sur l'entièreté de la base de données. La table 2 reprend l'identifiant PDB de notre base de travail. Notre méthode finale d'optimisation a été testée sur deux jeux de données. Le premier est un sous-ensemble de la table 2 et ce dans un but de validation de la méthode. Une fois cette modalité remplie, le second ensemble permet d'évaluer notre modèle en généralisation, c'est-à-dire sur des données qui n'ont pas été utilisées dans l'apprentissage. Cet ensemble ne contient, à nouveau, qu'un faible nombre d'éléments et est tiré de la base de données utilisée pour évaluer le prédicteur de structures secondaires PSIPRED [42]. Les identifiants PDB des protéines retenues (toujours de taille inférieure à 100) sont mentionnés à la table 3.

19. `PDB30Tiny`, `PDB30Small`, `PDB30Medium`, `PDB30Boinc` et `PDB30Large`

1AP0	1K91	1LOI	1LWR	1M9G	1MKC	1NE3	1O6W	1OAI	1OF9
1OIG	1P7A	1PJV	1PTQ	1PUZ	1PZW	1QFR	1QJL	1QLO	1RES
1SPF	1T0C	1T0W	1T2Y	1TPG	1U39	1U86	1UHM	1UOY	1UQV
1V66	1W1N	1WI9	1WIH	1WII	1WO3	1WQE	1X3A	1XU6	1Z2T
1Z6H	1Z9I	1ZAE	1ZMP	1ZZV	2AYD	2BZT	2C7H	2CK5	2CP9
2D46	2D8R	2D9O	2DAL	2ESY	2GTJ	2HAC	2HN8	2HTG	2I0N
2J2S	2JMD	2JP6	2JR1	2JUI	2JWH	2JY0	2K4D	2K6L	2KBI
2KCR	2KEA	2KGH	2KL7	2KPJ	2KPS	2KT8	2KVU	2NWT	2QZF
2RM8	2RMF	2RMR	2STT	2XK0	2YT5	3FRY	4GAT		

TABLE 2 – Liste des identifiants PDB des protéines faisant partie de notre base de données de travail

1A0A	1AF0	1AGG	1AGT	1AHL	1AN4	1APF	1ARD	1ATX	1B5M

TABLE 3 – Liste des identifiants PDB des protéines faisant partie de notre base de données de validation

6.2 Fonction d'énergie

Comme mentionné précédemment, la fonction d'énergie que nous avons utilisée est la fonction *standard* de Rosetta. Cette fonction est de type *full atom*, c'est-à-dire qu'elle considère l'entièreté des atomes afin de calculer l'énergie de la protéine. Un autre type de fonction est utilisable dans Rosetta. Ce sont les fonctions *centroid* qui représentent par leur barycentre les chaînes latérales de la protéine. Ce type de fonction est donc moins précis mais également moins soumis aux erreurs.

Au cours de son utilisation, la fonction full atom a exhibé un comportement bizarre et inattendu. L'application d'un RigidBodyMover peut, dans certains cas, conduire à la division de la protéine en plusieurs parties. Ce comportement est évidemment non désiré et doit, au cours de l'optimisation, être rejeté grâce à la fonction d'énergie. Celle-ci prendrait dans ce cas une valeur très élevée indiquant que la structure générée n'est pas valide. Le phénomène de division est apparu un certain nombre de fois au cours de nos manipulations, mais l'énergie telle que calculée par Rosetta était, après division, plus faible que l'énergie de la structure précédente. Cela pose évidemment un problème majeur au vu de l'importance de la fonction d'énergie dans notre procédure. Pour résoudre ce problème, nous avons donc dû modifier le calcul de la fonction d'énergie. Pour ce faire, l'énergie est toujours calculée via l'oracle $\mathcal{E}$ fourni par Rosetta auquel nous avons ajouté un terme correctif permettant de tenir compte des distances entre les atomes de la chaîne principale. La chaîne principale est composée d'une succession d'atomes C, C_α et N reliés entre eux par des liaisons covalentes. Pour une protéine de n acides aminés, la chaîne principale est composée de $3n-1$ liaisons. Pour chacune de ces liaisons, nous calculons un facteur permettant de déterminer si la longueur d de la liaison est proche de la longueur standard d_{standard}. Ainsi, ce facteur a la forme d'une exponentielle

$$\exp\left\{\left(\frac{d-d_{\text{standard}}}{\sigma}\right)^2\right\}. \tag{38}$$

	Longueur moyenne	Ecart-type
C-C_α	1.525	0.018
C_α-N	1.466	0.02
C-N	1.323	0.02

TABLE 4 – Exemples de valeurs moyennes des différents types de liaisons présents dans la chaîne principale des protéines [52].

On sait que les longueurs des liaisons entre atomes de la chaîne principale sont très stables d'une protéine à l'autre, c'est pourquoi une telle formule peut être utilisée. Ces longueurs standards ont ainsi pu être déterminées par une analyse des protéines de la PDB [52]. La table 4 reprend ces valeurs pour les différentes liaisons possibles. La valeur de σ permet d'accepter une certaine tolérance dans la variation de la longueur, nous avons choisi 0.1 qui semblait être un bon compromis entre pénalisation des mauvaises structures et tolérance au niveau de la longueur des liaisons.

On le voit, la valeur de σ utilisée est supérieure à la valeur de l'écart-type donnée par [52]. Ceci est dû au fait que les longueurs présentées dans la table 4 ne sont pas uniques et peuvent varier en fonction du type d'acide aminé dans des proportions très faibles. C'est pourquoi une plus grande valeur de σ permet de tenir compte de cette différence naturelle. La valeur de d_{standard} tient également compte des différences de longueur de liaisons entre acides aminées puisque celle-ci est la moyenne des différentes valeurs possibles sur l'ensemble des acides aminés (cf. [52]). La formule (38) peut donc être calculée pour chaque liaison de la chaîne principale. Sa valeur sera proche de 1 si la longueur de la liaison courante est acceptable et sera plus grande si ce n'est pas le cas. Ainsi, pour des protéines dont la structure est valide, un offset a été introduit par cette formule. Pour le corriger, on soustrait $3n - 1$ à la valeur calculée de sorte que, pour des protéines dont les longueurs de liaisons se trouvent dans l'intervalle admissible, le score d'énergie donné par notre formule soit sensiblement égal à l'énergie donnée par Rosetta. Au final, l'énergie est calculée par

$$E(s) = \mathcal{E}(s) + \sum_{i \in \text{MC}} \exp\left\{ \left(\frac{d_i - d_{\text{standard}}}{\sigma} \right)^2 \right\} - 3n + 1 \tag{39}$$

où MC désigne l'ensemble des liaisons de la chaîne principale.

6.3 Algorithmes d'optimisation

Nous avons implémenté trois types d'algorithmes simples d'optimisation : algorithme glouton, algorithme de Monte-Carlo [65] et l'algorithme de recuit simulé [47]. Tous ces algorithmes fonctionnent sur le même principe, à savoir choisir aléatoirement un opérateur, l'appliquer sur la structure courante, évaluer son énergie et puis décider si la nouvelle structure est conservée. La seule différence réside dans le critère d'acceptation de la structure.

La politique gloutonne (ou greedy en anglais) conserve une structure que si celle-ci est meilleure que la précédente en termes d'énergie. Ce type de politique de décisions peut entraver l'optimisation dans un minimum local. La politique Monte-Carlo consiste à appliquer le critère de Metropolis (cf. équation (13)) avec une température constante. Ici, la politique de

décision permet, dans certains cas, de conserver une structure qui est moins bonne (en termes d'énergie) dans le but de trouver, par la suite, une structure qui est globalement meilleure que précédemment. L'influence des minima locaux est ainsi réduite. Le recuit simulé, finalement, utilise une politique de décision qui est expliquée dans la section 5.2.

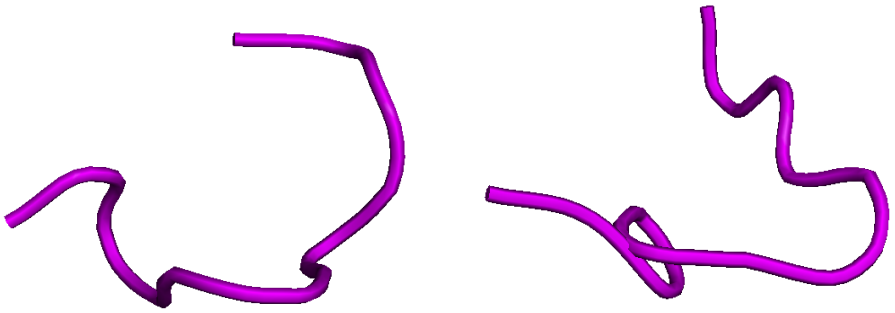
En vue d'une utilisation dans notre problème, l'algorithme glouton est tout de suite exclu à cause de sa propension à trouver des minima locaux. Nous avons donc dû choisir entre l'algorithme de Monte-Carlo et le recuit simulé. Nous avons choisi ce dernier car il permet plus de flexibilité, au travers de ses paramètres (température initiale, température finale et décroissance de la température), que l'algorithme de Monte-Carlo. Nous avons brièvement testé ces trois algorithmes sur une protéine jouet composée de 12 alanines. La figure 18 illustre la structure initiale telle que créée par Rosetta. Les figures 19, 20 et 21 représentent respectivement les résultats de l'optimisation sur un nombre de 10^5 itérations de cette structure au moyen de l'algorithme glouton, de l'algorithme de Monte-Carlo et du recuit simulé. Une structure est également illustrée pour une optimisation de 10^6 itérations par l'algorithme glouton. On remarque directement que l'optimisation gloutonne donne de piètres conformations malgré un nombre élevé d'itérations. Par contre, les résultats de l'algorithme de Monte-Carlo et du recuit simulé sont bien meilleurs et fort similaires dans le cas de cette protéine très simple. On peut logiquement supposer que les différences apparaissent au fur et à mesure que la complexité de la structure augmente.

FIGURE 18 – Conformation initiale lors de la création de la protéine composée de 12 alanines.

Les test précédents ont été obtenus sur une protéine jouet qui nous a permis d'observer rapidement le comportement des différents algorithmes. Les protéines sur lesquelles nous travaillons sont évidemment plus grosses et de structure plus complexe. Les paramètres que nous avons utilisés pour optimiser ces protéines sont :

- Température initiale : $kT = 4$,
- Température finale : $kT = 0.01$,
- Décroissance de la température : par paliers, avec 50 paliers.

Ces valeurs ont été choisies sur base de celles utilisées dans certains exemples présentés dans un tutoriel officiel de Rosetta. Il est évident qu'une analyse plus poussée de l'influence de ces paramètres est nécessaire afin de les sélectionner au mieux. Cependant, cette analyse dépasse



(a) 10^5 itérations (b) 10^6 itérations

FIGURE 19 – Conformation obtenue après l'application de l'algorithme d'optimisation locale.

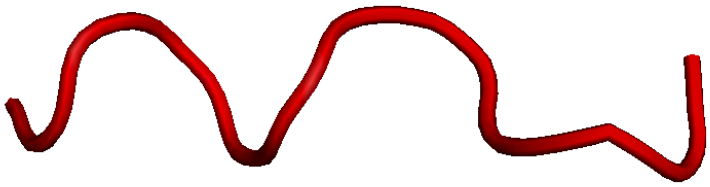


FIGURE 20 – Conformation obtenue après l'application de l'algorithme d'optimisation de Monte-Carlo sur la protéine créée.

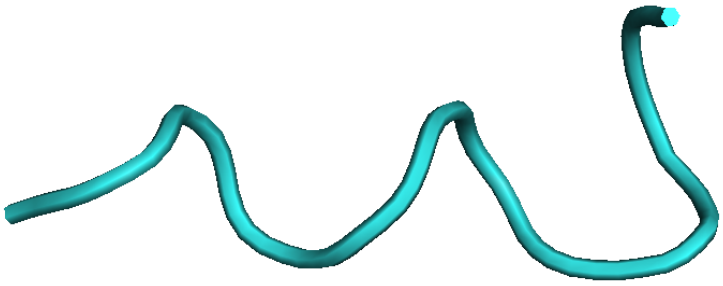


FIGURE 21 – Conformation obtenue après application du recuit simulé sur la protéine créée.

le cadre de ce travail dont le but consiste à démontrer l'efficacité de l'approche *learning for search*.

6.4 Environnements d'exécution

Le problème de la prédiction de structure tertiaire de protéines demande une très grande puissance de calcul. Plusieurs facteurs déjà évoqués dans ce qui précède justifient ce constat. Premièrement, le problème de prédiction tel que nous l'avons formulé ici consiste en un problème d'optimisation très difficile caractérisé par un espace de recherche exponentielle-

ment grand. De plus, les opérations qui doivent être réalisées à chaque étape de l'optimisation (évaluation de l'énergie, application d'un opérateur sur la structure) sont non-triviales puisque le nombre d'opérations qu'elles requièrent dépend lui-même de la taille du problème. Ces deux facteurs combinés impliquent que la manipulation et, plus particulièrement, l'optimisation de structures de protéines est un problème très exigeant en termes de puissance de calcul. Dans des suites logicielles évoluées, telles que Rosetta, les programmes sont parallélisés afin d'accélérer les temps de calcul. Hélas, dans notre application d'optimisation par recuit simulé, la parallélisation est relativement complexe à mettre en oeuvre puisque la succession des opérations suit un cheminement bien précis dont l'ordre ne peut être interverti. Néanmoins, la parallélisation est possible si on souhaite optimiser plusieurs protéines simultanément. C'est cette voie qui a été suivie dans ce travail. Hélas, vraisemblablement à cause de conflits internes à Rosetta, cette implémentation présentait des bogues. Nous avons cependant préféré nous consacrer à la réalisation d'une méthode pertinente plutôt qu'à la recherche de l'origine de ces bogues qui pouvait être très difficile à trouver. Afin d'exécuter nos programmes, nous avons eu la chance de pouvoir profiter des infrastructures offertes par l'Université. Ainsi, la plupart de nos programmes ont été exécutés soit sur `nic3` (le supercalculateur de l'Université), soit sur `monster24`, un serveur possédant 24 coeurs d'exécution disponible au sein du service. Sans cette infrastructure, la réalisation de ce travail aurait été bien moins évidente.

