

à Alice et Elliott.

Table des matières

Liste des figures	vii
Liste des tables	xiii
Abréviations	xv
1 Le « CPD » : Conception de protéine par ordinateur	1
1.1 L'espace des séquences-conformations	2
1.1.1 L'état replié	2
Les chaînes latérales	3
Le squelette	3
1.1.2 L'état déplié	4
1.2 L'énergie d'une protéine	4
1.2.1 La mécanique moléculaire	4
Les interactions liées	5
Les interactions non liées	6
1.2.2 D'autres approches	7
1.3 La modélisation du solvant	7
1.3.1 Le modèle de solvant explicite	8
1.3.2 Le modèle implicite	8
1.3.3 Le modèle CASA	9
1.3.4 Le modèle Poisson-Boltzmann	10
1.3.5 Le modèle de Born Généralisé	11
« Native Environment Approximation » (NEA)	13
1.4 L'algorithme d'exploration	13
1.4.1 L'algorithme du champ moyen	14
1.4.2 Le Dead-End Elimination	15
1.4.3 Le CFN ou « Cost Function Network »	16
L'algorithme « Depth-First Branch and Bound » (DFBB)	17
Equivalence Preserving Transformation	18
1.4.4 L'heuristique Multistart Steepest Descent (MSD)	19
1.4.5 L'algorithme génétique	20
1.4.6 Le Monte Carlo	20

Le principe de la balance détaillée	22
L'algorithme de Metropolis	23
1.4.7 Le Monte Carlo avec échanges de répliques (REMC)	24
2 Proteus et nos outils d'analyse	27
2.1 Un modèle fondé sur la Physique	27
2.2 Organisation générale	28
2.3 Décomposition par paires de la fonction d'énergie	29
2.3.1 Décomposition du terme de surface	29
2.3.2 « Native Environnement Approximation » (NEA)	30
2.3.3 « Fluctuating Dielectric Boundary » (FDB)	30
2.4 La matrice d'énergie	31
2.4.1 Les énergies de l'état déplié	32
2.4.2 Déroulement de la construction de la matrice	32
2.4.3 Les fichiers d'énergies	34
2.5 Configuration de l'exploration	34
2.5.1 Les déplacements Monte Carlo	37
2.5.2 Restriction de l'espace séquence-conformation	37
2.5.3 Définition de la fonction de score	37
2.6 Les fichiers de sortie	38
2.7 Outils d'analyse de séquences	39
2.7.1 Superfamily/SCOP	39
2.7.2 Taux d'identité de séquences	40
2.7.3 Taux d'identité par position	40
2.7.4 Alignements Pfam	40
2.7.5 Score BLOSUM	41
2.7.6 Similarité d'un ensemble à un alignement Pfam	41
2.7.7 Entropie par position	41
Définition de groupes	47
Déclaration de la fonction de score	48
3 Développements algorithmiques	51
3.1 Les Modèles	51
3.2 OpenMP pour le REMC	52
3.2.1 Présentation d'OpenMP	52
3.2.2 REMC dans proteus	53
3.3 Amélioration de la fonction d'acceptation MC	54
3.4 Nouveau système de déplacements	56
3.5 Label	57
3.6 Allocation variable de la matrice	57

4 Comparing three stochastic search algorithms for CPD	59
4.1 Introduction	60
4.2 Methods	62
4.2.1 Monte Carlo : general framework	62
4.2.2 MC and REMC : implementation details	64
4.2.3 Heuristic sequence optimization	65
4.2.4 Cost function network method	65
4.2.5 Energy function	66
4.2.6 Test systems and preparation	67
4.2.7 Sequence characterization	67
4.3 Results	68
4.3.1 Quality of the designed sequences	68
4.3.2 Finding the GMEC	72
CPU and memory limits for each method	72
Optimal sequences/structures with up to 10 designed positions	74
Optimal sequences with 20 or 30 designed positions	75
4.3.3 Density of states above the GMEC	77
4.4 Concluding Discussion	79
5 Dessin computationnel de domaines PDZ	89
5.1 Introduction	89
5.2 Le modèle d'état déplié	90
5.2.1 Les énergies de référence	90
5.2.2 La vraisemblance des énergies de référence	90
5.2.3 Recherche du maximum de vraisemblance	92
5.3 Méthodes de calcul	93
5.3.1 Énergie de l'état replié	93
5.3.2 Les énergies de référence de l'état déplié	93
5.4 Séquences expérimentales et modèles structuraux	95
5.4.1 L'ensemble des protéines PDZ	95
5.4.2 Alignements Blast croisés	95
5.4.3 Sélection des homologues	95
5.4.4 Alignements des protéines expérimentales et leurs homologues	97
5.4.5 Similarité des homologues	97
5.4.6 Les fréquences d'acides aminés	99
5.5 Séquences calculées	99
5.5.1 Préparation	99
5.5.2 Simulation Monte Carlo	99
5.5.3 Génération de séquence Rosetta	100
5.5.4 Caractérisation des séquences calculées	100
5.6 Résultats du modèle NEA	100

Table des matières

5.6.1	Optimisation du modèle de l'état déplié	100
5.6.2	Tests de reconnaissance de famille	102
5.6.3	Séquences et diversité de séquences	103
5.6.4	Scores de similarité Blosum	104
5.6.5	Tests de validation croisée	104
5.7	Résultats du modèle FDB	107
5.7.1	Optimisation du modèle de l'état déplié	107
5.7.2	Tests de reconnaissance de famille	107
5.7.3	Scores de similarité Blosum	108
5.7.4	Taux d'identité à la séquence native	109
5.7.5	Logos des séquences obtenues	110
5.8	Application : Croissance du noyau hydrophobe	110
5.9	Conclusion	115
5.9.1	Modèle mis en œuvre	115
5.9.2	Tests et application	118
Publications		133
Bibliographie		135
Remerciements		149

Liste des figures

1.1	Le CPD pour « Computational Protein Design » recherche les séquences d'acides aminés compatibles avec une protéine dont la structure 3D est connue, c'est-à-dire compatibles avec un repliement et éventuellement une affinité pour un ligand.	2
1.2	Représentation des énergies liées De la gauche vers la droite et du haut vers le bas, sont représentées les énergies de liaison, d'angle, du dièdre et du dièdre impropre.	6
1.3	Les trois types classiques de surfaces pour une molécule À gauche et en rouge la surface de Van der Waals (SVdW), au centre et en bleu, la surface accessible au solvant (SA), à droite, et en vert la surface de Connolly ou surface moléculaire (SM). La surface SA est l'ensemble des positions pouvant être occupées par le centre d'une sphère (figurant une molécule du solvant) roulant sur SVdW. La surface SM est la plus petite enveloppe de SVdW dont chaque point peut être en contact avec la sphère.	10
1.4	Représentation schématique de l'organisation des dipôles des molécules d'eau autour d'un soluté sphérique chargé positivement à sa surface. Les dipôles des premières couches de solvatation s'orientent suivant les lignes du champ électrique produit par le soluté.	10
1.5	Des rayons de Born de deux atomes dans une protéine Le rayon de Born pour l'atome enfui est environ le rayon de la protéine, alors que le rayon de Born d'un atome à la surface est environ son rayon de Van der Waals.	12
1.6	DEE, graphes de la fonction d'énergie pour des rotamères fixés à la position i Le critère simple permet l'élimination du rotamère r_i^1 parce que l'énergie des conformations qui le contiennent, l'énergie rouge, est toujours moins bonne que la noire. Mais seul de critère de Goldstein permet d'éliminer r_i^2 parce que l'écart entre le graphe bleu et le graphe noir est positif pour chaque conformation.	16

1.7	Principe de l'algorithme du Depth-First Branch and Bound Les solutions d'un CFN sont représentées sous forme d'un arbre dans lequel le sommet S_0 représente l'ensemble des solutions et les fils S_1 et S_4 une partition de S_0 , avec pour chacun une première variable v_1 fixée. Le DFBB, descend jusqu'au sommet S_2 qu'il peut évaluer : un minorant de S_2 est 25. 25 devient le nouveau majorant du GMEC. L'algorithme remonte et redescend vers S_3 . Ici le coût constant plus le coût des affectations de v_1 et v_2 est supérieur au majorant : On peut élaguer l'arbre au sommet S_3 .	18
1.8	Exemple de transformations EPT sur un CFN composé de 2 variables v_i et v_j ayant deux valeurs possibles. La transformation de A vers B est une projection de v_j vers le coût constant C_0 . Puis une projection des fonctions binaires mène en C . La seconde valeur de v_j est distribuée sur les arcs en D . Cela permet deux nouvelles projections qui mènent à E puis à F . A et F sont en cohérence locale.	19
1.9	L'algorithme génétique	21
2.1	Cycle thermodynamique qui définit la stabilité relative de deux séquences-conformations S_A et S_B .	28
2.2	Une représentation de la surface accessible de trois résidus. Les résidus A et B réduisent mutuellement leur surface exposée au solvant : c'est la zone verte. De même pour B, C : la zone rouge et A, C, la zone bleue. Un calcul par paires de résidus naïf surestime la surface accessible en comptant deux fois la zone noire.	30
2.3	La matrice d'énergie. Cet exemple montre un polypeptide de 6 résidus, chaque position possède 2 types d'acides aminés possibles et 3 rotamères possibles (2 pour le type A et 1 pour le type B). La matrice organise toutes les interactions de paires de chaînes latérales possibles. Les interactions de la bande jaune de la matrice impliquent le résidu numéro 3, celles de la bande bleue impliquent le résidu numéro 4. Les points rouge et vert correspondent aux interactions notées respectivement par les flèches rouge et verte à gauche.	32
2.4	Les fichiers d'énergies en mode CASA	35
2.5	Les fichiers d'énergies en mode GB/FDB	36
2.6	Cycle thermodynamique qui définit l'affinité.	38
2.7	Les fichiers en sortie de proteus en haut pour le mode MONTECARLO, en bas pour le mode POSTPROCESS	39
2.8	Les principales structures « physiques » dans proteus	44
2.9	Les principales structures « logiques » dans proteus	45
2.10	Les principales structures « dynamiques » dans proteus	46

2.11 Les contributions aux énergies de groupes et d'interaction entre groupes dans la matrice d'énergie. L'énergie des groupes orange, bleu et vert est une somme de termes situés des carrés de même couleur dans la matrice d'énergie. L'interaction entre le groupe bleu et le groupe vert est la somme de tous les termes du rectangle rouge.	48
2.12 La matrice des énergies de groupe. Les énergies des groupes grp1, grp2, grp3 et grp4 et leurs interactions présentées sous forme de matrice. Il n'y a pas d'interaction entre grp3 et grp4 parce qu'ils sont associés à un même sous-système (dupliqué).	49
3.1 Exemple de fichier PDB avec déclaration de 3 modèles. L'avant-dernière colonne contient l'index des modèles. Par exemple, il y a un modèle pour GLY à la position 39 du segment A et un autre pour ASP, position 111, segment A.	52
3.2 La correspondance entre les directives OpenMP à droite et le comportement des fils d'exécution à gauche sur un REMC simplifié. Sont schématisés, la création d'une région parallèle, deux synchronisations (les lignes pointillées à gauche), une région exécutée uniquement par un fil maître, une affectation séquentielle.	54
3.3 Comportement de proteus pour un REMC à huit marcheurs. En haut, la trajectoire des marcheurs dans l'ensemble des huit températures. En bas, la distribution des énergies par marcheurs. On peut imaginer la distribution cumulée des marcheurs et remarquer que son graphe serait proche de celle d'une distribution de Boltzmann	55
4.1 Sequence logos for the core region of two designed proteins : 1CKA (SH3) and 2BYG (PDZ). The low energy CPD sequences are compared to the sequences of the full Pfam collection of experimental sequences. Positions shown correspond to the hydrophobic core of each protein ; residue numbers are indicated (PDB numbering).	69
4.2 The lowest energy sequence/structure combination is shown for the 2BYG PDZ domain (stereo view ; selected sidechains only ; white sticks), superimposed on the Xray structure (yellow sticks) ; the sidechains shown are the ones in the sequence logo, Fig.2. The main mutated positions are labelled ; other positions are either not mutated or mutated within the ILV group. Figure produced with pymol [1]	70
4.3 Histogram of Blosum40 similarity scores to Pfam sequences for the core region of two designed proteins : 1ABO (SH3) and 1BM2 (SH2). The similarity between Pfam sequences is also shown, considering either the Pfam seed alignment or the much larger full alignment.	71

4.4 Run times for different test calculations and search methods. CPU times per core are shown ; only the REMC calculations use multiple cores (OpenMP parallelization). CFN results are only given for 20 designed positions or fewer. REMC was only done for the larger calculations. Results are shown for a large, representative subset of the test calculations. A few CFN calculations were allowed to run beyond the 24h CPU limit. For clarity, the average Heur and REMC values are included as dashed lines.	72
4.5 Comparison between selected heuristic, MC, and REMC protocols for whole protein design. The best energy obtained with each protocol is shown for the nine test proteins. Zero represents the overall best energy for each protein. The mean “error” is indicated on the right for each protocol (the mean difference from the overall best energy). The protocols are a heuristic method (Heur), an MC method, two REMC protocols with 4 walkers (REMC a, b), and three REMC protocols with 8 walkers (yellow lines and dots, REMC c, d, e). Details of the protocols are in table 4.1 ; each curve is labelled according to the protocol nam. The dashed red line shows heuristic results with a more aggressive protocol (330,000 heuristic cycles) for the four rightmost proteins.	73
4.6 The lowest energies obtained with the different exploration methods for 5-, 10-, 20-, and 30-position design. Differences with respect to the GMEC or the overall best energy are shown, excluding the smallest values (less than 0.4 kcal/mol above the best energy). The color of each point indicates the nature of the test ; its shape indicates which method gave the best energy. Two examples are highlighted by arrows : the black arrow shows the heuristic result for a 10-position test where the best energy was given by CFN ; the gray arrow shows the MC result for a 20- position test where the best energy was given by the heuristic.	74
4.7 Sequence entropy $S(E)$ and number of states $N(E)$ within a given energy range E above the GMEC for the 1CKA SH3 domain. Ten positions were allowed to vary. The entropy (large dots) is a single position sequence entropy, Eq. (4.13), averaged over all the variable positions. CFN results (black) are based on a complete enumeration of all states within the energy range (at most E kcal/mol above the GMEC). REMC results (gray) are based on the states sampled by all eight walkers during a single trajectory. The number of states (small dots) corresponds to all the different combinations of sequences and rotamers.	80
5.1 Le cœur PDZ sélectionné En haut, l’alignement des huit séquences sauvages étudiées, les positions du cœur sont en jaunes. Au centre, la structure 3D des six coeurs de \mathcal{S}_1 superposés. En bas, la séquence et les « positions PDB » de chaque cœur.	98

5.2	Similarité des séquences des 6 protéines produites par Proteus et Rosetta à l'alignement Pfam RP55, sur l'ensemble des positions.	105
5.3	Similarité des séquences des 6 protéines produites par proteus modèle NEA et Rosetta à l'alignement Pfam RP55, sur les positions du cœur hydrophobe.	106
5.4	Similarité des séquences Proteus (NEA et FDB), Rosetta et des séquences de l'alignement Pfam RP55, sur toutes les positions à gauche et sur les positions du cœur à droite.	111
5.5	Les séquences conçues par Proteus, les homologues à la native et les séquences naturelles représentées sous forme de logo pour les positions du cœur hydrophobe	112
5.6	Les séquences conçues par Proteus, les homologues à la native et les séquences naturelles représentées sous forme de logo pour les positions exposées	113
5.7	Séquences Tiam1 obtenues avec un delta des énergies de références à -0,4, -0,2, 0, 0,2 et 0,4 et la structure native. Les hydrophobes pour des deltas de -0,4, -0,2, 0, 0,2 et 0,4 sont représentés par un dégradé allant du rouge foncé au vert clair en passant par le jaune.	113
5.8	Structure native Tiam1 avec les hydrophobes pour des δ de -0,4, -0,2, 0, 0,2 et 0,4 sont représentés par un jeu de couleurs allant du rouge foncé au vert clair en passant par le jaune.	114
5.9	Structure native Cask avec les hydrophobes pour des δ de -0,4,-0,2,0,0,2 et 0,4 sont représentés par un dégradé allant du rouge foncé au vert clair, en passant par le jaune.	116
5.10	Séquences Cask obtenues avec un delta des énergies de références à -0,4,-0,2,0,0,2 et 0,4 et la structure native.Les hydrophobes pour des deltas de -0,4,-0,2,0,0,2 et 0,4 sont représentés par un jeu de couleurs allant du rouge foncé au vert clair, en passant par le jaune.	116
5.11	L'alignement de notre sélection de séquences homologues à la protéine NHREF (code PDB : 1G9O)	119
5.12	Une sélection de séquences proteus, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine NHREF (code PDB : 1G9O), modèle NEA	120
5.13	L'alignement de notre sélection de séquences homologues à la protéine INAD (code PDB : 1IHJ)	121
5.14	Une sélection de séquences proteus, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine INAD (code PDB : 1IHJ), modèle NEA	122
5.15	L'alignement de notre sélection de séquences homologues à la protéine Syntenin (code PDB : 1R6J)	123
5.16	Une sélection de séquences proteus, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine Syntenin (code PDB : 1R6J), modèle NEA	124

Liste des figures

5.17 L'alignement de notre sélection de séquences homologues à la protéine GRIP (code PDB : 1N7E)	125
5.18 Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine GRIP (code PDB : 1N7E), modèle NEA	126
5.19 L'alignement de notre sélection de séquences homologues à la protéine DLG2 (code PDB : 2BYG)	127
5.20 Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine DLG2 (code PDB : 2BYG), modèle NEA	128
5.21 L'alignement de notre sélection de séquences homologues à la protéine PSD95 (code PDB : 3K82)	129
5.22 Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine PSD95 (code PDB : 3K82), modèle NEA	130
5.23 L'alignement de notre sélection de séquences homologues à la protéine Tiam1 (code PDB)	131
5.24 L'alignement de notre sélection de séquences homologues à la protéine Cask (code PDB)	132

Liste des tables

2.1	Une partie des balises possibles du fichier de configuration de proteus	36
4.1	Selected MC and REMC protocols	65
4.2	Reference energies for unfolded state (kcal/mol).	66
4.3	Test proteins	67
4.4	Designed sequence quality measures	71
4.5	Tests with 10 designed positions	76
4.6	Tests with 20 and 30 designed positions	78
4.7	Designed and Pfam sequence entropies	79
4.8	La sélection des positions pour tests 5-actives	85
4.9	La sélection des positions pour tests 10-actives	86
4.10	La sélection des positions pour tests 20-actives	87
4.11	La sélection des positions pour tests 30-actives	88
5.1	Les groupes d'acides aminés utilisés pour l'optimisation des énergies de référence.	94
5.2	La sélection de domaines protéiques PDZ	95
5.3	E-value et pourcentage d'identité des alignements Blast native versus native pour nos séquences PDZ	96
5.4	Sélection des homologues.	96
5.5	Similarité des séquences expérimentales homologues des huit protéines PDZ. .	97
5.6	Les énergies de référence (kcal/mol) optimisée sur six protéines.	101
5.7	Les compositions en acide aminé (%) des séquences expérimentales et Proteus après optimisation des E_t^s . Les différences entre expérimentale et théorique sont indiquées entre parenthèses. Les types d'acides aminés sont rassemblés selon les groupes d'optimisations.	102
5.8	Résultats Superfamily pour les séquences Proteus avec le modèle NEA.	103
5.9	Résultats Superfamily pour les séquences Rosetta	103
5.10	Moyenne de l'exponentielle de l'entropie sur les ensembles des positions des protéines	104
5.11	La composition en acide aminé (%) des séquences expérimentales et Proteus après optimisation FDB des énergies de référence. La différence entre expérimentales et Proteus sur les classes est donnée entre crochets.	108

Liste des tables

5.12 Les énergies de référence obtenues avec l'optimisation sur 3 protéines. La constante diélectrique est fixée à 4.	109
5.13 Résultats Superfamily pour les séquences Proteus avec le modèle FDB (énergies de références optimisées sur 20 cycles selon les classes + 20 cycles selon les types),le modèle NEA et Rosetta.	110
5.14 Pourcentage d'identité moyen à la séquence native	110

Abréviations

3D tri-dimensionnel(le)

AMBER Assited Model Building and Energy Refinement

backbone (ou squelette) chaîne principale de la protéine

BLOSUM BLOcks of amino acid SUbstitution Matrix

CASA Coulomb Accessible Surface Area

CHARMM Chemistry at HARvard Macromolecular Mechanics

CPD Computational Protein Design

DEE Dead-End Elimination

DFBB Depth-First Branch and Bound

EPT Equivalence Preserving Transformation

FDB Fluctuating DIelectric Boundary

GB Born Généralisé

GMEC « Global minimal energie cost »

MC algorithme Monte-Carlo

MSD Multistart Steepest Descent heuristic

NEA Naive Environnement Aproximation

PB Poisson-Boltzmann

PDZ « Postynaptic density-95 / Discs large / Zonula occludens-1 »

Pfam « Protein family databank »

REMC Replica Exchange Monte-Carlo

SH2 Src Homology region 2

SH3 Src Homology region 3

Chapitre 1

Le « CPD » : Conception de protéine par ordinateur

L'ingénierie des protéines est l'ensemble des techniques pour modifier la structure ou la fonction d'une protéine en modifiant sa séquence d'acides aminés. Les objectifs sont variés. On peut citer l'augmentation de la stabilité, la modification des fonctionnements enzymatiques ou encore l'ajout d'une conformation alternative à une protéine.

Une approche est la mutagenèse dirigée. La première étape est l'identification des mutations intéressantes, puis des méthodes de génie génétique sont utilisées pour produire les mutants dont les propriétés souhaitées pourront être vérifiées a posteriori. Une deuxième approche est l'évolution dirigée, dans laquelle un ensemble de mutations aléatoires est effectué sur protéine et toutes les séquences ainsi produites sont testées afin de trouver la fonction attendue. La sélection se fait alors, comme celle de l'évolution naturelle dont elle reprend les mécanismes, sur les séquences positives aux tests.

Une autre approche est apparue avec la naissance des méthodes d'ingénierie des protéines « *in silico* ». L'une d'elles, le « Computational Protein Design » ou CPD consiste à déterminer les séquences d'acides aminés compatibles avec une structure protéique donnée, on parle également de résolution du problème inverse du repliement (voir 1.1). Cela implique la connaissance de la structure tridimensionnelle de la protéine. Cette méthode comporte trois éléments principaux :

1. La définition d'un espace de conformations de la protéine

C'est sur elle que repose la prédiction de structure des séquences considérées. Elle doit inclure une ou plusieurs conformations de la chaîne principale du polypeptide et un ensemble de positionnements de la chaîne latérale de chaque résidu.

2. Une fonction d'énergie

Elle permet d'évaluer la pertinence des conformations.

3. Un algorithme d'exploration de l'espace de séquences-conformations

Il exploite la fonction d'énergie pour échantillonner les séquences favorables.

Dans la suite de ce chapitre, nous détaillons les trois composantes du CPD. Nous aborderons le problème de la modélisation des protéines, de leur espace de conformation et

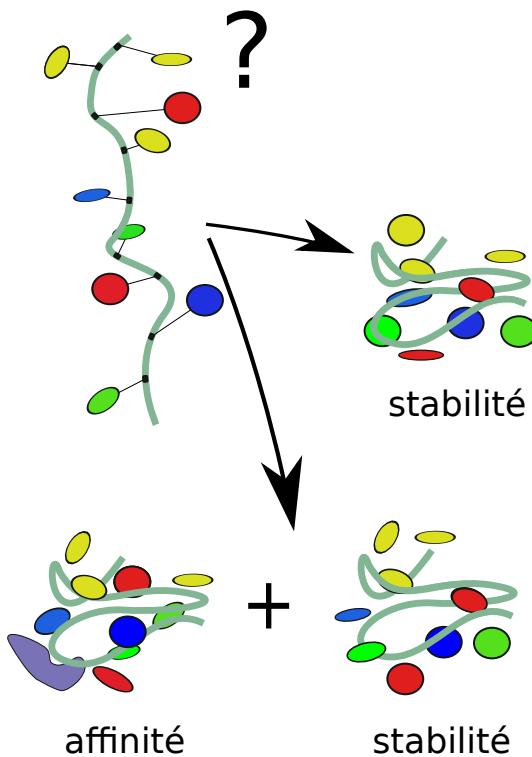


Figure 1.1 – Le CPD pour « Computational Protein Design » recherche les séquences d’acides aminés compatibles avec une protéine dont la structure 3D est connue, c’est-à-dire compatibles avec un repliement et éventuellement une affinité pour un ligand.

de l'espace séquences-conformations. Puis, nous verrons les fonctions d'énergies classiques. Seront détaillées les approches pour la modélisation du solvant qui est une partie importante de la fonction d'énergie. Plusieurs algorithmes d'exploration de l'ensemble des états possibles vont être abordés.