7 Résultats

7.1 Evaluation des différents types de movers avec le recuit simulé de base

Nous avons appliqué, aux protéines issues de la table 2 (cf. page 57), l'algorithme naïf de recuit simulé avec différents types de movers : PhiPsiMover, PhiPsiMover avec une distribution de probabilités gaussienne, ShearMover, RigidBodyMover avec une composante de rotation nulle (uniquement translation), RigidBodyMover avec une composante de translation nulle (uniquement rotation) et RigidBodyMover complet. Tous les movers considérés génèrent leurs paramètres continus aléatoirement en échantillonnant des valeurs provenant de distributions uniformes, la seule exception est le PhiPsiMover avec distribution gaussienne qui utilise une distribution éponyme. Dans tous les cas, une distribution uniforme est utilisée afin de sélectionner les résidus sur lesquels le mover est appliqué. Le but de ces expériences est d'évaluer la qualité globale des différents types de movers.

Les figures 22, 23, et 24 représentent respectivement, en fonction des itérations, l'énergie moyenne calculée sur les protéines optimisées, la mesure de similarité AL0 entre les structures courantes et leur structure cible et le QScore calculé entre les structures courantes et leur structure cible. Les énergies sont toujours exprimées en kCal/mol tandis que les QScores et les mesures AL0 sont adimensionnels. Les paramètres de l'algorithme sont $kT_{\text{initial}} = 4$, $kT_{\text{final}} = 0.01$ avec une décroissance de kT par paliers, avec 50 paliers (voir aussi la section 6.3). Notons enfin que le nombre d'itérations réalisé par l'algorithme d'optimisation est de 10^6. Bien que les algorithmes aient été appliqués à l'entièreté du jeu de données, les résultats présentés ici concernent un sous-ensemble de 11 protéines. La raison de cela est essentiellement pratique car la récolte des données provenant de toutes les optimisations aurait nécessité trop de temps. De plus, un sous-ensemble de protéines permet de déduire les mêmes résultats que si l'analyse avait été effectuée sur la totalité de la table 2.

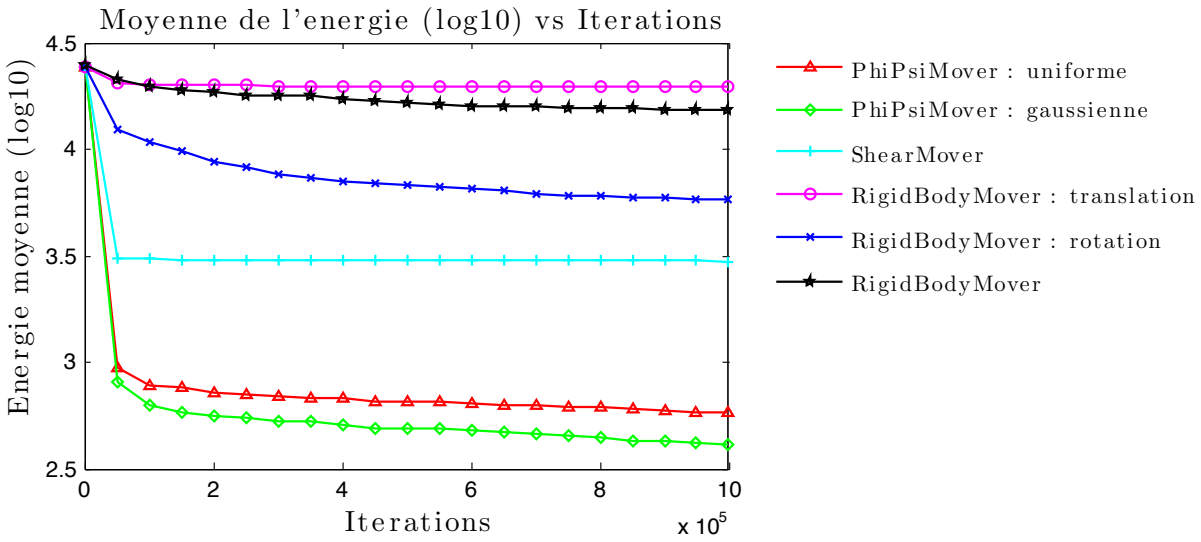


FIGURE 22 – Evolution de l'énergie au cours de l'optimisation. L'échelle de l'axe des énergies est ici logarithmique afin d'améliorer la lisibilité du graphe.

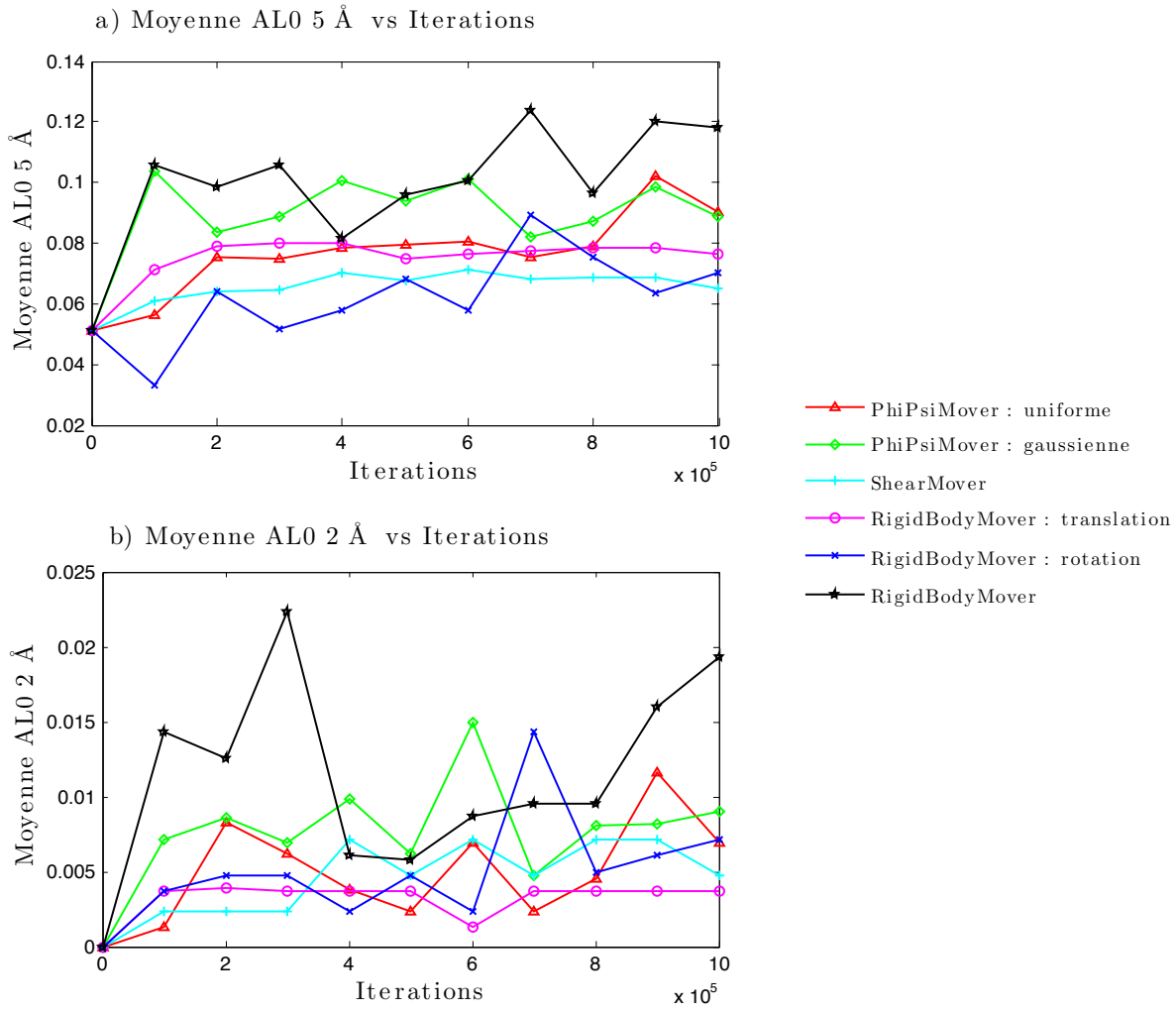


FIGURE 23 – Moyenne des scores AL0 évalués pour des distances de cutoff de 2 et de 5 Å. Le score AL0 est calculé entre les structures en cours d'optimisation et leur structure correcte à différents instants de l'optimisation. La moyenne est finalement évaluée sur tous les scores obtenus pour une même itération.

Comme l'indiquent les graphes des figures 22, 23, et 24, les structures optimisées s'améliorent globalement au cours du processus. En effet, ce constat est confirmé par la décroissance de l'énergie ainsi que par la croissance des mesures de similarité au fil des itérations. Cependant, malgré des améliorations perceptibles, les structures générées sont encore très éloignées des structures réelles, même après 10^6 itérations. En effet, les QScores tertiaires [20] atteignent péniblement 0.04, tandis que les mesures AL0, pour une distance de cutoff de 5Å, ne dépassent pas les 12%. Globalement, les protéines optimisées, via cette procédure naïve et ce choix basique des opérateurs, conservent une structure relativement filiforme et très peu compacte.

20. Rappelons qu'un QScore de 0.4 correspond à un RMSD d'environ 6 Å. Les valeurs calculées pour nos structures sont très faibles indiquant ainsi que la structure est encore loin de la conformation optimale.

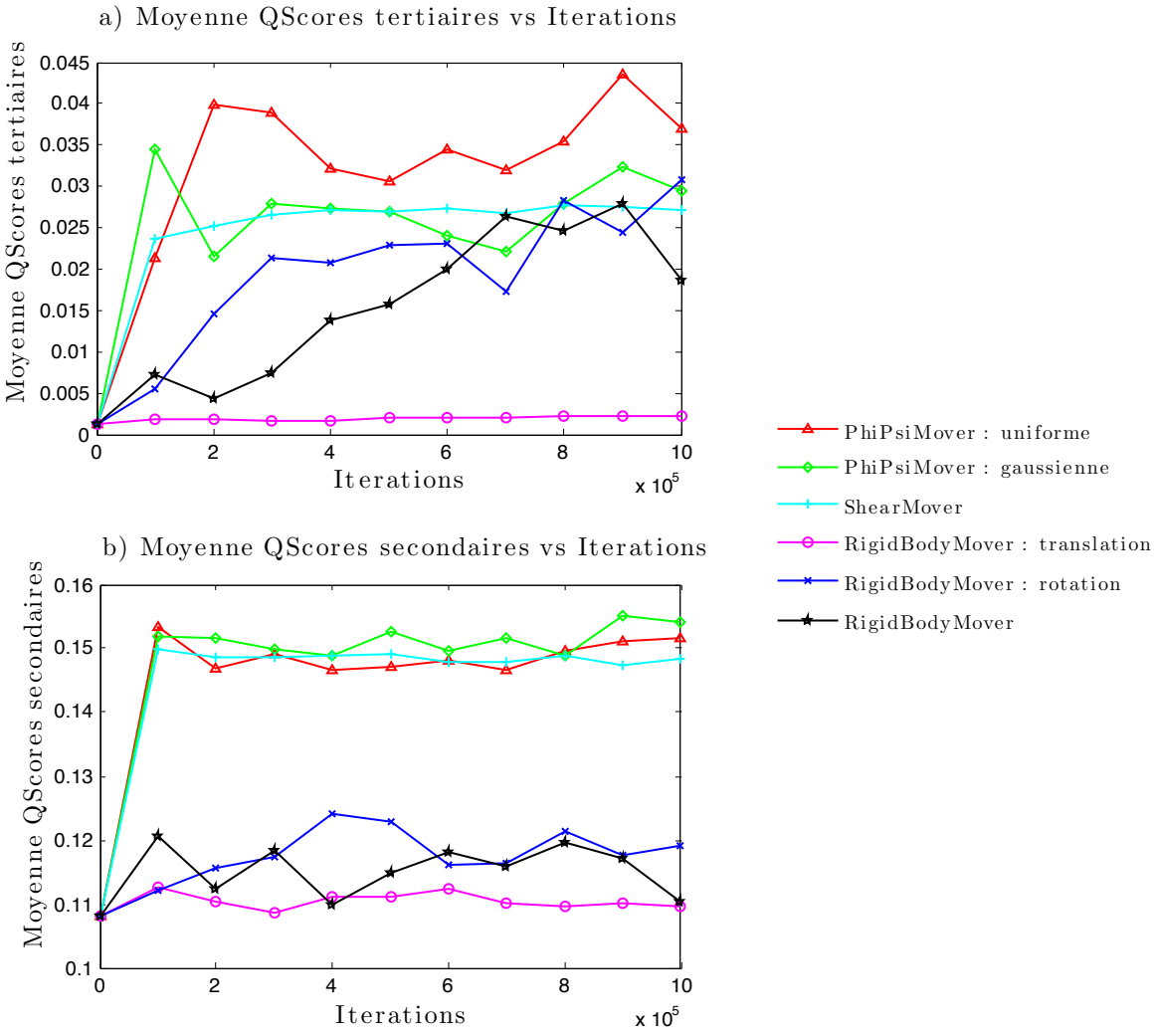


FIGURE 24 – Moyenne des QScores évalués pour les structures tertiaires et secondaires. Les QScores sont, d'abord, calculés entre les structures en cours d'optimisation et leur structure correcte à différents instants de l'optimisation. La moyenne est finalement évaluée sur tous les scores obtenus pour une même itération.

Ainsi, les valeurs du score AL0 indiquent que moins de 12% des résidus sont correctement alignés avec une tolérance de 5 Å, ce qui est évidemment très peu.

Le but réel de ces expérimentations est d'évaluer, grossièrement, la qualité des différents opérateurs. On le voit aisément sur ces figures, le PhiPsiMover est le mover le plus efficace, que ce soit en termes de diminution de l'énergie ou en termes de similarité avec la structure cible bien que sa supériorité soit moins marquée dans la mesure AL0. Les RigidBodyMover sont, ainsi, sensiblement moins performants que les autres types de movers. Cela s'explique certainement par leur complexité plus élevée qui est surtout utile lorsque la protéine est déjà, en partie, repliée, or, comme l'indiquent les précédents résultats (cf. figures 23 et 24), ces struc-

tures sont encore relativement mauvaises même en fin d'optimisation. Comme on le comprend aisément, le caractère filiforme et non compact des structures explique très probablement l'inefficacité des RigidBodyMovers. Notons finalement que le RigidBodyMover avec translation réalise de très mauvaises performances. Ceci est probablement dû à la structure telle qu'elle a été initialisée (pour rappel, sous forme d'hélice). Appliquer des translations entre résidus qui sont, pour la plupart, alignés ne produira de toute évidence pas de bons résultats. Le ShearMover, quant à lui, bien que meilleur que les RigidBodyMovers, n'est pas aussi efficace que le PhiPsiMover. Ce constat n'est pas surprenant puisque le ShearMover a été spécifiquement développé afin d'agir sur la structure de manière locale, ses moins bonnes performances, lors des premières étapes de l'optimisation d'une protéine, sont donc tout à fait compréhensibles. Vraisemblablement, de telles figures permettent de conclure que les ShearMovers et les RigidBodyMovers sont moins efficaces que les PhiPsiMovers lorsque l'optimisation en est à ses débuts et, *a fortiori*, lorsque des changements de structures globaux sont nécessaires. Nous sommes convaincus, cependant, que chaque mover a son utilité en des circonstances bien particulières. Néanmoins, la démonstration de cette hypothèse dépasse le cadre de ce travail.