1.1 L'espace des séquences-conformations

Pour faire du CPD, il faut une représentation de l'ensemble des conformations que les chaînes polypeptidiques peuvent adopter. L'ensemble des cas possibles peut se concevoir comme le choix d'une séquence S d'acides aminés de longueur N et la détermination d'une conformation C prise par S dans l'espace 3D. Ainsi l'espace d'états est celui des couples (S,C) pour N donné. Dans la suite, on appelle une séquence-conformation un élément de cet ensemble de couples. L'ensemble des séquences de N acides aminés se conçoit sans difficulté comme les N -uplets de l'ensemble à 20 éléments formé des différents types d'acide aminé. En revanche, la définition de l'ensemble des conformations d'une chaîne polypeptidique doit être développée.

1.1.1 L'état replié

La mécanique moléculaire propose d'appliquer les représentations de la mécanique classique aux molécules. Les atomes sont représentés sous forme de sphères. La protéine

dans un milieu aqueux est flexible et en permanence en mouvement. C'est en particulier le cas pour les chaînes latérales et les boucles flexibles. L'espace d'états d'une chaîne polypeptidique, dans le cadre de la mécanique moléculaire est alors constitué d'un vaste espace continu de conformations possibles.

D'autre part, si un domaine polypeptidique a un nombre N de résidus et que chacun peut adopter l'un des 20 types d'acides aminés, le nombre de séquences à considérer est égale à 20^N . Il en résulte un espace des séquences-conformations trop grand pour être utilisable. Il faut donc réduire la taille de l'espace des conformations. Pour cela, Ponder et Richards [2] proposent une approche en deux points :

1. Le squelette de la protéine est fixé.
2. Les conformations des chaînes latérales sont réduites à un ensemble fini de positionnements possibles dans l'espace euclidien, les « rotamères ».

Ensuite, des variations sur ce principe ont été introduites, avec notamment la prise en compte de la mobilité de la chaîne principale dans un ensemble discret ou continu d'états. L'approche qui consiste à générer un ensemble de squelettes et à faire des calculs CPD pour l'ensemble a été utilisée par Su et Mayo [3]. Présentons maintenant, la modélisation des chaînes latérales et celle du backbone, c'est-à-dire le squelette polypeptidique.

Les chaînes latérales

Finkelstein et Ptitsyn [4], Janin et al. [5] ainsi que Ponder et Richards [2] ont établi que la chaîne latérale des résidus adopte de façon préférentielle un petit ensemble de conformations. Janin introduit alors le terme de « rotamère » pour désigner ces conformations. Il est alors possible de réduire l'espace continu des conformations des acides aminés à cet ensemble discret de rotamères. La plupart des méthodes de CPD utilisent cette discréétisation. Beaucoup de librairies de rotamères ont été proposées dans la littérature. La plupart sont indépendantes du backbone. Mais il existe également des librairies qui dépendent du squelette de la protéine, voir [6, 7].

Le squelette

Comme le positionnement des chaînes latérales n'influence que faiblement la structure adoptée par le squelette, celui-ci est fixé dans beaucoup de programmes CPD. Le problème de la prédiction de structure est alors ramené à celui du placement des chaînes latérales sur ce squelette. De par la configuration particulière de la proline et de la cystéine dans le backbone, ces deux types sont souvent traités à part. Cette approche a obtenu de nombreux succès, voir par exemple [8].

Cependant, cette approximation peut avoir des conséquences importantes. Un type d'acide aminé considéré comme défavorable peut devenir favorable après une petite adaptation du backbone et il a été établi que quelques mouvements de squelettes peuvent faire varier significativement l'énergie de la conformation [9]. En réponse à ce problème,

Druart et al. [10] proposent une méthode de Monte Carlo hybride travaillant sur un modèle multi backbone. Une autre approche consiste à donner une certaine liberté aux angles Φ , Ψ en introduisant des variations aléatoires [9]. Dantas et al. [11] font des simulations avec minimisation après chaque mouvement de chaîne latérale. Kuhlman et al. [12] optimisent alternativement la structure du squelette et la séquence d'acides aminés.

Enfin, citons l'utilisation d'une classe particulière de mouvement des squelettes protéiques appelée « backrub ». Ce sont des mouvements naturels du backbone mis en évidence par David et al. [13], à partir de structures cristallographiques. Ces mouvements consistent en des déplacements de l'ensemble $C_\alpha - C_\beta$ à une position i donnée de la chaîne, sans déplacement des carbones $C_{\alpha_{i+1}}$ et $C_{\alpha_{i-1}}$. Ces mouvements backrub ont permis [14, 15] d'améliorer la qualité des prédictions des mutants par rapport à des simulations à squelette rigide.

1.1.2 L'état déplié

En général, la stabilité d'une protéine est évaluée par une variation de l'énergie libre entre son état déplié et son état replié. Il faut alors connaître l'énergie de l'état déplié. Mais cet état est déstructuré par définition et ne correspond pas à une conformation unique ; la modélisation exhaustive est difficile. Une approche simple consiste à représenter cet état par une chaîne étendue dans laquelle un résidu de la protéine est en interaction principalement avec le solvant et avec le backbone. Ainsi l'énergie libre de l'état déplié dépend de la séquence uniquement par la composition en acides aminés de celle-ci. En pratique, on peut utiliser pour chaque type d'acide aminé X de la protéine un tripeptide ALA–X–ALA, et on identifie son exposition au solvant à celle dans l'état déplié [16]. On en déduit une énergie par type X que l'on somme sur la séquence pour obtenir l'énergie de la protéine dépliée.

1.2 L'énergie d'une protéine

La fonction d'énergie permet d'évaluer la stabilité de chaque conformation de la protéine. Cette fonction doit prendre en compte les détails des interactions entre les atomes de la protéine, les effets de l'environnement aqueux et en même temps, être suffisamment rapide à calculer pour évaluer en un temps raisonnable une partie la plus significative possible de l'espace des conformations. Une classe importante est basée sur la mécanique moléculaire.

1.2.1 La mécanique moléculaire

La mécanique moléculaire représente les atomes comme des particules sphériques ayant une charge électrique nette fixe généralement placée au centre de la sphère et chaque liaison est modélisée par un ressort. Cette description, associée à un choix de paramètres numériques optimisés est appelée un champ de force. Ce champ de force décrit

les interactions interatomiques du point de vue énergétique et est invariant au cours d'une simulation. Dans toute la suite, nous appelons E_{MM} l'énergie qui dérive d'un tel champ de force.

Il existe beaucoup de champs de force à la disposition des simulateurs. Quatre des principaux optimisés pour les protéines sont :

- AMBER : Assisted Model Building with Energy Refinement [17]
- CHARMM : Chemistry at HARvard Molecular Mechanics [18]
- OPLS : Optimized Potential for Liquid Simulations [19]
- GROMOS : GROningen MOlecular Simulation [20]

L'énergie d'une conformation se définit alors comme la somme de l'énergie E_{MM} et de l'énergie de solvatation :

$$E = E_{MM} + E_{solv} \quad (1.1)$$

Le terme E_{MM} se décompose à son tour en deux termes :

$$E_{MM} = E_{liée} + E_{non liée} \quad (1.2)$$

avec $E_{liée}$ l'énergie d'interactions des atomes éloignés d'au plus deux liaisons covalentes et $E_{non liée}$ l'énergie des autres interactions. Détaillons ces deux énergies.

Les interactions liées

L'énergie d'interaction liée comprend un terme d'elongation des liaisons, un terme de déformation des angles, de rotation des angles dièdres et un terme « impropre » :

$$E_{liée} = E_{liaison} + E_{angle} + E_{dièdre} + E_{impr} \quad (1.3)$$

L'énergie d'elongation des liaisons $E_{liaison}$ s'exprime de la façon suivante :

$$E_{liaison} = \frac{1}{2} \sum_{i=1}^n k_i (r_i - r_i^0)^2 \quad (1.4)$$

avec l'ensemble des liaisons indexé par i , k_i la force du ressort, r_i la longueur de la liaison et r_i^0 la longueur optimale. L'énergie de déformation des angles E_{angl} s'exprime :

$$E_{angl} = \frac{1}{2} \sum_{ij} k_{\theta,ij} (\theta_{ij} - \theta_{ij}^0)^2 \quad (1.5)$$

avec θ_{ij} l'angle entre les liaisons i et j , θ_{ij}^0 l'angle optimal et $k_{\theta,ij}$ la force du ressort. L'énergie de déformation des angles dièdres $E_{dièdre}$ s'exprime :

$$E_{dihe} = \frac{1}{2} \sum_i A_{i,n} [1 + \cos(n\Phi_i - \Phi_0)] \quad (1.6)$$

où n est périodicité de la rotation, $A_{i,n}$ est la constante de torsion, Φ_i l'angle dièdre, c'est-à-dire pour 4 atomes a_1, a_2, a_3 et a_4 , reliés par 3 liaisons $a_1 - a_2, a_2 - a_3$ et $a_3 - a_4$, l'angle formé par les plans (a_1, a_2, a_3) et (a_2, a_3, a_4) , Φ^0 est la phase. L'énergie de déformation des angles impropre E_{impr} exprime la déformation d'un ensemble de 4 atomes par rapport à une conformation plane ou tétraédrique. Pour un atome a_1 relié à 3 atomes a_2, a_3 et a_4 , elle a la forme :

$$E_{impr} = \frac{1}{2}A(\omega - \omega^0)^2 \quad (1.7)$$

Ici, A est la constante de force et ω représente l'angle entre les plans (a_1, a_2, a_3) et (a_2, a_3, a_4) , voir la figure 1.2 pour une représentation visuelle.

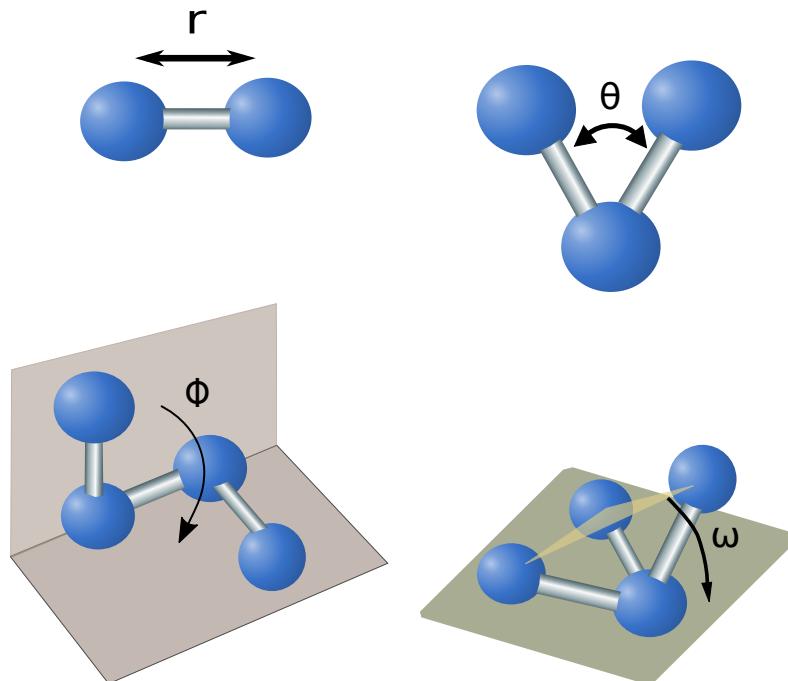


Figure 1.2 – **Représentation des énergies liées** De la gauche vers la droite et du haut vers le bas, sont représentées les énergies de liaison, d'angle, du dièdre et du dièdre impropre.

Les interactions non liées

Les interactions non liées sont les interactions entre atomes séparés par plus de trois liaisons covalentes ou qui appartiennent à des molécules différentes. Les interactions sont caractérisées par les deux termes suivants :

$$E_{non\ liées} = E_{elec} + E_{vdw} \quad (1.8)$$

L'énergie E_{elec} pour l'énergie électrostatique est donnée par un potentiel de Coulomb de la forme :

$$E_{elec} = \frac{q_i q_j}{\epsilon r_{ij}} \quad (1.9)$$

q_i et q_j représentent les charges respectives des atomes i et j et r_{ij} la distance entre les atomes i et j , ϵ la constante diélectrique du milieu.

L'énergie E_{VdW} de Van der Waals s'explique par les interactions électriques de faible intensité entre deux atomes, dus au fait que la répartition des charges électriques n'est pas uniforme autour des noyaux. Elle rassemble les effets des forces de Keesom, Debye, London et Pauli. Le potentiel de Lennard-Jones est l'approximation classique de cette énergie. Voici son expression :

$$E_{vdw} = \sum_{i < j} D_0 \left[\left(\frac{r_0}{r_{ij}} \right)^{12} - \left(\frac{r_0}{r_{ij}} \right)^6 \right] \quad (1.10)$$

avec D_0 et r_0 des constantes, r_{ij} la distance entre l'atome i et l'atome j . Le premier terme est répulsif à courte distance, ce qui représente l'évitement de l'encombrement stérique entre i et j . Le second terme domine à grande distance, c'est un terme attractif.

1.2.2 D'autres approches

Même si la mécanique moléculaire prédomine dans le monde du CPD, il existe d'autres approches. Zollars et al. utilisent une fonction empirique pour la modélisation des liaisons hydrogène avec une fonction d'énergie de Coulomb qui contient un ϵ qui varie en fonction de la distance interatomique [21]. Il existe des fonctions d'énergie comportant des éléments relevant de statistiques sur les protéines [22]. Il existe encore des fonctions d'énergie à gros grains notamment dans la modélisation des forces de Van der Waals utiles pour les interactions protéine-protéine [23].

1.3 La modélisation du solvant

Les protéines sont étudiées dans un solvant aqueux. La solution qui est considérée dans les calculs est un mélange dans lequel l'eau est présente en quantité largement plus importante que la protéine. Bien qu'il existe d'autres types de solvant, ils ne sont pas abordés ici. Les interactions entre solutés et solvant jouent un rôle clé dans la structure de la protéine, mais également dans sa fonction. La modélisation du solvant est alors un point capital pour le CPD.

Une molécule d'eau a une charge électrique nette nulle. Mais il existe une dissymétrie du nuage électronique : les électrons se situent de façon préférentielle au voisinage du noyau de l'oxygène par rapport à ceux des deux noyaux d'hydrogènes. On dit que la molécule est polaire, et cette polarisation peut être approchée par un moment dipolaire. Cette polarité permet la formation de liaisons hydrogène entre molécules d'eau et avec tout autre groupe polaire. Ainsi, l'eau à l'état liquide possède beaucoup de liaisons hydrogène.

Lorsqu'une protéine est solvatée, les molécules du solvant se placent de telle façon que le nombre de liaisons hydrogène soit maximal. Les molécules de la première couche de

solvatation forment alors des liaisons avec les groupes polaires de la protéine, près des groupes non polaires, s'orientent pour former des liaisons avec d'autres molécules d'eau. Cette réorganisation de la structure moléculaire de l'eau a trois conséquences importantes :

- Une diminution de l'entropie au voisinage de la protéine
- La création de polarisation à la surface du soluté
- L'eau crée une couche de charges partielles opposées à celles de la protéine qui atténue le champ de celle-ci. Ce phénomène s'appelle l'écrantage.

Il a été démontré qu'une protéine et une molécule d'eau peuvent interagir de façon non négligeable jusqu'à une distance de 15 Angströms. Cela implique que pour solvater correctement une protéine, il faut prendre en compte l'effet de plusieurs milliers d'atomes du solvant. De nombreuses méthodes ont été développées pour faire face à la difficulté que cela représente. La connaissance des positions et les vitesses des atomes de toutes les molécules d'eau nécessaire dans le cadre de la mécanique moléculaire, apparaît alors comme une gageure. Des approches moins fines doivent être utilisées.

1.3.1 Le modèle de solvant explicite

Pour calculer les grandeurs d'intérêt du solvant, il faut pouvoir situer le système dans l'espace des phases, un espace à $6N$ dimensions pour une simulation du solvant avec N molécules d'eau. Une première méthode est la dynamique moléculaire dans laquelle, à partir d'une configuration initiale des molécules d'eau, les équations de la mécanique classique sont résolues pour déterminer les positions et les vitesses au cours du temps. Une seconde méthode consiste à échantillonner l'espace des phases par la méthode Monte Carlo dans laquelle l'espace est visité grâce à une série de déplacements aléatoires qui sont acceptés ou non en fonction de critères basés sur l'énergie (voir la section 1.4.6).

Quelle que soit la méthode utilisée, pour pouvoir obtenir une représentation de qualité des effets du solvant sur la protéine, il faut échantillonner correctement les états du solvant dans l'espace des phases, ce qui demande de multiplier les dynamiques moléculaires avec différentes configurations initiales ou de calculer des trajectoires Monte Carlo suffisamment longues. L'utilisation d'un modèle de solvant explicite entraîne alors une situation dans laquelle une simulation très coûteuse en termes de puissance de calculs consacre la plus grande partie du temps à évaluer des interactions entre des molécules d'eau. Ce n'est pas un objectif pour le CPD.

1.3.2 Le modèle implicite

Le principe des modèles implicites du solvant est de représenter l'effet moyen du solvant sur la protéine par l'utilisation d'un milieu continu dans lequel la protéine serait immergée. Le solvant et la protéine sont représentés chacun par un milieu diélectrique uniforme. Pour

mesurer l'effet de la solvatation, la bonne quantité est l'énergie libre de solvatation mise en jeu pendant ce processus. Elle se définit comme la différence entre l'énergie libre de la solution et l'énergie libre d'un système dans lequel le solvant et le soluté sont séparés et à l'état pur. Notons cette différence ΔG_{solv} . On peut décrire ΔG_{solv} comme la somme de trois termes :

$$\Delta G_{solv} = \Delta G_{solv}^{elec} + \Delta G_{solv}^{vdw} + \Delta G_{solv}^{cav} \quad (1.11)$$

avec :

- ΔG_{elec} l'effet électrostatique, qui correspond à la réorganisation des charges de la protéine dans le solvant, y compris ses charges partielles (polarisation).
- ΔG_{vdw} est l'effet des forces de Van der Waals.
- ΔG_{cav} est l'effet qui correspond au coût de la création de la cavité, en termes d'entropie et de pression. Il inclut le coût de réorganisation des molécules du solvant.

Une méthode standard pour approcher ΔG_{solv} est de traiter le premier terme séparément des autres. En effet, il est possible d'approcher la somme des deux derniers termes de l'équation 1.11 en utilisant la surface accessible au solvant de la protéine. Cette surface est l'ensemble des points par lesquels peut passer le centre d'une sphère, modélisant une molécule d'eau, qui roule sur la surface de Van der Waals de la protéine, voir figure 1.3. L'approximation s'écrit :

$$\Delta G_{solv}^{vdw} + \Delta G_{solv}^{cav} \approx E_{solv}^{surf} = \sum_i \sigma_{t_i} A_i \quad (1.12)$$

avec A_i la surface accessible au solvant de l'atome i et σ_{t_i} un facteur pour chaque type atomique t_i ajusté pour retrouver les énergies de solvatation obtenues par expérience. Dans la suite, on appelle cette somme, le terme surfacique de l'énergie de solvatation, et on le note E_{solv}^{surf} .

1.3.3 Le modèle CASA

Pendant la réorganisation du solvant, les molécules d'eau s'orientent, au moins pour les plus proches du soluté, selon les lignes du champ électrique créé par la protéine, voir figure 1.4. L'idée du modèle « Coulomb accessible surface area » (CASA) est alors de considérer que l'effet électrostatique induit par le solvant est proportionnel à l'effet électrostatique produit par la protéine. On a alors, en utilisant le modèle SA pour le terme surfacique :

$$\Delta G_{solv} \approx E_{screen} + \sum_i \sigma_{t_i} A_i \quad (1.13)$$

avec

$$E_{screen} = \left(\frac{1}{\epsilon} - 1 \right) E_{coul} \quad (1.14)$$

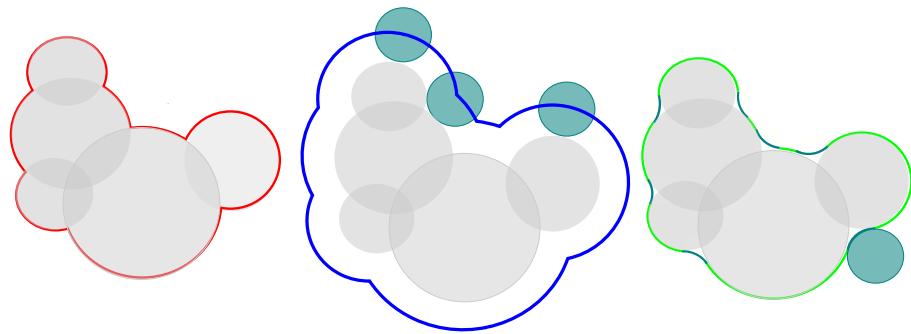


Figure 1.3 – Les trois types classiques de surfaces pour une molécule À gauche et en rouge la surface de Van der Waals (SVdW), au centre et en bleu, la surface accessible au solvant (SA), à droite, et en vert la surface de Connolly ou surface moléculaire (SM). La surface SA est l'ensemble des positions pouvant être occupées par le centre d'une sphère (figurant une molécule du solvant) roulant sur SVdW. La surface SM est la plus petite enveloppe de SVdW dont chaque point peut être en contact avec la sphère.

et ϵ la constante diélectrique du solvant. Ainsi, l'effet du solvant est décrit par un facteur unique pour toutes les interactions électrostatiques. La simplicité de ce modèle proposé par Wesson et Einsenberg [24] fait de lui un modèle fréquemment utilisé. Cependant, il est en difficulté pour le traitement de la surface des protéines.

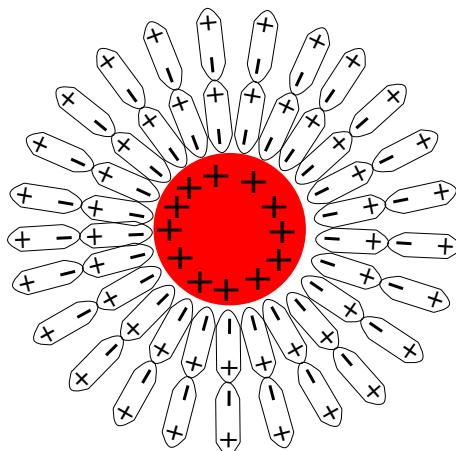


Figure 1.4 – Représentation schématique de l'organisation des dipôles des molécules d'eau autour d'un soluté sphérique chargé positivement à sa surface. Les dipôles des premières couches de solvatation s'orientent suivant les lignes du champ électrique produit par le soluté.