7.2 Génération de la base d'apprentissage

Cette section présente les résultats associés à la génération de la base d'apprentissage F.

7.2.1 Génération des structures intermédiaires

Les structures intermédiaires utilisées dans la base d'apprentissage sont celles qui ont été générées via l'algorithme naïf de recuit simulé appliqué aux structures de la table 2 (voir page 57) avec, comme seul mover, le PhiPsiMover. Ces structures intermédiaires ont été récoltées au cours de l'optimisation qui a permis de générer les graphes de la section 7.1 précédente. Les structures intermédiaires ont été générées 10 fois à intervalles réguliers au cours de cette optimisation. Le nombre d'itérations de l'algorithme étant de 10^6, les structures ont donc été sauvegardées toutes les 10^5 itérations. L'optimisation ayant permis de générer les structures correspond à la courbe rouge de la figure 22. L'ensemble des structures intermédiaires ainsi générées est noté $\mathcal{S}$.

7.2.2 Génération des meilleurs opérateurs

L'algorithme EDA a été appliqué sur les structures intermédiaires $s \in \mathcal{S}$ (cf. section 7.2.1) afin de déterminer, pour chaque structure s, les meilleurs movers permettant d'améliorer la structure. Nous avons utilisé diverses distributions ainsi que diverses fonctions F_{proxy}. Tout d'abord, afin de garantir une certaine diversité dans les movers générés, nous avons utilisé des vecteurs de probabilité avec taux d'apprentissage (cf. section 5.5.3, page 45) aussi bien pour la modélisation des distributions des movers que des résidus. Dans un deuxième temps, nous avons utilisé, pour les movers et les résidus, des vecteurs de probabilité sans taux d'apprentissage (cf. section 5.5.3, page 45). Les distributions des paramètres continus ont été modélisées, dans les deux cas, par des mixtures de gaussiennes. Ces diverses modélisations ont été mises en oeuvre afin de générer, respectivement, un certain nombre de bons movers différents et un seul très bon mover.

Nous avons, de plus, utilisé trois fonctions F_{proxy} différentes. Les différences entre celles-ci se limitent à la pondération μ des termes d'énergie et de structure (cf. section 5.5.1, page

42). Le but est ici d'observer le comportement de l'EDA et de la procédure d'optimisation finale (voir section 7.3) lorsque des movers qui sont considérés comme optimaux, suivant des critères différents, sont utilisés. Dans nos trois fonctions proxy utilisées, le terme d'énergie est pondéré par les facteurs 0, 0.5 et 1, les facteurs qui pondèrent le terme de structure découlent directement de ceux de l'énergie puisque leur somme doit valoir 1.

Lors de chaque itération de l'EDA, la moyenne des gains obtenus en termes de fonctions proxy, $\overline{\Delta F_{\text{proxy}}}(s)$, en termes d'énergies, $\overline{\Delta \text{E}}(s)$, et de QScore, $\overline{\Delta \text{QScore}}(s)$, est calculée sur l'ensemble des opérateurs générés à cette itération et ce pour chaque structure $s \in \mathcal{S}$. Cela permet d'observer l'évolution des distributions des movers. Nous pourrons ainsi vérifier que ces distributions convergent bien vers des distributions qui génèrent de bons opérateurs en moyenne. La moyenne des $\overline{\Delta F_{\text{proxy}}}(s)$, $\overline{\Delta \text{E}}(s)$ et $\overline{\Delta \text{QScore}}(s)$ est ensuite évaluée sur l'ensemble des structures $s \in \mathcal{S}$ pour fournir des valeurs globales $\overline{\Delta F_{\text{proxy}}}$, $\overline{\Delta \text{E}}$ et $\overline{\Delta \text{QScore}}$. Ces valeurs sont reprises à la table 5 pour la première et la dernière itération de l'EDA. Les formules suivantes reprennent les définitions mathématiques des différentes valeurs introduites ci-dessus pour une itération donnée de l'EDA.

$$
\begin{aligned}
\overline{\Delta F_{\text{proxy}}}(s) &= \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \left(F_{\text{proxy}}(s, m, s^*) - F_{\text{proxy}}(s, \mathbb{I}, s^*) \right) \\
\overline{\Delta \text{E}}(s) &= \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \left(\mathcal{E}(s) - \mathcal{E}(s(m)) \right) \\
\overline{\Delta \text{QScore}}(s) &= \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \left(\text{Q}_{\text{score}}(s(m), s^*) - \text{Q}_{\text{score}}(s, s^*) \right) \\
\overline{\Delta F_{\text{proxy}}} &= \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \overline{\Delta F_{\text{proxy}}}(s) \\
\overline{\Delta \text{E}} &= \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \overline{\Delta \text{E}}(s) \\
\overline{\Delta \text{QScore}} &= \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \overline{\Delta \text{QScore}}(s)
\end{aligned}
$$

où m désigne un mover parmi l'ensemble $\mathcal{M}$ de movers générés à cette itération par l'EDA, $\mathbb{I}$ désigne l'opérateur identité, s^* la structure cible de la protéine et l'opérateur $|\cdot|$ désigne la cardinalité d'un ensemble.

Tout d'abord, notons que les valeurs calculées ici, lorsqu'elles sont positives, indiquent que, en moyenne, la structure est améliorée après application d'un mover échantillonné via la distribution courante, l'effet inverse est constaté si les valeurs sont négatives. Le but de l'EDA est donc de maximiser les scores repris dans la table 5 et donc de sélectionner, en moyenne, des movers qui permettent d'obtenir des améliorations sur notre structure. Idéalement, tous les scores devraient être positifs. Une première observation est que ce but n'est pas atteint. Cependant, il importe également de remarquer l'évolution de ces valeurs. En effet, on constate que, dans tous les cas, les valeurs $\overline{\Delta F_{\text{proxy}}}$ sont toujours plus grandes à la dernière itération qu'à la première. Ce phénomène indique donc bien une évolution : bien que, en moyenne, les movers échantillonnés ne produisent pas d'amélioration sur la structure, leur effet qui était,

en moyenne, négatif est diminué. Cela veut dire que la proportion de bons movers générés augmente.

Les scores $\overline{\Delta\text{QScore}}$ présentent également la même croissance dans tous les cas, indiquant, de ce fait, que, quelles que soient les distributions ou la fonction proxy choisies, le QScore est, en moyenne, toujours amélioré après application d'un opérateur échantillonné. Cependant, le même constat ne peut être établi en ce qui concerne $\overline{\Delta\text{E}}$. En effet, lorsque $\mu = 0$, c'est-à-dire lorsque la fonction proxy ne tient compte que de la structure, on constate une détérioration globale de l'énergie. Ces observations semblent indiquer que des améliorations en termes de structure n'induisent pas nécessairement des améliorations en termes d'énergie. Néanmoins, étant donné l'hypothèse que la structure optimale correspond à un minimum d'énergie, cette observation met en évidence un bon moyen d'éviter les minima locaux de la fonction d'énergie. Par contre, on remarque que des améliorations en termes d'énergie entraînent toujours des améliorations, bien que moins marquées, en termes de structure.

Distributions	μ	$\overline{\Delta F_{\text{proxy}}}$		$\overline{\Delta\text{E}}$		$\overline{\Delta\text{QScore}}$	
		Initial	Final	Initial	Final	Initial	Final
Avec taux d'apprentissage	0	−0.0045	0.0789	−2855	−13640	−0.0043	0.0792
	0.5	−0.219	0.00282	−2976	−76.37	−0.0043	0.0160
	1	−0.432	−0.0014	−2965	−38.88	−0.0041	−0.0010
Sans taux d'apprentissage	0	−0.0045	0.0965	−2909	−15740	−0.0043	0.0972
	0.5	−0.214	0.0519	−2788	335.7	−0.0041	0.0230
	1	−0.425	0.087	−2797	359.8	−0.0043	−0.0011

TABLE 5 – Génération des meilleurs opérateurs. Cette table représente les gains, en moyenne sur l'ensemble des structures intermédiaires et sur l'ensemble des opérateurs aléatoirement sélectionnés, obtenus sur la structure après application de l'opérateur échantillonné. Ces valeurs correspondent à celles obtenues à la première et à la dernière itération de l'EDA.

Globalement, on remarque que les optimisations menées avec des distributions avec taux d'apprentissage fournissent de moins bons résultats en moyenne que les distributions pures. Ce fait s'explique par la diversité, volontairement maintenue, qui entraîne donc la génération de movers plus variés, mais moins bons en moyenne. Les distributions sans taux d'apprentissage, par contre, convergent beaucoup mieux mais au détriment de la diversité. De fait, les distributions avec taux d'apprentissage convergent vers un seul bon mover (donc pas de diversité), mais celui-ci a l'avantage d'être très bon. Cependant, il n'est pas possible de déterminer si les movers optimaux, trouvés à la fin de la procédure, sont meilleurs dans un cas que dans l'autre.

Nous avons calculé la corrélation ρ entre les colonnes 6 et 8 de la table 5. Celle-ci vaut $\rho = -0.973$. Ce résultat montre que, généralement, un bon résultat en termes d'énergie ne peut être concilié avec un bon résultat en termes de structure puisque les deux évoluent en sens inverse ($\rho < 0$). Cependant, cette étude se base sur des movers considérés comme bons, localement, sur des structures rencontrées en début d'optimisation. On peut, vraisemblablement, penser que la corrélation devient positive dans les dernières phases du repliement de la protéine conciliant ainsi la minimisation de l'énergie et la maximisation du QScore.

Les figures 25, 26 et 27 représentent l'évolution des valeurs $\overline{\Delta F_{\text{proxy}}}$, $\overline{\Delta \text{E}}$ et $\overline{\Delta \text{QScore}}$ en fonction des itérations pour une pondération $\mu = 0.5$. Comme attendu, ces courbes sont croissantes indiquant ainsi une amélioration des distributions générant les opérateurs. On remarque, de plus, que les courbes sans apprentissage convergent plus vite et vers de meilleurs opérateurs.

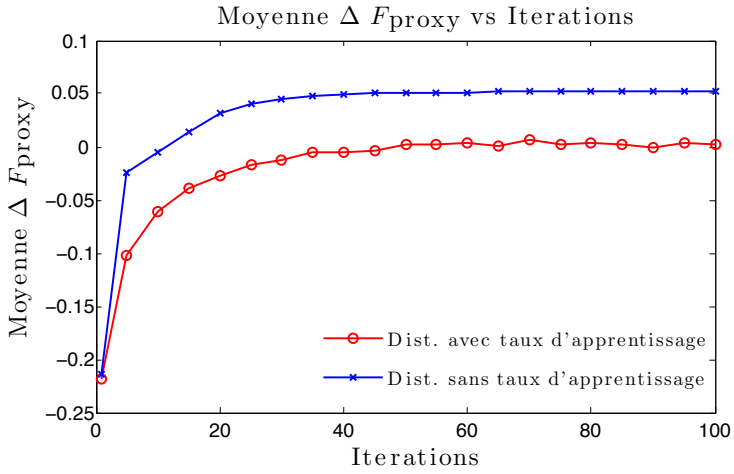


FIGURE 25 – Evolution de $\overline{\Delta F_{\text{proxy}}}$ en fonction des itérations pour $\mu = 0.5$.

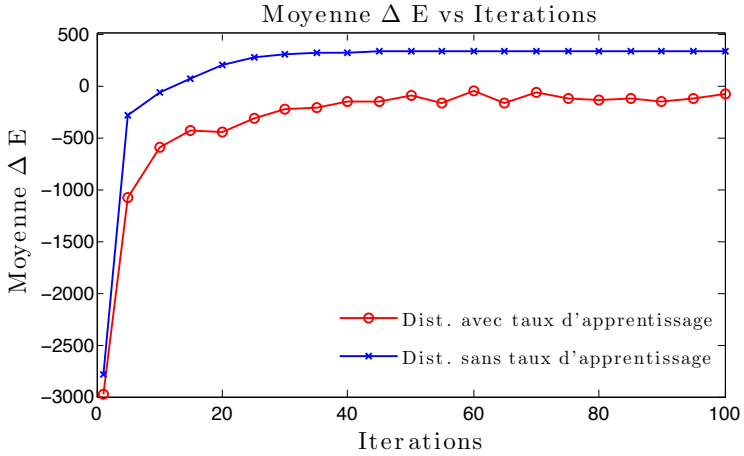


FIGURE 26 – Evolution du $\overline{\Delta \text{E}}$ en fonction des itérations pour $\mu = 0.5$.

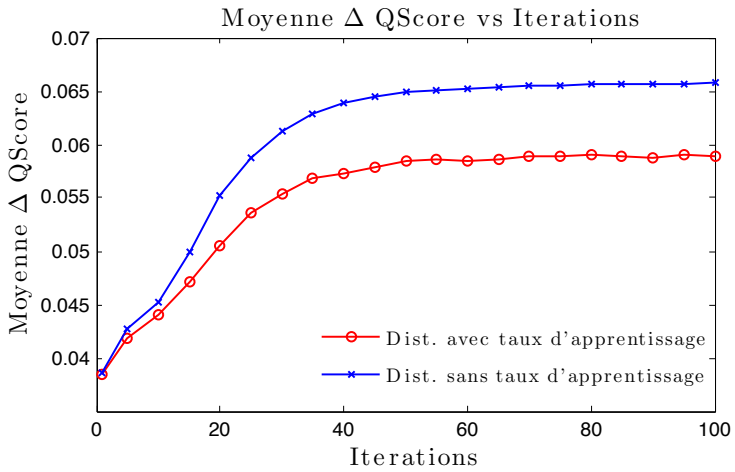


FIGURE 27 – Evolution de $\overline{\Delta \text{QScore}}$ en fonction des itérations pour $\mu = 0.5$.