1.3.4 Le modèle Poisson-Boltzmann

La méthode de Poisson-Boltzmann est très utilisée parce qu'elle prend en compte l'effet ionique et toute la forme de la protéine. Dans cette méthode, la protéine est supposée fixe, elle forme une cavité dans un milieu continu diélectrique qui peut être polarisé (on parle de continuum), il s'agit donc encore d'un modèle de solvant implicite. Ainsi le modèle prend en compte l'écrantage par le solvant et les interactions électrostatiques entre groupes chargés de la protéine et du solvant polarisé. Si le continuum est muni d'une distribution

de charge ρ alors le potentiel électrostatique dans ce milieu est donné par l'équation de Poisson :

$$\nabla[\epsilon(x)\phi(x)] = -4\pi\rho(x) \quad (1.15)$$

avec $\epsilon(x)$ la constante diélectrique (ici c'est une fonction à deux valeurs possibles selon le milieu), $\phi(x)$ le potentiel électrostatique, $\nabla\phi(x)$ la divergence de ϕ en x . Typiquement ϵ prend une valeur faible, entre 1 et 8 pour la protéine et élevée pour le solvant, proche de 80.

L'équation de Poisson-Boltzmann est une extension de l'équation de Poisson dans laquelle les charges d'ions mobiles sont prises en compte. La théorie de Debye-Hückel donne la distribution des ions comme suivant une loi de Boltzmann, conduisant à :

$$\nabla[\epsilon(x)\phi(x)] = -4\pi(\rho(x) - \epsilon(x)\kappa^2\phi(x)) \quad (1.16)$$

avec κ le paramètre de Debye-Hückel qui tient compte de la concentration des ions en solution.

Malheureusement, il n'existe pas, pour les protéines, de solution analytique à 1.15. Il faut effectuer une résolution numérique. Plusieurs programmes sont à la disposition de la communauté, on peut citer Delphi [25] et APBS [26] qui sont basés sur la méthode des différences finies et des éléments finis. Une fois l'équation résolue, le terme électrostatique de l'énergie de solvatation est donné par :

$$\Delta G_{solv}^{elec} = \frac{1}{2} \int \rho(x)\phi(x)dx \quad (1.17)$$

Quelle que soit l'approche utilisée, les calculs sont coûteux pour une protéine entière.

1.3.5 Le modèle de Born Généralisé

Dans le cas d'un soluté sphérique de rayon a et de charge ponctuelle q portée par le centre de la sphère, Born en 1920 [27] établit des expressions analytiques des solutions de l'équation de Poisson-Boltzmann et de ΔG_{solv}^{elec} :

$$\Delta G_{solv}^{elec} = -\tau \frac{q^2}{2a} \quad (1.18)$$

avec $\tau = \frac{1}{\epsilon_p} - \frac{1}{\epsilon_s}$, ϵ_p la constante diélectrique de la sphère et ϵ_s celle du solvant. Partant de cette solution en 1990, Still et al. [28] proposent une généralisation à un ensemble d'atomes chargés formant une cavité de forme quelconque. L'objectif est de donner les résultats de qualité proche de celles de PB avec un coût numérique bien inférieur.

Dans un premier temps, on évalue l'énergie libre électrostatique totale d'une paire d'atomes i, j de rayon a_i, a_j de charge q_i, q_j séparés d'une distance r_{ij} , en utilisant la solution de Born :

$$G_{elec} = E_{Coul} + \Delta G_{solv} \quad (1.19)$$

ce qui donne, si les particules sont éloignées ;

$$G_{elec} = \frac{1}{2} \frac{q_i q_j}{\epsilon_s r_{ij}} - \frac{1}{2} \tau \left(\frac{q_i^2}{a_i} + \frac{q_j^2}{a_j} \right) \quad (1.20)$$

ou, si $i = j$:

$$G_{elec} = -\tau \frac{q_i^2}{2a_i} \quad (1.21)$$

avec $\tau = \frac{1}{\epsilon_p} - \frac{1}{\epsilon_s}$, ϵ_p la constante diélectrique des particules i et j et ϵ_s celle du solvant. L'idée consiste alors à étendre la somme des deux derniers termes de 1.20 aux distances r_{ij} intermédiaires :

$$G_{elec} = \frac{1}{2} \frac{q_i q_j}{\epsilon_s r_{ij}} - \tau \frac{1}{2} g_{ij} \quad (1.22)$$

La forme des g_{ij} la plus communément utilisée est la forme paramétrique suivante :

$$g_{ij} = \tau q_i q_j (r_{ij}^2 + b_i b_j \exp(-\frac{r_{ij}^2}{4b_i b_j}))^{-1/2} \quad (1.23)$$

avec b_i et b_j , de nouveaux paramètres, que l'on nomme les rayons de solvatations des atomes i et j . Dans le cas où $i = j$, g_{ij} est égale à deux fois l'énergie de solvation de Born pour un ion de rayon b_i . Lorsque r_{ij} tend vers l'infini, g_{ij} tend vers la loi de Coulomb dans le solvant : $\frac{q_i q_j}{\epsilon_s r_{ij}}$. On écrit finalement pour une protéine constituée de N atomes :

$$G_{elec} = \frac{1}{2} \sum_{i \neq j} \frac{q_i q_j}{\epsilon_s r_{ij}} - \tau \frac{1}{2} \sum_{i,j}^N g_{ij} \quad (1.24)$$

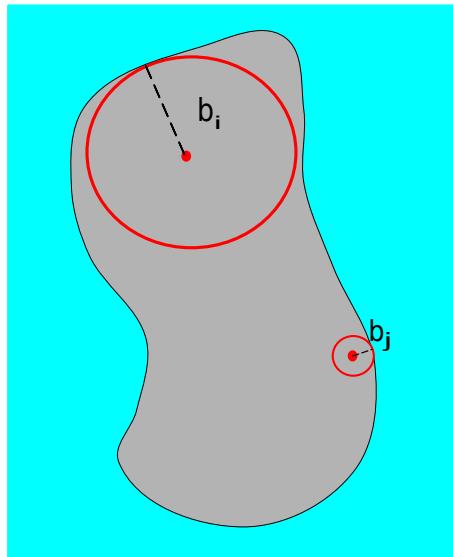


Figure 1.5 – Des rayons de Born de deux atomes dans une protéine Le rayon de Born pour l'atome enfui est environ le rayon de la protéine, alors que le rayon de Born d'un atome à la surface est environ son rayon de Van der Waals.

Il reste à choisir une approximation pour les rayons de Born. La méthode classique se fait par l'énergie libre électrostatique de l'atome i , dans la situation où toutes les

autres particules ont une charge ramenée à zéro. Ainsi, chaque charge de la protéine est caractérisée par sa distance au solvant, voir la figure 1.5. Alors, ce modèle permet le calcul de l'énergie libre de solvatation électrostatique à partir de la seule connaissance du volume de la protéine. En revanche, comme ces b_i sont fonction de la position relative à i de tous les atomes de la protéine, ils dépendent de sa conformation :

$$b_i = b_i(r_1, r_2, \dots, r_N)$$

Ainsi, l'interaction entre une paire i et j dépend des positions de tous les atomes.

« Native Environnement Approximation » (NEA) Dans la variante NEA pour « Native Environnement Approximation » du modèle GB, le rayon de solvatation des atomes de chaque chaîne latérale est calculé avant l'étape d'exploration, en fixant tout le reste du système dans sa séquence native et sa conformation native [29–31]. Par la suite, les b_i ont des valeurs constantes, qui ne dépendent pas de la conformation. Ainsi G_{elec} prend la forme d'une somme de termes qui ne dépendent chacun que d'une paire d'atomes i, j : on dit qu'elle est « décomposable par paires ».

1.4 L'algorithme d'exploration

Il reste à déterminer un algorithme d'exploration de l'espace d'états qui permet la sélection d'un sous-ensemble de séquences les plus pertinentes. L'objectif du CPD est d'obtenir l'ensemble des séquences d'acides aminés compatibles avec une structure 3D ou qui réalisent une fonction donnée. Un tel algorithme a pour grand défi de faire face à l'immensité de l'espace d'états. On peut classer les différentes approches utilisées en deux groupes.

Le premier groupe est constitué des méthodes déterministes dans lesquelles les choix effectués sont toujours déterminés *a priori* ou déterminés en fonction des éléments obtenus au cours de l'exécution du programme. On trouve par exemple dans ce groupe, les méthodes exhaustives qui exhibent la totalité des solutions du problème. Elles peuvent être appliquées à de petites espaces conformationnels. On trouve aussi des méthodes dites semi-exhaustives dans lesquelles la complexité combinatoire est réduite en autorisant uniquement certaines conformations à certains moments de l'exploration.

Une classe intéressante d'approches déterministes est constituée des algorithmes exacts qui se focalisent non pas sur l'objectif du CPD, mais sur la conformation de meilleure énergie ou « Global Minimum Energy Conformation » (GMEC). Typiquement, ces algorithmes exploitent les structures de la fonction d'énergie et de l'espace d'états. Un type de méthodes est alors exploitable pour un type de fonction d'énergie. Bien souvent, ces algorithmes nécessitent également la discréétisation de l'espace des phases. Cela peut devenir problématique dans les situations où le backbone est flexible.

Le second groupe est celui des méthodes stochastiques et/ou heuristiques. Elles ont vocation à déterminer des solutions de qualité sans obtenir de garantie sur l'optimum en termes d'énergie, dans un temps d'exécution réaliste. Elles sont non-déterministes c'est-à-dire qu'elles choisissent certains éléments de façon aléatoire, en pratique la « source de hasard » est constituée de générateurs de nombres pseudo-aléatoires. Elles permettent des espaces d'états non discrétilisés, par exemple [32] où les rotamères peuvent varier de façon continue. Plusieurs peuvent s'utiliser sans exiger de structure particulière pour la fonction d'énergie. Par contre, la convergence bien qu'elle puisse être établie en théorie, reste en pratique difficile à cerner et n'offre pas la possibilité de reconnaître le GMEC. On doit au besoin choisir une condition d'arrêt de l'exécution sans lien direct avec ce minimum.

Pour rendre possible l'utilisation de plusieurs outils algorithmiques, la fonction d'énergie doit être décomposable par paires, c'est-à-dire qu'elle doit être décomposable en une somme sur des paires de résidus. L'énergie d'une conformation C a alors la forme :

$$E(C) = \sum_i E_i(r_i) + \sum_{i \neq j} E_{ij}(r_i, r_j) \quad (1.25)$$

avec r_i et r_j les rotamères de la conformation C aux résidus i et j , $E_i(r_i)$ l'énergie propre de r_i et $E_{ij}(r_i, r_j)$ l'énergie d'interaction entre r_i et r_j .

À présent, nous présentons quelques algorithmes parmi les plus utilisés.

1.4.1 L'algorithme du champ moyen

La méthode du champ moyen substitue l'ensemble des interactions entre un rotamère r_0 et les autres rotamères par une interaction unique moyenne. Pour calculer une interaction moyenne, la probabilité de Boltzmann est utilisée de la façon suivante. On note $P(r_i)$ la probabilité que la chaîne latérale à la position i soit dans le rotamère r_i . On a :

$$P(r_i) = \frac{\exp(-\beta E(r_i))}{\sum_{l_i=1}^{N_i} \exp(-\beta E(l_i))} \quad (1.26)$$

avec N_i le nombre total de rotamères à la position i , et $\beta = \frac{1}{kT}$, k étant la constante de Boltzmann et T la température.

Si on se limite au cas où une énergie d'interaction pour un résidu est la somme des interactions avec les autres résidus de la chaîne (énergie décomposable par paires), alors l'énergie d'interaction moyenne en i est la somme des interactions impliquant le rotamère r_i pondéré par le poids de Boltzmann de l'autre rotamère, c'est à dire :

$$E(r_i) = \sum_{i \neq j} \sum_{l_j}^{N_j} E(r_i, l_j) P(l_j) \quad (1.27)$$

avec l_j parcourant tous les rotamères aux positions autres que i .

Les formules 1.26 et 1.27, constituent un système itératif. Ainsi, l'algorithme se déroule de la façon suivante :

1. À chaque position, les rotamères sont équiprobables.
2. Les énergies moyennes sont calculées grâce à la formule 1.27.
3. De nouvelles probabilités de Boltzmann sont calculées à partir des énergies moyennes précédentes et la formule 1.26.
4. Retour à l'étape 2 jusqu'à convergence des énergies.

Cette méthode garantit la convergence vers un ensemble de rotamères stables à chaque position placée dans son « environnement moyen ». Le temps de calcul avec cet algorithme augmente de façon linéaire avec le nombre de résidus de la protéine, ce qui en fait un des algorithmes les plus rapides.