7.3 Evaluation des politiques d'exploration obtenues par apprentissage automatique

La procédure d'optimisation finale a été évaluée sur deux jeux de données différents A et B. Le premier (A) est un sous-ensemble composé de 11 protéines parmi celles représentées à la table 2, voir page 57. Cette étape permet de vérifier le procédé d'apprentissage et permet d'observer que des améliorations, en termes d'optimisation, sont effectivement apportées par la méthode en comparaison avec l'optimisation sans apprentissage. Le deuxième jeu de données (B) est représenté à la table 3 (voir page 57). Il est utilisé afin de vérifier que notre méthode fonctionne en généralisation et que l'algorithme ne surapprend pas le problème.

Lors de nos tests, nous avons utilisé 12 configurations d'apprentissage différentes. La base d'apprentissage que nous avons utilisée est constituée d'approximativement 100 structures intermédiaires chacune associée au meilleur mover rencontré pour celles-ci. La taille de ce jeu de données est ainsi d'environ 100 couples structure-mover. L'algorithme d'optimisation, quant à lui, utilise les paramètres classiques du recuit simulé. Cependant, le nombre maximum d'itérations a été diminué à 250000 (afin d'accélérer les temps de calcul).

Les différentes configurations d'apprentissage sont détaillées dans le tableau 6. Dans les colonnes *Apprentissage* de ce tableau, un trait d'union - signifie que le paramètre correspondant n'est pas appris et que les distributions initiales sont utilisées. Sinon, le nom de la distribution est utilisé. De même, un trait d'union - dans une des colonnes *Features* indique que la feature en question n'est pas utilisée, un $\surd$ indiquant par contre que l'algorithme se base sur cette feature afin de générer un bon mover pour la structure courante. La configuration 1, que nous appelons naïve, consiste à sélectionner les trois types de movers de manière complètement aléatoire en utilisant des vecteurs de probabilités afin de sélectionner le type de mover et les résidus et des distributions gaussiennes afin de générer les paramètres continus. Cette configuration nous servira de point de repère pour évaluer notre procédure.

	Apprentissage		Features			
	Mover	Param. continus	Long.	Energie	Histo-gramme	Compa-cité
Config. 1 (naïf)	-	-	-	-	-	-
Config. 2	Vec. prob.	-	-	-	-	-
Config. 3	Max ent.	-	√	-	-	-
Config. 4	Max ent.	-	-	√	-	-
Config. 5	Max ent.	-	-	-	√	-
Config. 6	Max ent.	-	-	-	-	√
Config. 7	Max ent.	-	√	√	√	√
Config. 8	Max ent.	Gauss. cond.	√	-	-	-
Config. 9	Max ent.	Gauss. cond.	-	√	-	-
Config. 10	Max ent.	Gauss. cond.	-	-	√	-
Config. 11	Max ent.	Gauss. cond.	-	-	-	√
Config. 12	Max ent.	Gauss. cond.	√	√	√	√

TABLE 6 – Configurations d'apprentissage. Dans cette table, « Vec. prob. » correspond à vecteur de probabilités (cf. section 5.5.3, page 45), « Max ent. » correspond à classifieur à maximum d'entropie (cf. section 5.6.3, page 53) et « Gauss. cond. » correspond à la gaussienne conditionnelle (cf. section 5.6.3, page 53).

Afin de déterminer si une configuration est meilleure que l'optimisation naïve, nous avons effectué un test statistique. Plus précisément, nous avons vérifié que les différences étaient statistiquement significatives au seuil de signification 10%. Pour ce faire, nous calculons d'abord la moyenne m et l'écart-type σ des énergies obtenues lors de l'optimisation avec la configuration naïve 1.

Nous supposons que les valeurs finales de l'énergie suivent une distribution normale. Ensuite, nous déterminons pour quelle valeur de x tirée de $X \sim \mathcal{N}(0,1)$ l'aire sous la courbe vaut 10%, c'est-à-dire

$$\int_{-\infty}^{x} X = 0.1 \tag{40}$$

Nous trouvons (dans les tables consacrées) que cette valeur vaut approximativement $x = -1.28$. Nous pouvons, maintenant, déterminer V tel que, si une valeur est inférieure à V, on peut dire que cette valeur est inférieure à m au seuil de signification 10%. V est ainsi calculé par

$$V = -1.28\sigma + m.$$

Nous allons analyser les performances de notre méthode sur les protéines d'apprentissage (A) et de test (B) en utilisant les divers ensembles de movers générés (grâce aux différentes fonctions proxy pour $\mu = 0$, $\mu = 0.5$ et $\mu = 1$). Ces résultats sont repris à la table 7. Ces optimisations ont été réalisées en apprenant le modèle sur l'ensemble d'apprentissage (100-1). Les valeurs en gras dans cette table correspondent aux meilleures valeurs (d'un point de vue statistiquement significatif[21]). Les valeurs des moyennes et écart-types sont calculées

21. Rappelons que cette méthode, pour être parfaitement valide, devrait reposer sur des statistiques plus fiables et donc nécessiteraient un grand nombre de valeurs afin de déterminer les moyennes et écart-type de

uniquement à partir de la table 7 et valent

$$\begin{aligned} m_{\mathrm{A}} &= 1510.8, \\ \sigma_{\mathrm{A}} &= 56.32, \\ V_{\mathrm{A}} &= 1438.7, \\ m_{\mathrm{B}} &= 774.62, \\ \sigma_{\mathrm{B}} &= 86.52, \\ V_{\mathrm{B}} &= 663.87. \end{aligned}$$

	Moyenne des énergies finales (kCal/mol)					
	A (Ensemble app.)			B (Ensemble test)		
	$\mu = 0$	$\mu = 0.5$	$\mu = 1$	$\mu = 0$	$\mu = 0.5$	$\mu = 1$
Config. 1	1492.9	1573.93	1465.64	834.71	813.71	675.45
Config. 2	3437.19	**646.74**	**727.65**	1092.34	**376.32**	**484.79**
Config. 3	1510.14	**1331.96**	1565.85	736.68	860.96	828.64
Config. 4	2228.85	**1208.73**	1758.45	748.71	822.36	795.98
Config. 5	**1220.00**	1685.65	1492.38	709.08	745.02	731.09
Config. 6	2111.84	**1277.50**	1556.68	785.71	730.83	725.95
Config. 7	**1408.74**	**1324.44**	1618.24	946.09	769.68	**634.18**
Config. 8	1532.93	**480.27**	1634.73	1865.87	**554.76**	1095.58
Config. 9	2469.69	**504.47**	**930.39**	1198.19	**396.61**	724.21
Config. 10	1846.10	**405.90**	1586.41	797.22	**312.44**	947.86
Config. 11	2647.50	**485.46**	1596.15	985.57	**296.82**	724.79
Config. 12	22315.12	**426.28**	**1189.35**	980.11	**288.89**	**581.98**

TABLE 7 – Comparaison des énergies des structures finales obtenues après diverses configurations d'apprentissage et d'optimisation. L'apprentissage a été réalisé sur la base de données (100-1).

Nous remarquons tout de suite que notre méthode s'avère efficace dans un certain nombre de cas, en particulier lorsque les distributions modélisant les variables continues sont utilisées. Notre procédure ne fonctionne cependant pas dans tous les cas de figures et cela démontre l'importance de l'apprentissage aussi bien que de la sélection des features décrivant la protéine. D'un point de vue général, on voit que lorsque toutes les features sont utilisées (Config. 7 et 12), les optimisations réagissent très bien, que ce soit sur l'ensemble d'apprentissage A ou sur l'ensemble de test B. Nous pensons que des distributions plus complexes et dépendant, en plus de la structure courante, des résidus et du type de mover, pourraient être utilisées afin d'encore améliorer les performances de l'optimisation. De plus, les résultats des Config. 7 et 12 tendent à montrer que plus les features décrivant les protéines sont riches, meilleures est la détermination des movers. Ce constat encourage donc le développement de nouveaux descripteurs de protéines plus performants.

manière précise.

Le fait le plus remarquable de la table 7 est que l' ensemble d'apprentissage constitué avec $\mu = 0$ et, dans une moindre mesure, celui généré avec $\mu = 1$, réalisent de piètres performances. Les raisons de ce phénomène concernant l'ensemble $\mu = 1$ ne sont pas claires. Cependant, une explication vraisemblable pour l'ensemble $\mu = 0$ peut être avancée. Les movers générés avec $\mu = 0$ sont censés être bons en termes de structure, mais l'optimisation se base elle, pour l'acceptation d'une structure candidate, uniquement sur l'évaluation de l'énergie. Or, comme nous l'avons vu dans la table 5, page 67, les movers optimaux en termes de structure sont généralement très mauvais en termes d'énergie. Les movers, générés par notre modèle génératif, censés être bons en termes de structure, entraînent des augmentations d'énergie qui, par la suite, provoque la non-acceptation, par l'algorithme, de la structure candidate. Alors, seuls les quelques movers, mauvais au sens du modèle génératif, mais bons en termes d'énergies, sont efficaces aux yeux de l'algorithme d'optimisation. Ce constat met en évidence l'intérêt que pourrait présenter un critère, autre que l'énergie, permettant d'évaluer la qualité d'une structure.

A partir de ce tableau, nous pouvons, de plus, mettre en évidence un autre problème lié à Rosetta. Les valeurs de ce tableau situées à la première ligne et correspondant à un même ensemble d'évaluation, devraient être égales puisque les protéines optimisées sont les mêmes, de même que l'ensemble des paramètres. De plus, le flux des nombres aléatoires utilisé dans notre procédure, afin de générer les movers et leurs paramètres, est initialisé de la même manière lors de toutes les expériences. Le résultat devrait donc être le même pour toutes les optimisations naïves d'un même ensemble (A ou B). On voit que ce n'est cependant pas le cas. Cette différence remarquable provient très probablement de Rosetta qui utilise vraisemblablement des nombres aléatoires en son sein afin de calculer l'énergie (potentiels statistiques notamment) d'une structure. Cela expliquerait les différences qui existent entre les différentes colonnes de la première ligne du tableau. Les résultats doivent donc être analysés avec des précautions relatives.

Afin d'observer le comportement de notre algorithme, nous allons étudier plus en détails les couples d'optimisation (Conf. 1, Conf. 12) sur le jeu de données d'apprentissage (A) et sur le jeu de données de test (B). Les figures 28 et 29 représentent l'évolution de l'énergie moyenne calculée sur toutes les protéines optimisées à différents instants de l'optimisation, respectivement pour le jeu de données A et pour le jeu B. La tendance pressentie lors de l'analyse de la table 7 est confirmée puisque on voit clairement que la courbe correspondant à la configuration 12 qui correspond à notre procédure est nettement en-dessous de la courbe correspondant à l'optimisation naïve (configuration 1). En cela, on peut dire que l'algorithme que nous avons proposé permet d'améliorer l'exploration de l'espace des conformations d'une protéine.

Les figures 30 et 31 illustrent les évolutions des QScore et des scores AL0 évalués sur les ensembles d'apprentissage (A) et l'ensemble de test (B). Ainsi, malgré l'amélioration perceptible qui a été obtenue en termes d'énergie, le même constat ne peut s'appliquer aux mesures de similarité de structure. En effet, en termes de QScore par exemple, notre procédure est meilleure sur l'ensemble de test, mais moins performante sur l'ensemble d'apprentissage. Le score AL0 confirme ce constat mitigé puisque les performances de notre procédure sont moins bonnes que l'optimisation naïve sur l'ensemble d'apprentissage et comparables sur le jeu de

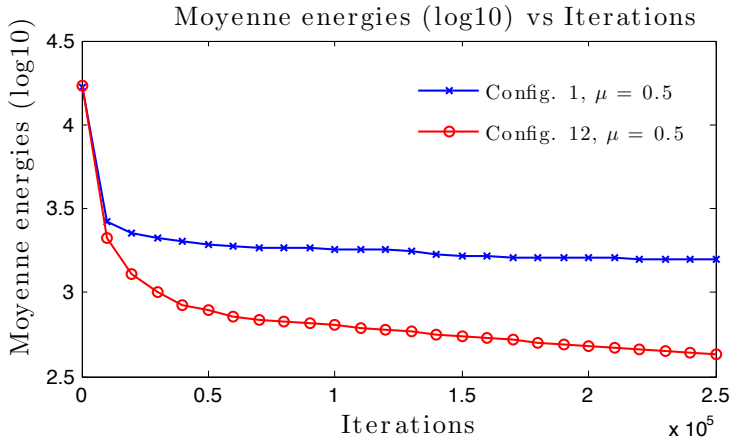


FIGURE 28 – Moyenne des énergies des structures en cours d'optimisation à différents instants de l'optimisation pour les protéines issues du jeu de données d'apprentissage A.

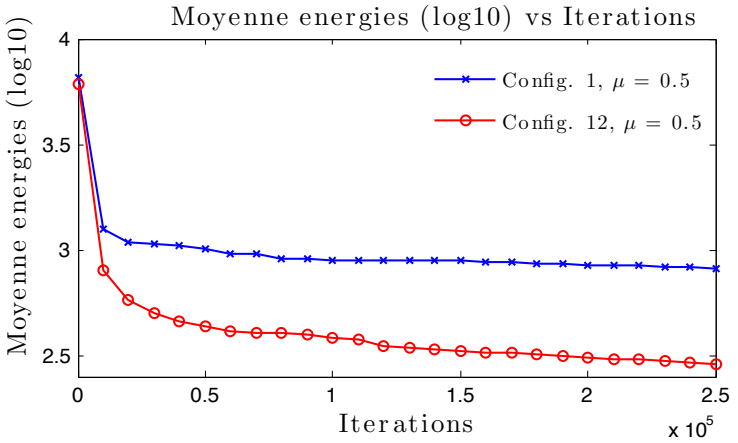
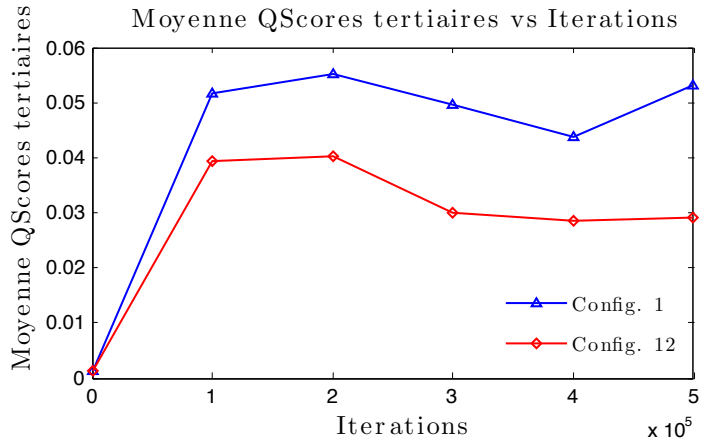


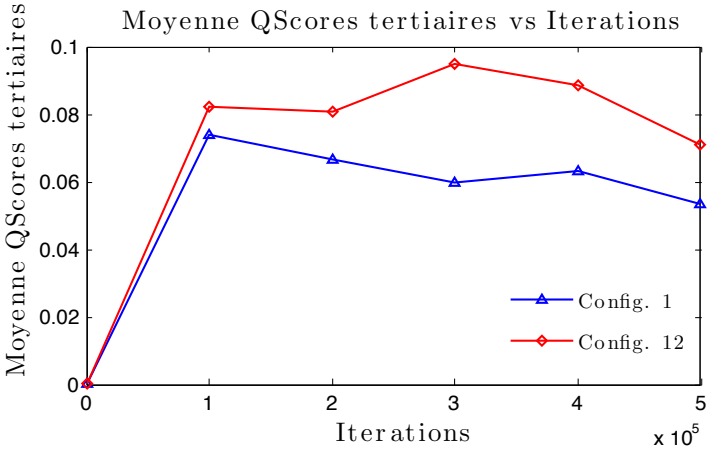
FIGURE 29 – Moyenne des énergies des structures en cours d'optimisation à différents instants de l'optimisation pour les protéines issues du jeu de données de test B.

données de test. Quoiqu'il en soit, rappelons que nous sommes en tout début d'optimisation et que les faibles valeurs de ces scores ne permettent pas d'établir des conclusions définitives.

A titre illustratif et pour mettre en avant le fait déjà évoqué que les structures obtenues via l'algorithme du recuit simulé sont très éloignées des structures réelles, la figure 32 représente la structure réelle de la protéine 1B5M (cyan), la structure obtenue après application de l'optimiseur naïf (mauve) et celle obtenue après application de notre méthode (Configuration 12)(jaune). Les résultats provenant de nos optimisations ont été récoltés après 250000 itérations. Les différences entre notre méthode et l'optimiseur naïf ne sont pas perceptibles à l'oeil nu, mais cette figure met en évidence l'écart colossal qui existe entre notre prédiction et la structure réelle.

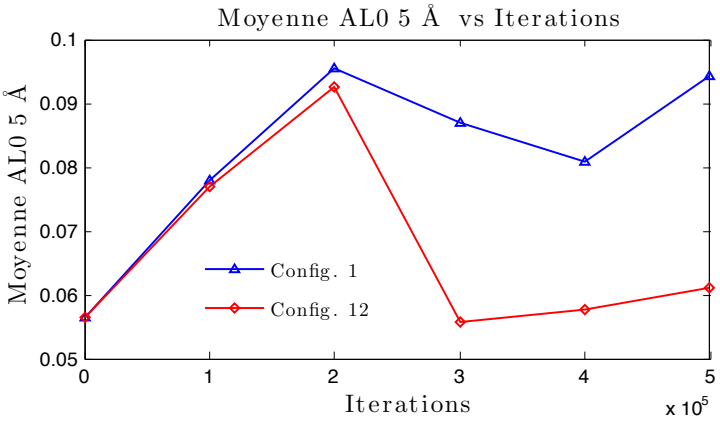


(a) Ensemble d'apprentissage A

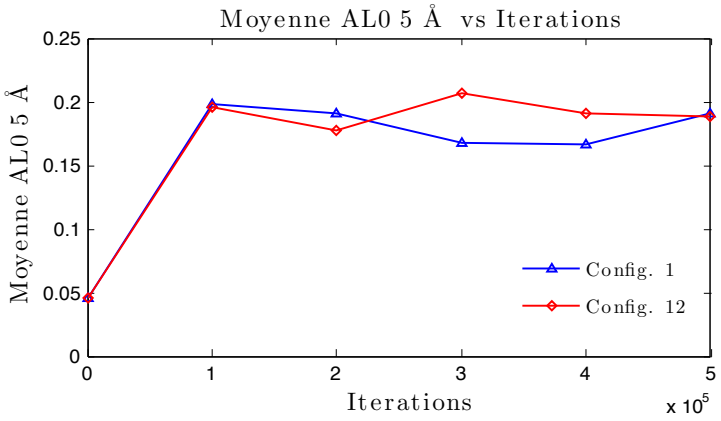


(b) Ensemble de test B

FIGURE 30 – Moyenne des QScores évaluant la qualité des structures tertiaires. Scores évalués sur l'ensemble d'apprentissage (A) et l'ensemble de test (B). L'axe des abscisses représente les différents instants auxquels les mesures ont été prises. Les valeurs de cet axe doivent être multipliées par 0.5 afin d'obtenir les valeurs correspondantes en nombre d'itérations.



(a) Ensemble d'apprentissage A



(b) Ensemble de test B

FIGURE 31 – Moyenne des scores AL0 pour une distance de cutoff de 5Å. Scores évalués sur l'ensemble d'apprentissage (A) et l'ensemble de test (B). L'axe des abscisses représente les différents instants auxquels les mesures ont été prises. Les valeurs de cet axe doivent être multipliées par 0.5 afin d'obtenir les valeurs correspondantes en nombre d'itérations.

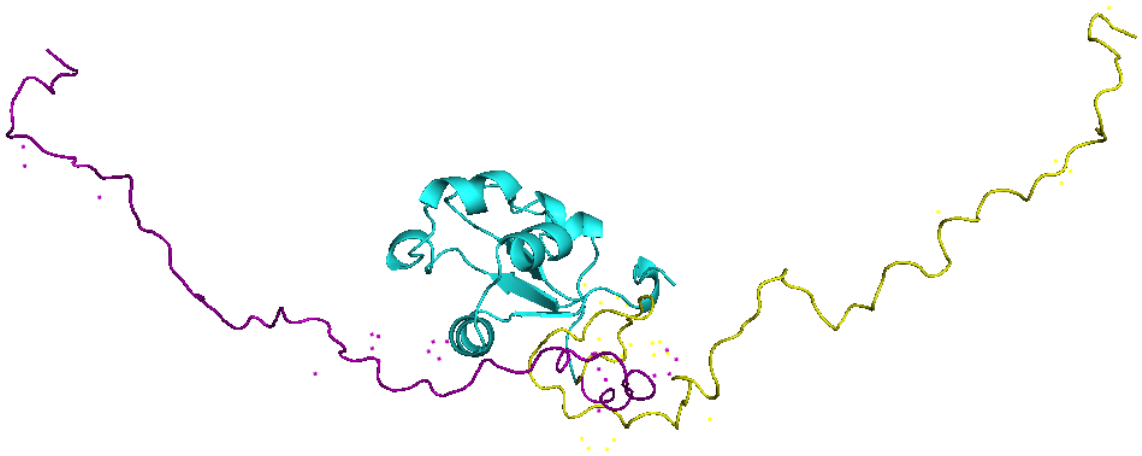


FIGURE 32 – Comparaison de structures. Structure réelle de la protéine 1B5M (cyan), structure obtenue par optimisation via un algorithme naïf (mauve) et structure obtenue via notre méthode (jaune).

7.4 Remarques et observations

Bien que l'optimisation n'en soit qu'à ses premières étapes, les résultats que nous avons obtenus sont très encourageants. En effet, on remarque clairement que, dans certains cas, notre méthode d'optimisation est beaucoup plus performante que la méthode naïve à laquelle nous nous comparons. Cependant, ces résultas ont été obtenus en utilisant une procédure pour laquelle certains choix ont dû être réalisés arbitrairement et *a priori* et ce pour diverses raisons (notamment les temps de calcul). La qualité de ces choix doit donc être discutée. Ainsi, ces constatations limitent les conclusions finales qui peuvent être tirées de nos résultats. En effet, de meilleurs choix peuvent vraisemblablement améliorer les résultats que nous avons obtenus jusqu'ici.

8 Problèmes rencontrés

Un grand nombre de problèmes rencontrés lors de la réalisation de ce travail proviennent de l'interfaçage entre deux programmes. En effet, nous avons deux suites logicielles qui doivent communiquer entre elles : d'une part LBCpp qui nous fournit de nombreuses fonctionnalités pré-implémentées (apprentissage, génération de graphes, génération de fichiers log, etc.) et d'autre part Rosetta qui fournit des éléments indispensables tels que la fonction d'énergie et l'implémentation des movers.

Rosetta est un ensemble de programmes qui remplit une multitude de fonctions très avancées. On le conçoit aisément, le code source d'un tel programme est abondant et complexe. Hélas, très peu de documentation est disponible quant à l'utilisation des différentes méthodes, fonctions et classes que Rosetta offre. La plupart du temps, nous avons dû parcourir le code de Rosetta afin de comprendre comment utiliser ses fonctionnalités. Heureusement, les éléments de Rosetta dont nous avons besoin ici sont des éléments standards utilisés massivement. Ainsi, nous avons pu disposer d'une quantité suffisante d'exemples. Néanmoins, le manque de documentation est un facteur qui a fortement ralenti le développement de ce travail, tout du moins à ses prémisses.

Les structures utilisées pour représenter les protéines sont différentes au sein de LBCpp et de Rosetta. Cependant, comme ces deux logiciels doivent communiquer entre eux, il importe de pouvoir effectuer la conversion de l'un vers l'autre. Nous avons essayé, dans un premier temps, d'effectuer la conversion directement, en créant un objet [22] vide dans la suite vers laquelle la conversion devait être effectuée et en le remplissant manuellement, à partir des données de l'objet représentant la protéine dans la suite de départ. Hélas, cette méthode s'est révélée infructueuse vu le manque de documentation précédemment évoqué. Nous avons donc dû opter pour une autre méthode. Cette deuxième approche consiste à générer, à partir de la suite logicielle de départ, un objet [23] contenant une représentation de la protéine sous format PDB et à le donner à la suite d'arrivée afin que celle-ci génère automatiquement un objet correct. Ce procédé présente essentiellement deux désavantages. Tout d'abord, la conversion est beaucoup plus lente, et, ensuite, celle-ci est beaucoup moins précise. En effet, le format PDB ne représente les coordonnées d'un atome dans l'espace que par des nombres présentant 3 décimales au maximum. Evidemment, les movers appliqués à la protéine modifient les positions de manière importante et ces positions finales, résultant de l'application du mover, sont définies avec beaucoup plus que 3 décimales (format double en C++). Lors du passage par le format PDB, la protéine perd une grande résolution et cet effet a été ressenti lors de l'analyse de nos résultats. Ce problème de conversion a ainsi mis en évidence l'extrême sensibilité de la fonction d'énergie. Ce problème a ainsi grandement compliqué l'analyse de nos résultats.

En sus des pertes de résolution que la conversion d'une suite logicielle vers l'autre impose, Rosetta exhibe un comportement étrange lors du chargement d'une protéine en son sein. Ainsi, Rosetta a tendance à ajouter, ou à retirer, un certain nombre d'atomes. Ce comportement, bien que non désiré, peut être compréhensible. En effet, Rosetta calcule, par exemple, l'énergie

22. représentant une protéine.
23. de type chaîne de caractères

du solvant au moyen de potentiels statistiques. Il est possible que certains termes de la fonction d'énergie de Rosetta nécessitent que la protéine soit constituée des atomes standards de chaque acide aminé expliquant, par là même, les changements opérés. Ces ajouts ou retraits touchent principalement [24] les chaînes secondaires et les atomes d'hydrogène. Heureusement, la procédure d'optimisation développée dans ce travail ne nécessite aucune comparaison entre les chaînes secondaires, c'est pourquoi ce phénomène est d'importance minime. Cependant, une autre modification introduite par Rosetta au chargement des protéines s'avère bien plus sérieuse. Certaines parties de la séquence de la protéine qui est chargée au sein de Rosetta sont purement et simplement supprimées. A ce jour, nous ne comprenons toujours pas les raisons de ce comportement. Quoiqu'il en soit, ses effets sont particulièrement radicaux : les protéines atteintes ne peuvent tout simplement pas être utilisées dans notre approche puisque celle-ci nécessite la comparaison de la structure contenue dans PDB et de la structure obtenue après application de nos opérateurs. Ce phénomène est une des causes ayant entraîné la diminution de la taille de la population des protéines étudiées.

L'avant dernier problème que nous nous proposons de mettre en avant est lié à la fonction d'énergie. Celle-ci est, dans certaines circonstances, incapable de calculer l'énergie de la structure et termine brutalement le programme sans autre forme de procès. La fonction d'énergie bogue après que le processus d'optimisation ait appliqué un certain nombre de movers à la structure. Une fois encore, nous n'avons pu déterminer l'origine de ce problème qui est vraisemblablement interne à Rosetta puisque toutes les opérations de modification ainsi que le calcul de l'énergie sont des opérations de cette même suite logicielle. Ce facteur constitue la deuxième raison de la diminution de la taille de notre base de données initiale. Ce problème ne trouvant pas de réponse, et dans le souci d'éviter que cela ne se reproduise au plus mauvais moment, toutes les protéines qui ont souffert de ce problème ont été retirées de notre base de données.

Pour finir, nous rappelons certains éléments qui ont déjà été mentionnés au début de ce document : les temps de calcul très élevés et les incohérences de la fonction d'énergie lors de la division d'une protéine en plusieurs morceaux. Le premier problème aurait pu être contourné si la parallélisation avait pu être mise en oeuvre. Hélas, ce ne fut pas le cas et nous avons préféré nous consacrer à la réalisation de notre méthode plutôt qu'à la résolution de bogues. Ainsi, les temps de calcul non-négligeables ont rendu impossible l'évaluation de notre méthode à grande échelle. Le deuxième problème a, quant à lui, renforcé le premier, puisque sa correction entraîne des calculs supplémentaires dont le temps d'exécution vient s'ajouter aux autres tâches déjà bien lentes.

24. Aucun ajout ou retrait n'a été constaté dans les atomes de la chaîne principale, mais une règle ne peut être tirée d'une observation incomplète.

9 Futurs travaux

Un travail imposant est encore nécessaire avant que notre méthode ne soit réellement utile sur des problèmes pratiques. Cette section détaille quelques pistes qui mériteraient d'être investiguées.