1.4.2 Le Dead-End Elimination

Le « Dead End Elimination » (DEE) consiste à éliminer des rotamères ou des combinaisons de rotamères qui ne peuvent pas faire partie de la conformation qui minimise l'énergie ou GMEC. Comme pour le champ moyen, l'énergie doit être décomposable par paires. Il y a deux critères d'élimination [33] :

Le critère simple : le rotamère r_i du résidu i est éliminé si la meilleure énergie (la plus faible) qu'il est possible d'obtenir pour une conformation comprenant ce rotamère est moins bonne que la pire énergie obtenue avec un rotamère r'_i à la même position. Notons $E(c)$, l'énergie d'une séquence-conformation C . Une expression mathématique de ce critère peut être le suivant :

Si $\min_{C; r_i \in C} E(C) > \max_{C; r'_i \in C} E(C)$ alors r_i est éliminé.

En utilisant l'expression d'une fonction d'énergie décomposée par paires (équation 1.25), avec N le nombre de positions, le critère peut s'écrire de la façon suivante :

Si $E_i(r_i) + \sum_{j \neq i}^N \min_{r_j} E_{ij}(r_i, r_j) > E_i(r'_i) + \sum_{j \neq i}^N \max_{r_j} E_{ij}(r'_i, r_j)$ alors r_i est éliminé.

Le critère double : il exprime une condition analogue sur un couple de rotamères (r_i, r_j) suffisante pour qu'il ne puisse pas faire partie du GMEC. Pour son expression on introduit l'énergie d'une paire r_i, r_j : $E^{r_i, r_j} = E_i(r_i) + E_j(r_j) + E_{ij}(r_i, r_j)$. Il s'exprime ainsi comme suit :

Si

$E^{r_i, r_j} + \sum_{k=1}^N \min_{r_k} (E_{ik}(r_i, r_k) + E_{jk}(r_j, r_k)) > E^{r'_i, r'_j} + \sum_{k=1}^N \max_{r_k} (E_{ik}(r'_i, r_k) + E_{jk}(r'_j, r_k))$
alors la paire r_i, r_j est éliminée.

L'élimination d'un couple (r_i, r_j) , n'exclut pas pour autant la présence de r_i ou de r_j dans la solution optimale.

Le critère de Goldstein [34] : il s'agit d'une amélioration du critère simple du DEE, voir la figure 1.6. Il peut exister des situations dans lesquelles l'énergie avec un rotamère r_i est toujours diminuée en la remplaçant par un autre rotamère r'_i . On peut donc l'éliminer. Or, cela n'implique pas forcément la condition du critère simple. Ce nouveau critère peut s'exprimer de la façon suivante :

S'il existe $r_{i'}$ tel que $E_i^{r_i} - E_i^{r_{i'}} + \sum_{k=1}^N \min_{r_k}(E_{ik}(r_i, r_k) - E_{i'k}(r'_i, r_k)) > 0$ alors r_i est éliminé.

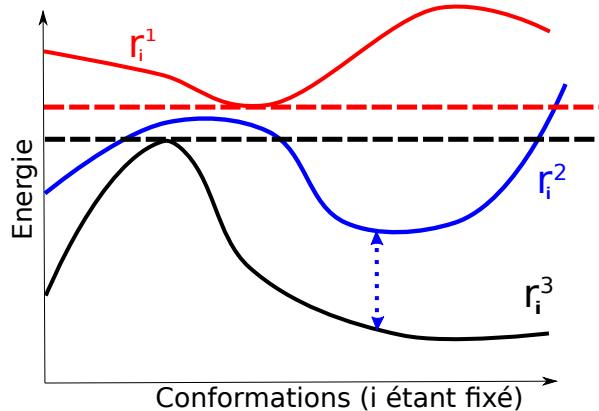


Figure 1.6 – **DEE, graphes de la fonction d'énergie pour des rotamères fixés à la position i** Le critère simple permet l'élimination du rotamère r_i^1 parce que l'énergie des conformations qui le contiennent, l'énergie rouge, est toujours moins bonne que la noire. Mais seul le critère de Goldstein permet d'éliminer r_i^2 parce que l'écart entre le graphe bleu et le graphe noir est positif pour chaque conformation.

Au cours de l'optimisation, les deux critères peuvent être utilisés alternativement, jusqu'à ce qu'il n'y ait plus de rotamères à éliminer. Il est alors possible d'utiliser une approche exhaustive sur l'espace d'états réduit pour obtenir le GMEC. Cette méthode permet dans les bons cas (par exemple pour les petits systèmes) de converger en un temps raisonnable [35]. Beaucoup de variantes du DEE [36, 37] ont été proposées notamment en travaillant sur des combinaisons de plus de deux rotamères.

1.4.3 Le CFN ou « Cost Function Network »

La méthode des réseaux de fonctions de coût est issue du domaine de l'optimisation combinatoire. Elle est utilisable si la fonction d'énergie est décomposable par paires. Il s'agit avant tout d'une recherche de la séquence-conformation qui minimise l'énergie globale ou GMEC. Dans certaines conditions, elle peut également fournir un ensemble de séquences-conformations proches du GMEC.

Un réseau de fonction de coût, ou « cost function network » (CFN), est constitué d'un ensemble de variables \mathcal{X} et d'un ensemble \mathcal{C} de fonctions des variables \mathcal{X} (les fonctions de coûts). Ici, nous considérons uniquement des CFN avec des fonctions d'au plus deux variables (constantes, unaires ou binaires). Un problème CPD est représenté par un CFN de la façon suivante. À chaque résidu i de la protéine on associe une variable v_i ayant comme valeurs possibles les rotamères de i . À chaque variable v_i , on définit une fonction unaire

$C_1(v_i)$ représentant l'énergie « self » en i , $C_1(r_i) = E_i(r_i)$. À chaque couple de positions i, j , on définit une fonction binaire $C_2(v_i, v_j)$ représentant les énergies d'interactions possibles entre i et j , $C_2(r_i, r_j) = E_{ij}(r_i, r_j)$. Alors, une séquence-conformation correspond à une valeur pour chaque variable de \mathcal{X} ; on parle de solution du CFN. La recherche du minimum global d'énergie revient ainsi à trouver une solution qui minimise la somme de toutes les fonctions de coût. Pour travailler avec des fonctions de coûts à valeurs entières positives, on multiplie toutes les valeurs par la précision des énergies et on translate les valeurs vers les nombres positifs.

L'algorithme « Depth-First Branch and Bound » (DFBB)

La résolution d'un CFN est tentée généralement par un algorithme de type « Depth-First Branch and Bound » (DFBB). Les ingrédients sont les suivants :

Un principe de séparation : il est utilisé pour la construction d'un arbre qui organise les solutions. Un ensemble de solutions peut être vu comme le premier sommet S_0 d'une arborescence en construction. La séparation est l'action de partager, selon certains critères, cet ensemble en sous-ensembles qui deviennent les fils de S_0 ; ce partage doit constituer une partition de l'ensemble des solutions du départ. Le critère de séparation classique est d'énumérer les valeurs possibles d'une variable v_i du CFN. À chacune des valeurs x possibles de v_i on définit le sous-ensemble de S_0 ayant dans toutes ses solutions $v_i = x$. Ainsi, si v_i à N valeurs possibles, N fils de S_0 sont créés.

Un majorant : le majorant du problème correspond au meilleur coût connu ($+\infty$ au début du calcul). Il majore le GMEC.

Un minorant d'un sommet : le minorant correspond à une valeur que l'on sait être inférieure ou égale au coût de toutes les solutions d'un sommet.

Un principe d'évaluation : évaluer un sommet S , c'est déterminer un de ses minorants. Si un sommet est évalué et est supérieur au majorant courant, on sait qu'aucune solution de S ne peut être le GMEC. La totalité du sous-arbre peut être élaguée, c'est-à-dire exclue de l'optimisation.

Une stratégie de développement : elle consiste à choisir une méthode de développement de l'arbre des solutions; c'est déterminer l'ordre sur les sommets de l'arborescence dans lequel on va appliquer le critère de séparation. Dans le DFBB, la stratégie consiste à descendre dans les branches jusqu'à trouver un sous-arbre qu'il est possible d'élaguer, alors l'algorithme remonte d'une branche pour redescendre dans une autre direction. Ce parcours en profondeur à l'avantage de limiter l'utilisation de la mémoire, parce qu'il n'est nécessaire de conserver que la description de la branche qui a été explorée. Un exemple est présenté à la figure 1.7. Il devient alors nécessaire de trouver de bons minorants.

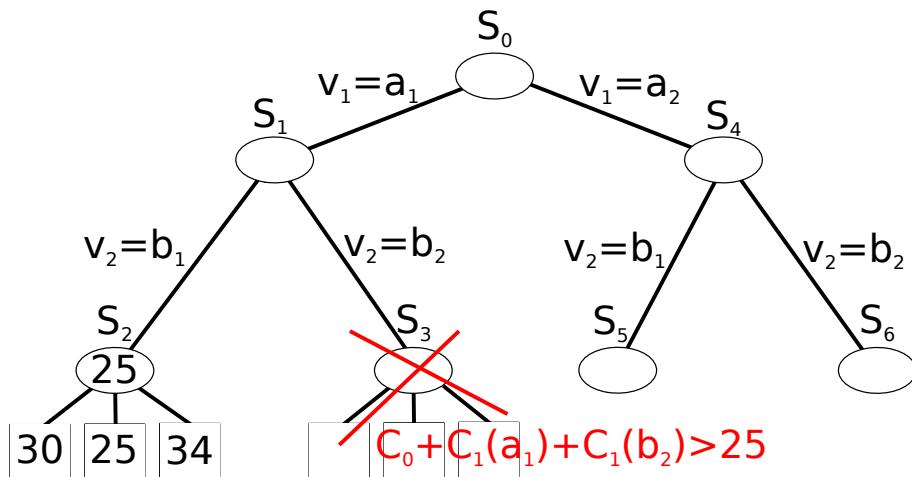


Figure 1.7 – **Principe de l’algorithme du Depth-First Branch and Bound** Les solutions d’un CFN sont représentées sous forme d’un arbre dans lequel le sommet S_0 représente l’ensemble des solutions et les fils S_1 et S_4 une partition de S_0 , avec pour chacun une première variable v_1 fixée. Le DFBB, descend jusqu’au sommet S_2 qu’il peut évaluer : un minorant de S_2 est 25. 25 devient le nouveau majorant du GMEC. L’algorithme remonte et redescend vers S_3 . Ici le coût constant plus le coût des affectations de v_1 et v_2 est supérieur au majorant : On peut élaguer l’arbre au sommet S_3 .

Equivalence Preserving Transformation

Une EPT, ou « Equivalence Preserving Transformation », transforme un CFN A en un CFN B tel que pour chaque solution (chaque affectation des variables), le total des coûts est identique. A et B sont dits « en cohérence locale ». Le principe est de déplacer les coûts entre fonctions pour les attribuer aux fonctions les plus simples, qui portent sur une seule variable au plus (les coûts unitaires et constants). L’objectif est de faire apparaître de bons minorants pour le DFBB [38]. L’avantage ici, est de pouvoir conserver les transformations dans un chemin de l’arbre de recherche tout au long de l’optimisation. Il existe deux types de transformations :

- la première appelée projection consiste à transférer un coût de l’ensemble des coûts unaires vers le coût constant, des coûts binaires vers le coût constant ou encore des coûts binaires vers les coûts unaires.
- le second type de transformation fait l’inverse, c’est à dire transfère un coût d’une fonction unaire vers les fonctions binaires impliquant la même valeur de variable. Une telle transformation permet une augmentation du transfert total vers le coût constant.

Un exemple inspiré de la thèse de Seydou Traoré est présenté à la figure 1.8.

Cette approche a montré sa supériorité, avec l’outil Toulbar2 [39, 40], par rapport à un ensemble de résolveurs de CFN parmi les plus réputés actuellement qui exploitent notamment l’approche DEE/A* ou le backtracking. Toulbar2 en combinant FFBB, EPT et DEE a été capable de trouver le GMEC dans le plus grand nombre de problèmes issus du CPD proposés aux différents outils.