Remarquons tout d'abord que l'algorithme d'optimisation utilisé est un algorithme relativement élémentaire. Celui-ci mérite donc une étude approfondie afin de déterminer si il permet, moyennant, un bon choix de paramètres d'obtenir la solution correcte. Auquel cas, la détermination de ses paramètres optimaux est une condition *sine qua non* pour qu'une optimisation efficace puisse être réalisée. Si le recuit simulé s'avère inefficace, un autre type d'algorithme devra être considéré, comme, par exemple, des algorithmes de recherche dans un arbre tel le beam-search. L'utilisation d'un bon algorithme d'optimisation est cruciale puisque celui-ci intervient à deux endroits dans notre procédure : au moment de la génération de la base de données des structures intermédiaires et au moment de l'optimisation sur de nouvelles protéines. Finalement, on peut également pointer du doigt les structures initiales utilisées lors de la procédure d'optimisation. En effet, une initialisation plus intelligente (autre que l'hélice utilisée ici) permettrait certainement de s'affranchir d'un grand nombre de minima locaux et d'améliorer les performances de la méthode.

Notre méthode s'appuie sur l'apprentissage, à partir d'un ensemble de structures, des meilleures actions à réaliser. Le choix des structures intermédiaires est donc primordial et conditionne la qualité du modèle génératif créé. Lors de ce travail, les structures utilisées ont été obtenues via une optimisation naïve avec un grand nombre d'itérations en utilisant un seul type de mover. Une première amélioration consisterait à générer les structures intermédiaires avec un grand nombre d'itérations et en utilisant tous les types de movers. Une deuxième piste consiste à partir des structures cibles connues et à déplier celles-ci afin d'obtenir des structures partiellement repliées qui sont plus proches de la structure cible. Finalement, la méthode développée pourrait être appliquée afin d'optimiser ces mêmes structures utilisées lors de l'apprentissage, les structures intermédiaires obtenues seront vraisemblablement meilleures et donc leur apprentissage apporterait plus d'informations. Ce procédé est appelé bootstrapping. Une autre solution consiste à appliquer la méthode sur des protéines de plus en plus grandes, en apprenant à chaque fois une nouvelle politique de décision, afin d'améliorer *in fine* notre procédure. Ce ne sont bien évidemment que des pistes qui sont présentées et leur pertinence ainsi que leur efficacité doivent encore être vérifiées.

La procédure permettant de détecter les meilleurs movers peut encore être améliorée. En effet, bien qu'un ensemble de bons movers ait pu être généré au moyen de la méthode que nous avons mise au point, rien ne garantit que cet ensemble soit optimal. L'utilisation des distributions avec taux d'apprentissage au sein de l'EDA n'est pas vraiment un atout lors de la génération des movers puisque celles-ci limitent la convergence et, peut-être, la qualité des movers générés. Il serait probablement plus intéressant d'utiliser les distributions sans taux d'apprentissage qui convergent mieux et de relancer plusieurs fois l'EDA afin d'obtenir de très bons movers et une bonne diversité parmi ceux-ci. Les distributions utilisées gagneraient très probablement à être complexifiées afin de ne pas converger aussi vite, et de, peut-être, permettre la découverte de nouveaux movers encore meilleurs. Dans tous les cas, un nombre

très grand de movers devrait être évalué à chaque itération (en tout cas lors des premières étapes). En effet, un rapide calcul permet de se rendre compte de la quantité de movers qu'il est possible de générer à partir des trois types de movers considérés pour une protéine de 100 acides aminés. Si on fait l'hypothèse que les trois movers possèdent 3 paramètres [25] (1 résidu et 2 paramètres continus), que les paramètres continus sont des angles qui varient de 0 à 360° et que nous discrétisons ces angles par pas de 5°, alors le nombre de movers qui peut être généré pour une structure est de

$$3 \times 100 \times 72 \times 72 = 1\ 555\ 200.$$

On voit qu'il faudrait générer plus de 10^6 movers pour espérer couvrir l'ensemble des movers possibles et donc pouvoir ainsi détecter les meilleurs. Ceci semble évidemment infaisable, mais nous nous apercevons rapidement que les faibles valeurs que nous avons utilisées pourraient être augmentées afin d'obtenir de meilleurs résultats. Cela indique également que les paramètres de l'EDA doivent être ajustés afin d'améliorer la qualité des movers trouvés.

Les features utilisées dans ce document pour décrire la protéine sont fort sommaires. La description d'une structure gagnera certainement en précision si des features plus évoluées sont implémentées. Il reste à définir l'impact réel de cette description sur la performance de l'optimisation et à définir le niveau de description qui est nécessaire. De toute évidence, une description exacte de la protéine est inconcevable puisque le bénéfice du travail accompli serait alors perdu. Cependant, des features trop générales ne permettent pas non plus de décrire la protéine correctement. Le juste milieu doit donc être déterminé afin que la politique d'exploration puisse effectuer les choix les plus pertinents.

Les méthodes d'apprentissage sont certainement perfectibles. En effet, la fonction utilisée afin d'apprendre la politique de choix des movers doit être juste assez complexe pour ne pas sur-apprendre le jeu de données tout en ne le sous-apprenant pas trop. Trouver la fonction d'apprentissage permettant d'atteindre le juste milieu sera un des points clés de la réussite, à grand échelle, de notre méthode.

Etant donné les résultats obtenus, il semble important de considérer un autre critère d'acceptation de structure candidate au sein de l'algorithme d'optimisation. Un critère de structure pourrait avantageusement remplacer le critère actuel en évitant les minima locaux de la fonction d'énergie. Cependant, à notre connaissance, un tel critère n'existe pas encore et n'est certainement pas trivial à développer.

25. Ceci est une sous-estimation puisque les RigidBodyMovers en possèdent en fait quatre.

10 Conclusion

La méthode *ab initio* proposée montre des résultats très encourageants bien que ceux-ci mettent aussi en évidence que seules les premières étapes de l'optimisation ont été rencontrées. Ce travail, dont le but était de prouver que l'apprentissage automatique, sur base de solutions connues du problème, permet d'améliorer l'optimisation de la structure, a donc tenu ses promesses et les perspectives qu'il met en lumière semblent assez prometteuses. Ainsi, il a été démontré expérimentalement que, dans certains cas, notre méthode améliore l'optimisation simple réalisée au moyen du recuit simulé de manière non négligeable. Néanmoins, avant que cette procédure ne soit utile sur des problèmes réels, de nombreux progrès sont nécessaires aussi bien au niveau de l'algorithme d'optimisation qu'au niveau des procédés d'apprentissage. Beaucoup de travail reste donc à faire dans ces domaines, preuve en est les quelques suggestions déjà proposées dans la section 9. Cette liste n'est pas exhaustive et bien d'autres aspects requièrent une attention particulière en vue d'améliorer au mieux la méthode proposée.

Bien que ce travail ait été réalisé sur des protéines, son envergure s'étend bien au-delà de la prédiction de structures de protéines. Il existe de très nombreux problèmes de la littérature scientifique qui sont, à ce jour, encore très mal résolus et pour lesquels aucun algorithme d'approximation n'existe. De tels problèmes pourraient grandement bénéficier d'une approche *learning for search* telle que celle mise en avant dans ce travail. Parmi les problèmes qui seraient susceptibles de bénéficier du *learning for search*, nous pouvons citer le problème de gestion de la production ou encore les problèmes de stabilité des réseaux électriques. De plus, la méthodologie mise en avant dans ce document est très générale puisqu'elle peut s'adapter à un grand nombre de problèmes, ainsi qu'à un grand nombre d'algorithmes d'optimisation.

Les résultats obtenus laissent penser que notre procédure pourrait être utilisée en complément des prédicteurs existant déjà dans le domaine en vue d'améliorer leurs performances. Nous espérons, par ce travail, avoir pu apporter une pierre à l'édifice et, avoir contribué à la résolution d'un problème aussi crucial que celui de la prédiction de la structure tertiaire de protéines.

A Acides aminés

Cette annexe présente les compositions chimiques des 20 acides aminés. Les acides aminés appartenant à la famille des acides aminés acides sont représentés à la figure 33, les acides aminés basiques à la figure 34 et les acides aminés polaires non chargés à la figure 35. Finalement, la dernière catégorie, à savoir les acides aminés non polaires, sont illustrés sur la figure 36.

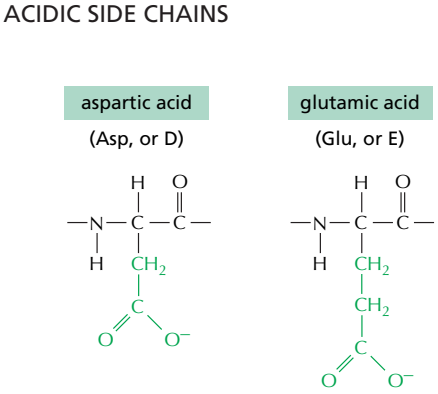


FIGURE 33 – Acides aminés polaires et chargés négativement (acides) [1]

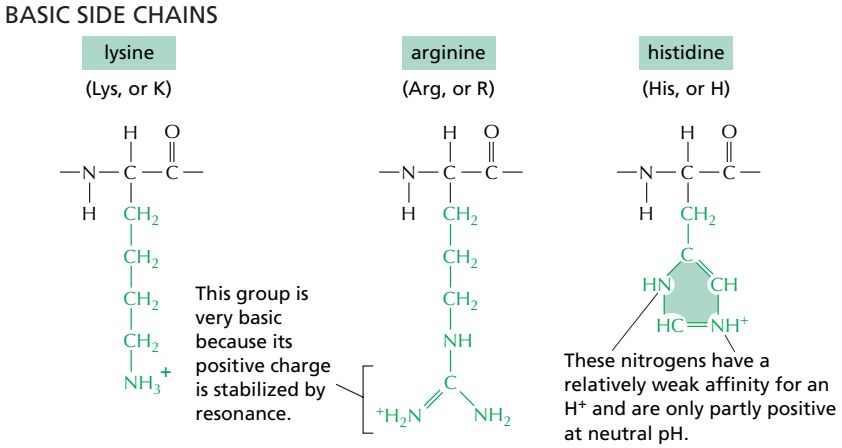


FIGURE 34 – Acides aminés polaires et chargés positivement (basiques) [1]

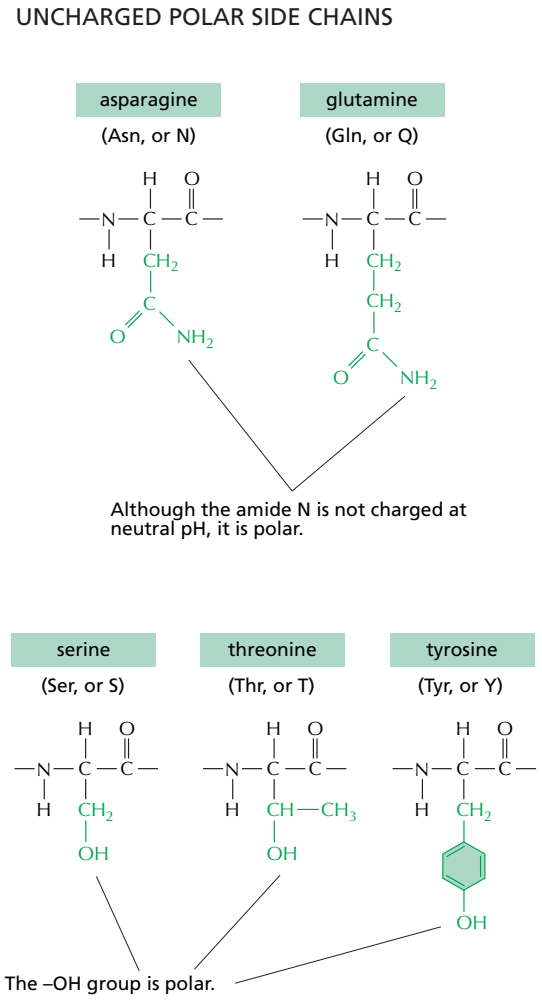


FIGURE 35 – Acides aminés polaires et non chargés [1]

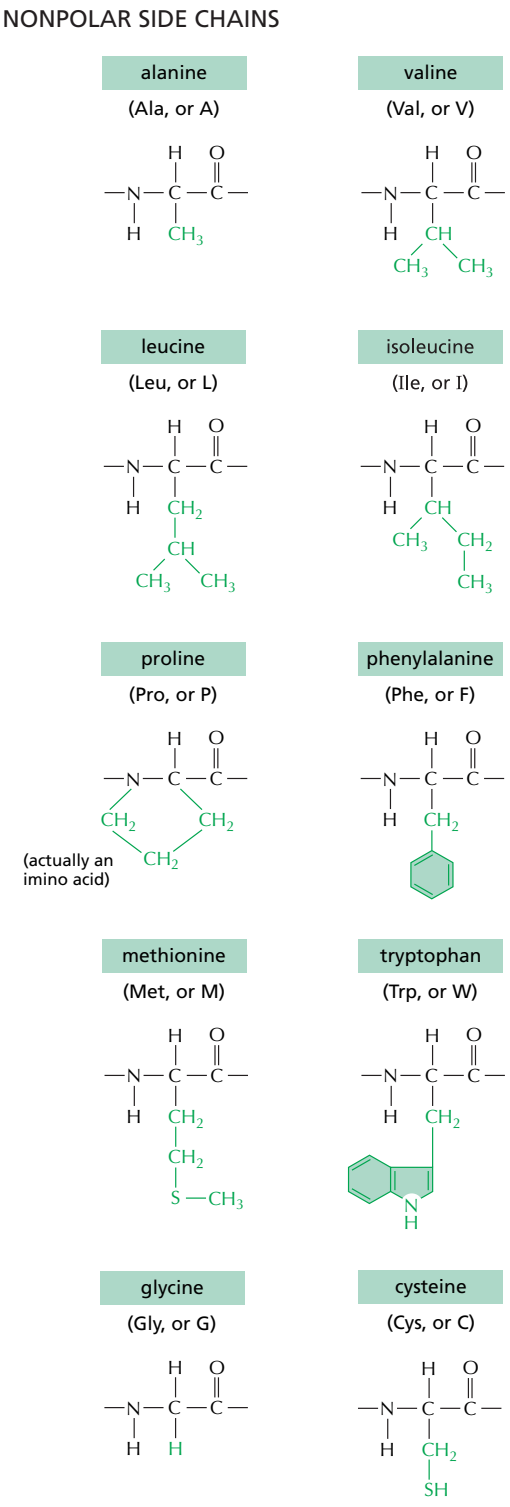


FIGURE 36 – Acides aminés non-polaires [1]

B MATT : un algorithme d'alignement de multiples séquences

L'algorithme MATT ayant été choisi à défaut d'un autre facilement utilisable, il nous a semblé important d'établir la fiabilité et la qualité de l'algorithme. Pour cela, nous avons commencé par poser un regard critique sur l'article décrivant la méthode [63]. Un point intéressant est qu'il s'agit d'une des toutes dernières méthodes proposées dans ce domaine car elle date de 2008. Elle bénéficie donc des dernières avancées en la matière et peut se comparer aux algorithmes d'alignement qui faisaient office, jusqu'à présent, de référence.

MATT a été testé sur les bases de données Homstrad et SABmark. Homstrad [67] est une base de données très populaire pour évaluer les algorithmes d'alignement. A la date à laquelle [63] fut publié, Homstrad possédait 1028 alignements différents qui contiennent chacun entre 2 et 41 structures alignables. Homstrad propose des structures relativement semblables et les alignements des structures sont vérifiés manuellement afin de garantir une qualité maximale. SABmark [85] est, quant à elle, une base de données regroupant deux catégories de structures alignables qui présentent des niveaux de similarité relativement faibles : la Twilight Zone et la catégorie des SuperFamilies. Au moment de la parution de [63], la Twilight Zone contenait 1740 séquences, dont la similarité va de très faible à faible, triées en 209 sous-ensembles. La catégorie des SuperFamilies contenait, quant à elle, 3645 structures dont le niveau de similarité varie de faible à intermédiaire qui ont été classifiées en 426 sous-catégories. Chaque sous-ensemble de structures contient entre 3 et 25 éléments. La classification de SABmark est effectuée au moyen de divers algorithmes et les résultats obtenus ne sont pas évalués manuellement, contrairement à Homstrad.

MATT a été lancé par ses créateurs sur Homstrad et sur SABmark et comparé aux algorithmes MultiProt [81], MUSTANG [49] et POSA [89]. Les résultats obtenus par rapport à POSA sont particulièrement intéressants puisque cette méthode est très similaire à MATT. Les critères choisis pour l'évaluation sont le nombre de résidus qui ont été déterminés comme faisant partie du coeur commun aux alignements et la mesure RMSD entre les positions des résidus faisant partie de ce coeur. Ce critère est un double critère puisqu'il s'agit de minimiser le RMSD tout en maximisant le nombre de résidus présents dans le coeur commun des protéines. Il apparaît que sur Homstrad, MATT est comparable en termes d'alignement aux autres algorithmes. Néanmoins, pour un même nombre de résidus dans le coeur, MATT présente un RMSD inférieur aux autres méthodes. Sur SABmark, l'évaluation est un peu différente puisque celle-ci inclut aussi une évaluation en termes courbes ROC. MATT est en effet également utilisé pour déterminer si deux structures sont oui ou non alignables. Ces résultats permettent d'évaluer le nombre de vrais et de faux positifs et peuvent ensuite être reportés sous forme de courbe ROC. MATT obtient aussi ici de meilleurs résultats que les autres algorithmes. Les détails sont présentés dans [63]. Une propriété remarquable de MATT est son aptitude à mieux détecter la fin des hélices α et des feuillets β. Il est intéressant de remarquer que, bien que les idées fondatrices de POSA et MATT soient semblables, ce dernier obtient de meilleurs résultats que le premier. Les bons résultats de MATT, aussi bien sur Homstrad que sur SABmark, seraient dus à la capacité qu'a MATT de modéliser les protéines dans différentes conformations ainsi que de modéliser des distorsions de la backbone de la protéine.

Pour pousser l'étude un peu plus loin, nous avons décidé de comparer directement, et avec les moyens dont nous disposions au moment du test, différents algorithmes d'alignement.

	MC vs. LO10e5	MC vs. LO10e6
MATT	7,766	6,142
SCALI	16,795	13,725
TM-align	8,755	7,441
`super` (PyMol)	7,981	13,126
`align` (PyMol)	4,831	4,297

TABLE 8 – Valeurs RMSD pour les différents algorithmes utilisés et les deux alignements réalisés.

Nous avons choisi de comparer MATT aux algorithmes SCALI, TM-align et aux algorithmes d'alignement inclus dans la suite PyMol [26] car les deux premiers sont disponibles sous la forme de serveurs web et le troisième est directement inclus dans l'outil de visualisation et de mesure que nous utilisons, à savoir PyMol. La plupart des méthodes sont capables d'aligner deux ou plusieurs séquences différentes, or nous portons notre intérêt uniquement sur l'alignement de structures de séquences identiques. Nous avons donc décidé de comparer les méthodes choisies en alignant deux conformations différentes d'une même chaîne polypeptidique. Ces conformations ont été obtenues en utilisant les algorithmes basiques d'optimisation décrits dans la section 6.3. Nous avons créé une protéine imaginaire constituée de 12 alanines (A) dont la conformation initialement créée par Rosetta est un cercle, celle-ci est représentée à la figure 18. Nous lui avons ensuite appliqué les algorithmes d'optimisation locale avec 10^5 itérations (voir figure 19(a)) et avec 10^6 itérations (voir figure 19(b)) et de Monte-Carlo (voir figure 20). Ces conformations seront nommées par la suite LO10e5, LO10e6 et MC respectivement. On sait que les alanines, quand ils se succèdent dans la chaîne polypeptidique ont tendance à former une hélice α, on remarque sur la figure 20 que c'est effectivement la cas.

Ces conformations sont les conformations de base que nous avons tenté d'aligner au moyen des divers algorithmes. La cible était toujours MC puisque c'est celle qui se rapproche le plus de la réalité. Les deux autres conformations ont donc été alignées sur la première. Les résultats sont repris dans les figures 37 et 38, ainsi que dans la table 8.

On voit directement que SCALI donne des résultats significativement moins bons que les autres méthodes. Les résultats de TM-align sont quant à eux proches de ceux de MATT bien que moins bons. La méthode `super` de PyMol ne fournit pas de bons alignements tandis que la méthode `align` permet d'obtenir les déviations les plus basses du tableau. La comparaison des valeurs de la déviation RMS ne suffit pas pour établir une hiérarchie dans les méthodes d'alignement. En effet, en termes de RMSD, la méthode `align` est clairement la meilleure, mais on voit, sur la figure 38, que le résultat de l'alignement n'est pas parfait. L'alignement produit par MATT et TM-align semble en effet meilleur que celui produit par `align` bien que le RMSD pour cette dernière soit plus faible.

Au vu des résultats présentés dans [63] et des tests (très sommaires) effectués, nous déduisons que MATT est un algorithme d'alignement acceptable qui peut être utilisé avec une relative confiance.

26. `http://www.pymol.org/`

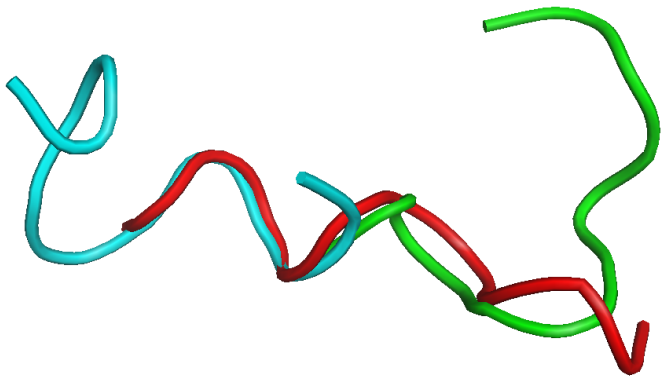


FIGURE 37 – Alignements obtenus par application de MATT (en vert) et de TM-align (en cyan).

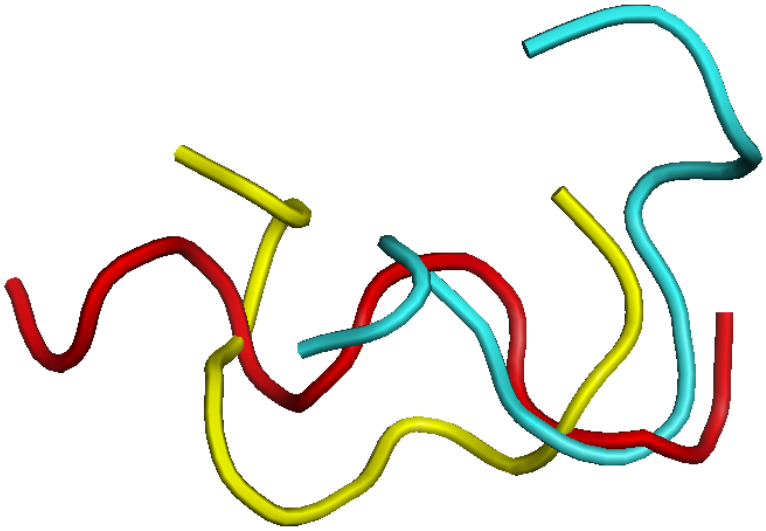


FIGURE 38 – Alignements obtenus par application des méthodes super (en cyan) et align (en jaune) de PyMol.

B.1 Détails algorithmiques à propos de MATT

MATT est l'acronyme de *Multiple Alignment with Translations and Twists*. C'est un algorithme de type chaînage de paires de fragments alignés (aligned fragment pair chaining). Au cours de l'exécution, MATT aligne au mieux des sous-ensembles de résidus indépendamment les uns des autres. Il autorise certains degrés de liberté entre deux fragments successifs de sorte que la structure intermédiaire obtenue n'est plus la structure réelle. Les translations et les rotations sont introduites entre les fragments afin de rapprocher ceux-ci les uns des autres. Ces transformations entre fragments sont autorisées même si elles sont impossibles physiquement. L'étape finale consiste à rétablir le caractère physique de la conformation [63].

Le raisonnement innovant proposé par MATT tient en la relaxation effectuée sur la structure intermédiaire que l'on désire aligner. Celle-ci peut ainsi, grâce aux translation et rotations autorisées entre fragments, se courber ou tourner sur elle-même pour s'aligner. Cette approche de l'alignement n'est apparue que très récemment [88, 80, 89]. La différence principale entre MATT et POSA est que MATT autorise une flexibilité partout sur la backbone, alors que POSA essaie de minimiser ces modifications au cours de son exécution.

Les programmes tels que MATT doivent généralement répondre à deux questions. La première concerne l'alignement des séquences. Le problème est alors de trouver le meilleur alignement possible entre des structures cibles qui possèdent toutes un coeur commun, c'est-à-dire un ensemble de résidus qui peut être aligné avec une faible déviation. Le deuxième problème, appelé problème de discrimination, consiste à répondre à la question *l'alignement trouvé est il oui ou non un bon alignement ?*. Il existe deux scores qui sont couramment utilisés pour mesurer la performance d'un alignement. Nous expliquerons d'abord comment le problème de l'alignement est traité par MATT, nous détaillerons ensuite le problème de discrimination.

MATT est un algorithme d'alignement de multiples séquences, or nous sommes intéressés uniquement par un alignement par paire. Nous détaillerons néanmoins l'exécution dans sa généralité. MATT prend en entrée un certain nombre de structures qui sont placées, lors de l'initialisation, dans autant d'ensembles séparés. La première phase de MATT est itérative et a pour but de regrouper, à chaque itération, les deux ensembles de structures qui sont les plus proches structurellement parlant, réduisant ainsi le nombre d'ensembles de structures de 1. Cette étape est réalisée jusqu'à ce qu'il ne reste plus qu'un seul ensemble. Cette phase itérative se décompose en trois étapes :

Paires de fragments : Cette étape aligne des fragments allant de 5 à 9 résidus entre les ensembles de structures. Un score est ensuite calculé pour chaque alignement.

Programmation dynamique : Les paires de fragments sont chaînées en utilisant un algorithme de programmation dynamique qui utilise comme score le score calculé au point précédent avec une pénalité due à la modification de la backbone que ce chaînage entraîne. Les deux ensembles possédant le plus grand score d'alignement sont ensuite regroupés.

Phase de réalignement et d'extension : Cette phase cherche des transformations locales qui minimisent le RMSD dans le groupe nouvellement formé. Ensuite, l'extension étend la chaîne formée tant que le RMSD est en dessous d'un certain seuil.

Cette étape itérative se termine lorsqu'il ne reste plus qu'un groupe. La phase finale consiste à rétablir la réalité géométrique en trouvant le coeur commun des structures et les transformations qui alignent ces résidus dans les protéines réelles.

Références

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular biology of the cell*. Garland Science, fifth edition, 2008.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Meyers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215 :403–410(8), 1990.

[3] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast : a new generation of protein database search programs. *Nucleic Acids Research*, 25(17) :3389–3402, 1997.

[4] C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096) :223–230, 1973.

[5] C. B. Anfinsen, E. Haber, M. Sela, and F. W. Jr. White. The kinetics of the formation of native ribonuclease during oxidation of the reduced polypeptide domain. *Proceedings of the National Academy of Sciences*, 47(9) :1309–1314, 1961.

[6] D. Baker and D. A. Agard. Kinetics versus thermodynamics in protein folding. *Biochemistry*, 33(24) :7505–7509, 1994.

[7] M. Ben-David, O. Noivirt-Brik, A. Paz, J. Prilusky, J. L. Sussman, and Y. Levy. Assessment of casp8 structure predictions for template free targets. *Proteins : Structure, Function, and Bioinformatics*, 77(S9) :50–65, 2009.

[8] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1) :39–71, 1996.

[9] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank : A computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, 112(3) :535 – 542, 1977.

[10] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 2002.

[11] P. E. Bourne and H. Weissig. *Structural bioinformatics*. Wiley, first edition, 2003.

[12] P. Bradley and D. Baker. Improved beta-protein structure prediction by multilevel optimization of nonlocal strand pairings and local backbone conformation. *Proteins : Structure, Function, and Bioinformatics*, 65(4) :922–929, 2006.

[13] S. H. Bryant and C. E. Lawrence. An empirical energy function for threading protein sequence through the folding motif. *Proteins : Structure, Function, and Bioinformatics*, 16(1) :92–112, 1993.

[14] A. A. Canutescu, A. A. Shelenkov, and R. L. Dunbrack. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Science*, 12(9) :2001–2014, 2003.

[15] B. Chazelle, C. Kingsford, and M. Singh. A semidefinite programming approach to side chain positioning with new rounding strategies. *INFORMS Journal on Computing*, pages 380–392, 2004.

[16] C. Chothia. One thousand families for the molecular biologist. *Nature*, 357 :543–544, 1992.

[17] C. Chothia and A. M. Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO Journal*, 5(4) :823–826, 1986.