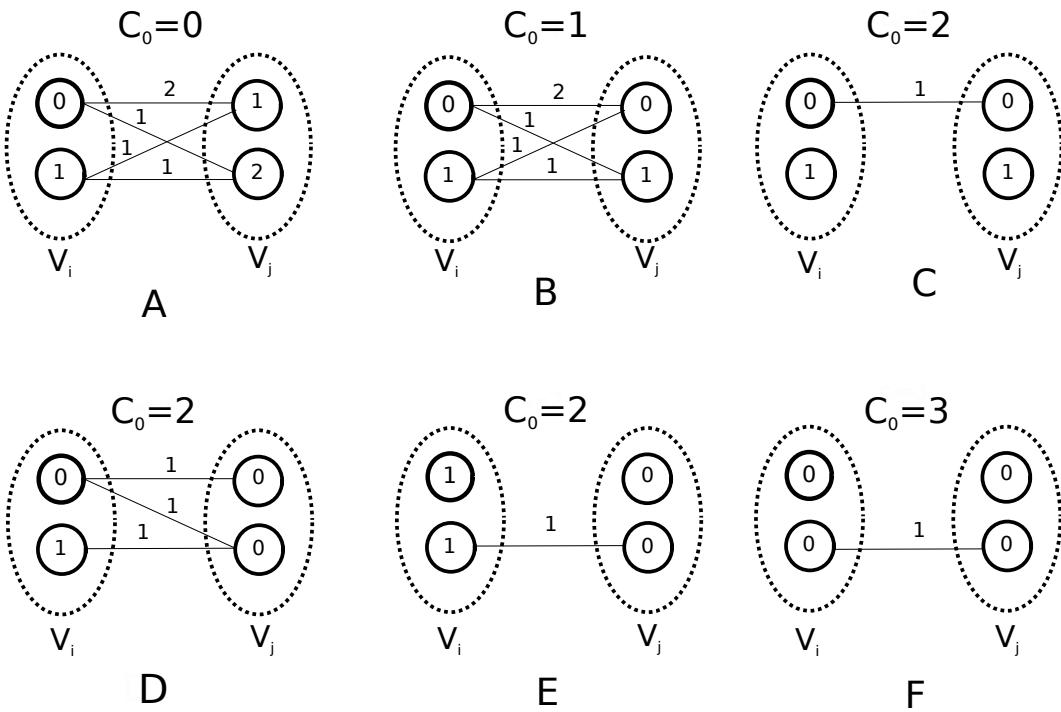


Figure 1.8 – **Exemple de transformations EPT sur un CFN composé de 2 variables v_i et v_j ayant deux valeurs possibles.** La transformation de *A* vers *B* est une projection de v_j vers le coût constant C_0 . Puis une projection des fonctions binaires mène en *C*. La seconde valeur de v_j est distribuée sur les arcs en *D*. Cela permet deux nouvelles projections qui mènent à *E* puis à *F*. *A* et *F* sont en cohérence locale.

1.4.4 L'heuristique Multistart Steepest Descent (MSD)

En 2000, Wernisch, Hery et Wodak [41] partent du constat que l'espace conformationnel et les fonctions d'énergie utilisées ne capturent que partiellement la réalité. Ainsi, le GMEC ne correspondra pas forcément à la séquence la plus stable. D'autre part, des protéines avec des taux d'identité de moins de 50% peuvent conserver quasiment la même structure 3D. Cela révèle l'immensité du nombre de séquences-conformations propre à un pli. Ainsi, leur objectif n'est pas de résoudre un problème d'optimisation, mais d'exhiber un ensemble de séquences de basse énergie. Ils proposent alors une heuristique conçue pour le CPD. Il s'agit d'une procédure simple qui produit une grande quantité de séquences-conformations de basse énergie, sans se focaliser sur le GMEC.

Un cycle heuristique se déroule de la façon suivante. Au départ, une séquence-conformation est construite en attribuant de façon aléatoire un type d'acide aminé et un rotamère à chaque position de la chaîne. Ensuite, en parcourant la séquence du début jusqu'à la fin, l'algorithme procède par des ajustements successifs à chaque position. Pour chaque position i de la séquence, tous les états possibles sont évalués, le reste de la séquence et des rotamères étant fixés. Le meilleur rotamère est alors attribué en i , puis on passe à la position suivante. La séquence est ainsi ajustée par plusieurs passages successifs, jusqu'à convergence de l'énergie, voir l'algorithme 1. Un cycle est très rapide, ce qui permet de produire dans les cas usuels, plusieurs milliers de séquences par heure. L'utilisation

mémoire hormis le stockage des énergies est quasi nulle. Il n'impose pas que la fonction d'énergie soit décomposable par paires, mais il s'en accorde très bien, en rendant possible l'utilisation de la mémoire par bloc pour l'ensemble des énergies impliquant une position i donnée.

```
1 pour chaque cycle heuristique faire
2   choisir une séquence-conformation  $C$  aléatoirement ;
3   tant que l'énergie de  $C$  est améliorée faire
4     pour  $i$  allant de la première position de  $C$  jusqu'à la dernière faire
5       fixer  $C$  sauf à la position  $i$  ;
6       déterminer le meilleur rotamère possible en  $i$  ;
7       fixer  $C$  en  $i$  avec ce rotamère ;
8   sauvegarder  $C$  ;
```

Algorithme 1 : L'algorithme Multistart Steepest Descent

1.4.5 L'algorithme génétique

Holland et ces collaborateurs [42] introduisent une nouvelle approche inspirée des principes biologiques de la sélection naturelle, avec des opérations comme des mutations, des croisements et la sélection. L'algorithme génétique a pour objectif d'obtenir un ensemble de solutions proches de l'optimum en un temps raisonnable. Le schéma général est le suivant. Une population de séquences-conformations est générée de façon aléatoire. L'énergie de tous les membres de la population est évaluée. Une sélection basée sur l'énergie est appliquée, ce qui diminue la taille de la population. Des mutations aléatoires et des croisements sont appliqués à la nouvelle population, qui augmente. Enfin, une condition d'arrêt est évaluée. Si elle n'est pas réalisée, l'algorithme retourne à l'étape d'évaluation (voir la figure 1.9).

L'individu de meilleure énergie au sein de la population tend à se reproduire le plus vite. Donc la valeur moyenne de l'énergie de l'ensemble des séquences-conformations converge. On peut voir ici que le nombre de membres de la population est un paramètre de l'algorithme. Plus ce nombre est faible plus la convergence va être rapide. A contrario, plus ce nombre est grand, meilleure est l'exploration de l'espace d'états. Les atouts de l'algorithme génétique sont sa capacité à franchir des barrières énergétiques par des changements de séquences rapides via des croisements et sa capacité à optimiser en parallèle différents secteurs de la structure.

1.4.6 Le Monte Carlo

Dans son acceptation la plus générale, un algorithme Monte Carlo est un algorithme stochastique (il utilise une source de hasard) qui approche probablement la solution exacte en un temps d'exécution déterminable a priori. Cela le distingue, parmi les algorithmes stochastiques, d'un algorithme de Las Vegas qui donne un résultat exact dans un temps d'exécution non déterministe, et d'un algorithme d'Atlantic City qui donne des résultats

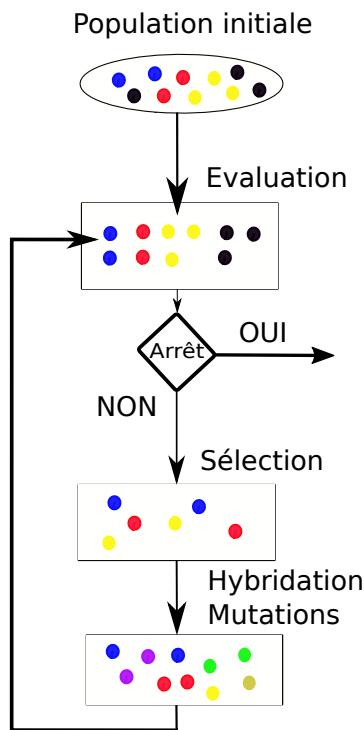


Figure 1.9 – L'algorithme génétique

probablement corrects dans un temps probablement rapide. Parmi les algorithmes Monte Carlo, les algorithmes Monte Carlo par Chaînes de Markov (MCMC) ont été particulièrement étudiés. Ce ne sont pas des algorithmes d'optimisation, mais des algorithmes d'échantillonnage d'une distribution de probabilité π . On veut générer des éléments x_i de l'espace des phases tels que la distribution $D = \{x_i, i \in \mathbb{N}\}$ converge vers π (dans un sens mathématique à préciser).

Il existe plusieurs méthodes pour répondre à cette question, mais les MCMC sont bien adaptés aux situations où l'espace des états est de grande dimension. Un autre avantage des algorithmes MCMC est qu'elles ne nécessitent pas la connaissance de la constante de normalisation de π . Cela constitue un attrait important parce que dans beaucoup de situations cette constante de normalisation est particulièrement difficile à calculer. Pour ces raisons, l'échantillonnage par MCMC est très populaire depuis plus de cinquante ans. Dans la suite, nous donnons des conditions suffisantes pour que ce type l'algorithme converge dans le cadre d'un espace d'états E fini, puis nous détaillons le plus célèbre d'entre eux, l'algorithme de Metropolis-Hastings. L'idée est de générer un élément x_i en utilisant dans les états déjà visités uniquement x_{i-1} . Ce type de processus, où la mémoire utilisée est réduite à l'état précédent caractérise les processus de Markov.

On appelle processus stochastique, une suite de variables aléatoires $\{X_n\}$ à valeurs dans E . Une chaîne de Markov est un processus stochastique tel que :

$$\forall n \geq 0, \forall (x_0, \dots, x_{n-1}, x) \in E^{n+1}, P(X_n = x | X_{n-1} = x_0, \dots, X_0 = x_{n-1}) = P(X_n = x | X_{n-1} = x_0)$$

Une chaîne de Markov est homogène si :

$$\forall n \geq 0, \forall x_i, x_j \in E, P(X_n = x_i | X_{n-1} = x_j) = P(X_{n+1} = x_i | X_n = x_j) \quad (1.28)$$

C'est-à-dire que la probabilité de transition P est indépendante de n . Dans la suite, on ne considère que des chaînes de Markov homogènes et on note :

$$P(X_n = a | X_{n-1} = b) = P(a|b)$$

Pour simplifier les notations, on introduit une matrice de transition P avec $P_{ab} = P(X_1 = b | X_0 = a)$ et on écrit $\mu_0 = (P(X_0 = a))_{a \in E}$ et $\mu_1 = (P(X_1 = a))_{a \in E}$. On a alors :

$$\mu_1 = P\mu_0$$

et plus généralement

$$\mu_n = P\mu_{n-1}$$

Le principe de la balance détaillée

Le principe de la balance détaillée est un principe général de comportement des systèmes dynamiques. Il a été utilisé par Boltzmann pour la construction de la physique statistique et Cercignani et Lampis ont montré qu'il était vrai pour les gaz polyatomiques [43]. Il peut s'exprimer de la façon suivante. Pour un système dynamique \mathcal{S} à l'équilibre, a et b deux éléments de l'espace des phases associé à \mathcal{S} , la probabilité de transition de a vers b est égale à la probabilité de transition de b vers a , ou encore :

$$\forall a, b \in \mathcal{S}, \mathcal{P}(a \rightarrow b) = \mathcal{P}(b \rightarrow a) \quad (1.29)$$

Pour pouvoir appliquer ce principe aux chaînes de Markov, introduisons une définition proche. Soit q une probabilité sur E . Une chaîne de Markov est dite réversible par rapport à q si

$$\forall a, b \in E, q(b)P(a|b) = q(a)P(b|a) \quad (1.30)$$

On a alors :

$$\forall a \in E, (Pq)(a) = \sum_{b \in E} q(b)P(a|b) = \sum_{b \in E} q(a)P(b|a) = q(a) \sum_{b \in E} P(b|a) = q(a) \quad (1.31)$$

Ainsi, la probabilité q est inchangée après la transition P , la chaîne est dite stationnaire pour q .