[18] P. Y. Chou and G. D. Fasman. Conformational parameters for amino acids in helical, beta-sheet, and random coil regions calculated from proteins. *Biochemistry*, 13(2) :211–222, January 1974.

[19] D. G. Covell and R. L. Jernigan. Conformations of folded proteins in restricted spaces. *Biochemistry*, 29(13) :3287–3294, 1990.

[20] D. Cozzetto, A. Kryshtafovych, K. Fidelis, J. Moult, B. Rost, and A. Tramontano. Evaluation of template-based models in casp8 with standard measures. *Proteins : Structure, Function, and Bioinformatics*, 77(S9) :18–28, 2009.

[21] D. Cozzetto and A. Tramontano. Relationship between multiple sequence alignments and quality of protein comparative models. *Proteins : Structure, Function, and Bioinformatics*, 58(1) :151–157, 2005.

[22] D. Dehareng. *Mécanique et dynamique moléculaire*. Université de Liège, 2010.

[23] R. L. Dunbrack and M. Karplus. Conformational analysis of the backbone-dependent rotamer preferences of protein sidechains. *Nature Structural Biology*, 1(5) :334–340, 1994.

[24] M. P. Eastwood, C. Hardin, Z. Luthey-Schulten, and P. G. Wolynes. Evaluating protein structure-prediction schemes using energy landscape theory. *IBM J. Res. Dev.*, 45 :475–497, May 2001.

[25] S. R. Eddy. Hidden markov models. *Current Opinion in Structural Biology*, 6(3) :361 – 365, 1996.

[26] I. Eidhammer, I. Jonassen, and W. R. Taylor. *Protein Bioinformatics : An Algorithmic Approach to Sequence and Structure Analysis*. Wiley, 2003.

[27] D. Eisenberg, R. Lüthy, and J. U. Bowie. Verify3d : Assessment of protein models with three-dimensional profiles. In Robert M. Sweet Charles W. Carter Jr., editor, *Macromolecular Crystallography Part B*, volume 277 of *Methods in Enzymology*, pages 396 – 404. Academic Press, 1997.

[28] B. Fain and M. Levitt. A novel method for sampling alpha-helical protein backbones. *Journal of Molecular Biology*, 305(2) :191 – 201, 2001.

[29] T. P. Flores, C. A. Orengo, D. S. Moss, and J. M. Thornton. Comparison of conformational characteristics in structurally similar protein pairs. *Protein Science*, 2(11) :1811–1826, 1993.

[30] O. V. Galzitskaya, N. S. Bogatyreva, and D. N. Ivankov. Compactness determines protein folding type. *Journal of Bioinformatics and Computational Biology*, 6(4) :667–680, 2008.

[31] J. Garnier, D. J. Osguthorpe, and B. Robson. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *Journal of Molecular Biology*, 120(1) :97 – 120, 1978.

[32] N. Go and H. Taketomi. Respective roles of short- and long-range interactions in protein folding. *Proceedings of the National Academy of Sciences*, 75(2) :559–563, 1978.

[33] A. Godzik, A. Kolinski, and J. Skolnick. Topology fingerprint approach to the inverse protein folding problem. *Journal of Molecular Biology*, 227(1) :227 – 238, 1992.

[34] A. Godzik and J. Skolnick. Sequence-structure matching in globular proteins : application to supersecondary and tertiary structure determination. *Proceedings of the National Academy of Sciences*, 89(24) :12098–12102, 1992.

[35] R. A. Goldstein, Z. A. Luthey-Schulten, and P. G. Wolynes. Optimal protein-folding codes from spin-glass theory. *Proceedings of the National Academy of Sciences*, 89(11) :4918–4922, 1992.

[36] S. Guiasu and A. Shenitzer. The principle of maximum entropy. *The mathematical intelligencer*, 7(1) :42–48, 1985.

[37] M. R. Gupta and Y. Chen. Theory and use of the em algorithm. *Foundations and Trends in Signal Processing*, 4(3) :223–296, 2011.

[38] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233(1) :123 – 138, 1993.

[39] R. W. Hooft, G. Vriend, and C. Sander. Errors in protein structures. *Nature*, 381 :272, 1996.

[40] M. Jacobson and A. Sali. Comparative protein structure modeling and its applications to drug discovery. In A.M. Doherty, editor, *Annual Reports in Medicinal Chemistry*, volume 39, pages 259 – 276. Academic Press, 2004.

[41] L. Jaroszwski, L. Rychlewski, B. Zhang, and A. Godzik. Fold prediction by a hierarchy of sequence, threading, and modeling methods. *Protein Science*, 7(6) :1431–1440, 1998.

[42] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2) :195 – 202, 1999.

[43] D. T. Jones, W. R. Taylor, and J. M. Thornton. A new approach to fold recognition. *Nature*, 358 :86–89, 1992.

[44] W. Just. Computational complexity of multiple sequence alignment with sp-score. *Journal of Computational Biology*, 8(6) :615–623, 2001.

[45] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5) :922–923, Sep 1976.

[46] D. A. Keedy, C. J. Williams, J. J. Headd, W. B. Arendall, V. B. Chen, G. J. Kapral, R. A. Gillespie, J. N. Block, A. Zemla, D. C. Richardson, and J. S. Richardson. The other 90% of the protein : Assessment beyond the c-alpha for casp8 template-based and high-accuracy models. *Proteins : Structure, Function, and Bioinformatics*, 77(S9) :29–49, 2009.

[47] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, New Series*, 220(4598) :671–680, 1983.

[48] P. Koehl. Protein structure prediction. In *Biomedical Applications of Biophysics*, volume 3 of *Handbook of Modern Biophysics*, pages 1 – 34. Humana Press, 2010.

[49] A. S. Konagurthu, J. C. Whisstock, P. J. Stuckey, and A. M. Lesk. Mustang : A multiple structural alignment algorithm. *Proteins : Structure, Function, and Bioinformatics*, 64(3) :559–574, 2006.

[50] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, pages 99–124. Springer, October 2002.

[51] R. A. Laskowski, M. W. MacArthur, D. S. Moss, and J. M. Thornton. Procheck : a program to check the stereochemical quality of protein structures. *Journal of Applied Crystallography*, 26 :283–291, 1993.

[52] R. A. Laskowski, D.S. Moss, and J. M. Thornton. Main-chain bond lengths and bond angles in protein structures. *Journal of Molecular Biology*, 231(4) :1049–1067, 1993.

[53] R. H. Lathrop and T. F. Smith. Global optimum protein threading with gapped alignment and empirical pair score functions. *Journal of Molecular Biology*, 255(4) :641 – 665, 1996.

[54] K. F. Lau and K. A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10) :3986–3997, 1989.

[55] A. Leaver-Fay, M. Tyka, S. M. Lewis, O. F. Lange, J. Thompson, R. Jacak, K. W. Kaufman, P. D. Renfrew, C. A. Smith, W. Sheffler, I. W. Davis, S. Cooper, A. Treuille, D. J. Mandell, F. Richter, Y. E. A. Ban, S. J. Fleishman, J. E. Corn, D. E. Kim, S. Lyskov, M. Berrondo, S. Mentzer, Z. Popovic, J. J. Havranek, J. Karanicolas, R. Das, J. Meiler, T. Kortemme, J. J. Gray, B. Kuhlman, D. Baker, and P. Bradley. Rosetta3 : An object-oriented software suite for the simulation and design of macromolecules. In Michael L. Johnson and Ludwig Brand, editors, *Computer Methods, Part C*, volume 487 of *Methods in Enzymology*, pages 545 – 574. Academic Press, 2011.

[56] C. M.-R. Lemer, M. J. Rooman, and S. J. Wodak. Protein structure prediction by threading methods : Evaluation of current techniques. *Proteins : Structure, Function, and Bioinformatics*, 23(3) :337–355, 1995.

[57] A. M. Lesk, L. Lo Conte, and T. J. P. Hubbard. Assessment of novel fold targets in casp4 : Predictions of three-dimensional structures, secondary structures, and interresidue contacts. *Proteins : Structure, Function, and Bioinformatics*, 45(S5) :98–118, 2001.

[58] C. Levinthal. How to fold graciously. *Mossbauer Spectroscopy in Biological Systems : Proceedings of a meeting held at Allerton House*, pages 22–24, 1969.

[59] H. Liu, M. Elstner, E. Kaxiras, T. Frauenheim, J. Hermans, and W. Yang. Quantum mechanics simulation of protein dynamics on long timescale. *Proteins : Structure, Function, and Bioinformatics*, 44(4) :484–489, 2001.

[60] F. Maes, J. Becker, and L Wehenkel. Prédiction structurée multitâche itérative de propriétés structurelles de protéines. *7e Plateforme AFIA*, 2011.

[61] M. A. Marti-Renom, A. C. Stuart, A. Fiser, R. Sanchez, F. Melo, and A. Sali. Comparative protein structure modeling of genes and genomes. *Annual Review of Biophysics and Biomolecular Structure*, 29(1) :291–325, 2000.

[62] H. Meirovitch. Recent developments in methodologies for calculating the entropy and free energy of biological systems by computer simulation. *Current Opinion in Structural Biology*, 17(2) :181 – 186, 2007.

[63] M. Menke, B. Berger, and L. Cowen. Matt : Local flexibility aids protein multiple structure alignment. *PLoS Comput Biol*, 4(1) :e10, 2008.

[64] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6) :1087–1092, 1953.

[65] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247) :335 – 341, 1949.

[66] S. Miyazawa and R. L. Jernigan. Estimation of effective interresidue contact energies from protein crystal structures : quasi-chemical approximation. *Macromolecules*, 18(3) :534–552, 1985.

[67] K. Mizuguchi, C. M. Deane, T. L. Blundell, and J. P. Overington. Homstrad : A database of protein structure alignments for homologous families. *Protein Science*, 7(11) :2469–2471, 1998.

[68] A. R. Ortiz, C. E. M. Strauss, and O. Olmea. Mammoth (matching molecular models obtained from theory) : An automated method for model comparison. *Protein Science*, 11(11) :2606–2621, 2002.

[69] B. H. Park and M. Levitt. The complexity and accuracy of discrete state models of protein structure. *Journal of Molecular Biology*, 249(2) :493 – 507, 1995.

[70] W. R. Pearson. Rapid and sensitive sequence comparison with fastp and fasta. In *Methods in Enzymology*, volume 183, pages 63 – 98. Academic Press, 1990.

[71] D. Petrey and B. Honig. Protein structure prediction : Inroads to biology. *Molecular Cell*, 22(6) :811–819, 2005.

[72] N. A. Pierce and E. Winfree. Protein design is np-hard. *Protein Engineering*, 15(10) :779–782, 2002.

[73] J. W. Ponder and F. M. Richards. Tertiary templates for proteins : Use of packing criteria in the enumeration of allowed sequences for different structural classes. *Journal of Molecular Biology*, 193(4) :775 – 791, 1987.

[74] C. A. Rohl, C. E. M. Strauss, K. M. S. Misura, and D. Baker. Protein structure prediction using rosetta. In Ludwig Brand and Michael L. Johnson, editors, *Numerical Computer Methods, Part D*, volume 383 of *Methods in Enzymology*, pages 66 – 93. Academic Press, 2004.

[75] B. Rost. Twilight zone of protein sequence alignments. *Protein Engineering*, 12(2) :85–94, 1999.

[76] R. B. Russell, M. A. S. Saqi, R. A. Sayle, P. A. Bates, and M. J. E. Sternberg. Recognition of analogous and homologous protein folds : analysis of sequence and structure conservation. *Journal of Molecular Biology*, 269(3) :423 – 439, 1997.

[77] A. Sali and T. L. Blundell. Comparative protein modelling by satisfaction of spatial restraints. *Journal of Molecular Biology*, 234(3) :779 – 815, 1993.

[78] R. Sanchez and A. Sali. Evaluation of comparative protein structure modeling by modeller-3. *Proteins : Structure, Function, and Bioinformatics*, 29(Issue Supplement 1), 1997.

[79] J. M. Sauder, J. W. Arthur, and R. L. Dunbrack Jr. Large-scale comparison of protein sequence alignment algorithms with structure alignments. *Proteins : Structure, Function, and Bioinformatics*, 40(1) :6–22, 2000.

[80] M. Shatsky, R. Nussinov, and H. J. Wolfson. Flexible protein alignment and hinge detection. *Proteins : Structure, Function, and Bioinformatics*, 48(2) :242–256, 2002.

[81] M. Shatsky, R. Nussinov, and H. J. Wolfson. A method for simultaneous alignment of multiple protein structures. *Proteins : Structure, Function, and Bioinformatics*, 56(1) :143–156, 2004.

[82] K. T. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *Journal of Molecular Biology*, 268(1) :209 – 225, 1997.

[83] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal w : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22) :4673–4680, 1994.

[84] A. Tramontano, R. Leplae, and V. Morea. Analysis and assessment of comparative modeling predictions in casp4. *Proteins : Structure, Function, and Bioinformatics*, 45(S5) :22–38, 2001.

[85] I. Van Walle, I. Lasters, and L. Wyns. Sabmark, a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, 21(7) :1267–1268, 2005.

[86] M. Wiederstein and M. J. Sippl. Prosa-web : interactive web service for the recognition of errors in three-dimensional structures of proteins. *Nucleic Acids Research*, 35(suppl 2) :W407–W410, 2007.

[87] G. A. Wu, E. A. Coutsias, and K. A. Dill. Iterative assembly of helical proteins by optimal hydrophobic packing. *Structure*, 16(8) :1257–1266, 2008.

[88] Y. Ye and A. Godzik. Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinformatics*, 19(suppl 2) :ii246–ii255, 2003.

[89] Y. Ye and A. Godzik. Multiple flexible structure alignment using partial order graphs. *Bioinformatics*, 21(10) :2362–2369, 2005.

[90] X. Yuan and C. Bystroff. Non-sequential structure-based alignments reveal topology-independent core packing arrangements in proteins. *Bioinformatics*, 21 :1010–1019, April 2005.

[91] A. Zemla. Lga : a method for finding 3d similarities in protein structures. *Nucleic Acids Research*, 31(13) :3370–3374, 2003.

[92] A. Zemla, C. Venclovas, J. Moult, and K. Fidelis. Processing and evaluation of predictions in casp4. *Proteins : Structure, Function, and Bioinformatics*, 45(S5) :13–21, 2001.

[93] Y. Zhang and J. Skolnick. Tm-align : a protein structure alignment algorithm based on the tm-score. *Nucleic Acids Research*, 33(7) :2302–2309, 2005.