Il nous faut alors construire une chaîne de Markov réversible pour la distribution cible π . Mais cela ne suffit pas. En effet, même si $\{X_n\}$ est stationnaire pour π , rien ne dit que la chaîne va, à un moment, distribuer les états comme π . Se pose la question de la

convergence vers π . Or, si E est discret, une chaîne de Markov réversible est dite ergodique si et seulement si :

- Pour tous a et b de l'espace d'états il existe un chemin de a vers b de probabilité non nulle.
- Il n'existe pas de a de E tel que la chaîne revienne en a périodiquement.
- Au cours du temps, tous les états sont visités par la chaîne avec une probabilité non nulle.

On a alors le résultat suivant : une chaîne ergodique converge vers son unique distribution stationnaire.

L'algorithme de Metropolis

Nous pouvons maintenant décrire l'algorithme de Metropolis-Hastings. L'idée initiale de Metropolis est de construire un algorithme qui échantillonne la distribution de Boltzmann. Elle donne la probabilité d'un état x_i du système en fonction de son énergie et de la température :

$$\pi(x_i) = \frac{\exp(-\beta E_{x_i})}{\sum_{x_j \in E} \exp(-\beta E_{x_j})} \quad (1.32)$$

avec $\beta = \frac{1}{kT}$, E_{x_i} l'énergie de x_i , T la température et k la constante de Boltzmann. La constante de normalisation de la probabilité s'appelle la fonction de partition du système. Elle est particulièrement difficile à calculer.

On définit alors la probabilité de transition P comme le produit de deux probabilités :

$$P(x_j|x_i) = sel(x_j|x_i)acc(x_j|x_i) \quad (1.33)$$

avec sel une probabilité de sélectionner l'état x_j de E lorsque le système est dans l'état x_i et acc la probabilité d'accepter l'état x_j étant en x_i . Si la chaîne respecte le principe de la balance détaillée par rapport à π , on a :

$$\pi(x_i)sel(x_j|x_i)acc(x_j|x_i) = \pi(x_j)sel(x_i|x_j)acc(x_i|x_j) \quad (1.34)$$

Si on se limite à une probabilité de sélection symétrique :

$$\pi(x_i)acc(x_j|x_i) = \pi(x_j)acc(x_i|x_j) \quad (1.35)$$

ce qui équivaut à

$$\frac{acc(x_j|x_i)}{acc(x_i|x_j)} = \frac{\pi(x_j)}{\pi(x_i)} = \frac{\exp(-\beta E_{x_j})}{\exp(-\beta E_{x_i})} = \exp(-\beta \Delta E) \quad (1.36)$$

avec $\Delta E = E_{x_j} - E_{x_i}$. Ainsi, le calcul de la fonction de partition est évité ! Métropolis propose alors la probabilité d'acceptation suivante :

$$acc(x_i|x_j) = \exp(-\beta\Delta E) \text{ si } \Delta E > 0; acc(x_i|x_j) = 1 \text{ sinon} \quad (1.37)$$

On a bien :

$$\forall x_i, x_j \in E, \frac{acc(x_j|x_i)}{acc(x_i|x_j)} = \exp(\beta\Delta E) \quad (1.38)$$

Il suffit alors de choisir une probabilité de sélection symétrique ne violant pas les conditions ergodiques pour obtenir notre convergence. Par la suite, Hastings généralise l'algorithme à une probabilité *sel()* non symétrique en choisissant *acc()* telle que :

$$acc(x_i|x_j) = \exp(-\beta\Delta E) \frac{sel(x_j|x_i)}{sel(x_i|x_j)} \text{ si } \Delta E > 0; acc(x_i|x_j) = 1 \text{ sinon} \quad (1.39)$$

Si l'on injecte cette nouvelle probabilité dans l'équation 1.34, on a encore le respect de la balance détaillée et la convergence. La séquence d'instructions est résumée par l'algorithme 2. Dans toute la suite, l'algorithme Monte Carlo (MC) désigne l'algorithme de Metropolis-Hastings, ce qui est un usage courant.

```

1 Une séquence-conformation  $S_0$  est choisie aléatoirement;
2 pour chacun des pas  $i$  de la trajectoire faire
3   choisir une proposition  $S'_i$  à partir d'une probabilité conditionnelle  $sel(.,S)$ ;
4   calculer une probabilité d'acceptation  $acc$ ; si  $acc \geq 1$  alors
5      $S_{i+1} = S'_i$ ;
6     sauvegarder  $S_{i+1}$ ;
7   sinon
8     alors  $S_{i+1} = S'_i$  avec une probabilité  $acc$ ;
9     sauvegarder  $S_{i+1}$ ;
10    sinon
11       $S_{i+1} = S_i$ 
```

Algorithme 2 : L'algorithme de Metropolis

1.4.7 Le Monte Carlo avec échanges de répliques (REMC)

Une amélioration de l'algorithme de Métropolis-Hastings, connu sous le nom de « Replica Exchange Monte Carlo » a été introduite par Swendsen and Wang [44]. La méthode est également connue sous le nom « parallel tempering ». L'objectif est d'accélérer la convergence en permettant au processus stochastique de visiter plusieurs zones énergétiques simultanément. Ainsi, l'algorithme couple l'exploration des bassins de basses énergies proches de l'optimum, avec l'exploration des zones de plus hautes énergies, ce qui facilite la sortie des minimums locaux.

On considère N simulations MC du système à N températures. Ces répliques du système sont indépendantes les unes par rapport aux autres. On note T_m avec $m = 1, \dots, N$ les températures. Toutes ces températures sont différentes et il y a toujours une réplique à chaque température. Ainsi, nous nous plaçons dans un ensemble généralisé noté E^N , constitué des N -uplets (x_1, \dots, x_n) avec les x_i éléments de E , et $1 \leq i \leq N$ indexant les répliques. On travaille avec une chaîne de Markov $\{X(t), t \in \mathbb{N}\}$, avec $X(t) = (x_1(t), \dots, x_n(t))$ une variable aléatoire à valeur dans E^N au temps t . On ajoute maintenant un nouveau type de déplacement, qui consiste à intervertir au temps t l'état x_i de la simulation à la température T_i avec l'état x_{i+1} à la température T_{i+1} . On a alors :

$$\begin{cases} x_{i+1}(t+1) = x_i(t) \\ x_i(t+1) = x_{i+1}(t) \end{cases} \quad (1.40)$$

L'algorithme consiste alors à effectuer de façon itérative :

- un ensemble de k déplacements de type MC, avec k une constante
- une tentative de déplacement de type 1.40

Pour que le processus respecte la balance détaillée, le nouveau déplacement doit respecter certaines conditions. Nous introduisons une probabilité d'acceptation acc_{swap} spécifique. Pour la déterminer, nous reprenons la même démarche que pour le Monte Carlo simple. Comme les répliques sont sans interactions, la distribution cible de la chaîne de Markov est égale aux produits des distributions cibles aux différentes températures :

$$\pi(X) = \frac{1}{Z} \exp\left(-\sum_{i=1}^N \frac{E_{x_i}}{kT_i}\right) \quad (1.41)$$

En injectant cette expression de π et les équations 1.40 dans 1.35, on obtient :

$$\frac{acc_{swap}(X(t+1)|X(t))}{acc_{swap}(X(t)|X(t+1))} = \frac{\pi(X(t))}{\pi(X(t+1))} = \frac{\exp(-E_{x_i}/kT_i) \exp(-E_{x_{i+1}}/kT_{i+1})}{\exp(-E_{x_{i+1}}/kT_i) \exp(-E_{x_i}/kT_{i+1})} = \exp(-\Delta)$$

avec $\Delta = (\frac{1}{kT_i} - \frac{1}{kT_{i+1}})(E_{x_i} - E_{x_{i+1}})$.

Ceci peut être satisfait par le critère de Metropolis :

$$acc_{swap}(X(t+1)|X(t)) = \exp(-\Delta) \text{ si } \Delta > 0; acc_{swap}(X(t+1)|X(t)) = 1 \text{ sinon}$$

Le REMC peut alors se décrire comme l'algorithme 3.

Il reste au simulateur à adapter le nombre de répliques, les températures utilisées et la fréquence des tentatives d'échange à sa problématique. La capacité de REMC à être exécuté sur des machines parallèles a entraîné sa grande popularité, en particulier dans le domaine de la modélisation moléculaire, mais aussi en physique, chimie, intelligence artificielle.

```
1 Lancement en parallèle de  $N$  marcheurs Monte Carlo aux températures ordonnées  
    ( $t_1, \dots, t_n$ ) ;  
2 pour les pas  $p$  multiples d'une constante  $P$  faire  
3     choisir aléatoirement  $i$  compris entre 1 et  $N - 1$ , ce qui sélectionne les marcheurs  
    aux températures  $t_i$  et  $t_{i+1}$  ;  
4     la probabilité d'acceptation  $\text{acc}_{\text{swap}}$  est calculée si  $\text{acc}_{\text{swap}} \geq 1$  alors  
5         Les marcheurs échangent leur température ;  
6         sinon  
7             Les marcheurs échangent leur température , avec la probabilité  $\text{acc}_{\text{swap}}$ 
```

Algorithme 3 : L'algorithme REMC

Chapitre 2

Proteus et nos outils d'analyse

Il existe plusieurs logiciels de CPD. Parmi les plus connus, on peut citer ORBIT (Optimisation of Rotamers By Iterative Techniques) [16], OSPREY (Open Source Protein REdesign for You) [45], Rosetta (le CPD n'est qu'une partie des fonctionnalités proposées par cette suite logicielle) [12] et Proteus [30, 46]. Proteus est le logiciel développé par notre équipe au laboratoire de Biochimie de l'École Polytechnique. Dans ce chapitre, nous détaillons quelques points importants de notre logiciel.

2.1 Un modèle fondé sur la Physique

Proteus se base sur la théorie de la mécanique statistique pour formaliser les problèmes auxquels il s'attaque et pour guider sa sélection de meilleures séquences-conformations. À partir d'un postulat fondamental et de l'hypothèse ergodique, la mécanique statistique « redécouvre » la thermodynamique classique et en plus, établit une relation à l'équilibre entre l'énergie d'un état i d'un système et la probabilité que le système soit en i par la probabilité de Boltzmann, voir l'équation 1.32 page 23. Ainsi, si l'on considère deux états A et B d'un système S à l'équilibre, le ratio des probabilités que S soit dans l'état A ou dans l'état B s'exprime en fonction de la différence entre l'énergie de A et de B. Pour le CPD, l'énergie qui est pertinente est celle qui prend en compte l'énergie interne de la protéine, mais aussi son environnement aqueux moyen. Il s'agit donc d'une énergie libre de Gibbs G . De plus, on prend la différence entre états replié et déplié.

Nous introduisons alors un cycle thermodynamique pour définir la stabilité d'une séquence-conformation, voir figure 2.1. Le cycle considère la stabilité de deux séquences A et B. La transformation de A en B correspond aux deux flèches horizontales. Le dépliement est figuré par les deux flèches verticales. La différence de stabilité entre les deux séquences correspond à la différence d'énergie libre des transformations horizontales (comme verticales). On a alors la différence :

$$\Delta\Delta G = (G(S_B^P) - G(S_A^P)) - (G(S_B^D) - G(S_A^D)) \quad (2.1)$$

avec respectivement S_A^P , S_B^P , S_B^D et S_A^D , le système avec la séquence A repliée, B repliée, A dépliée et B dépliée. Maximiser la stabilité d'une séquence-conformation B correspond alors à minimiser $\Delta\Delta G$.

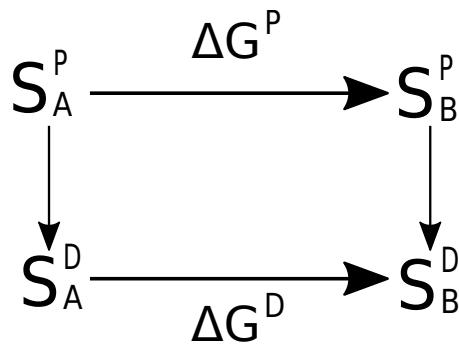


Figure 2.1 – Cycle thermodynamique qui définit la stabilité relative de deux séquences-conformations S_A et S_B .

Nous faisons des simulations avec deux mouvements élémentaires possibles : un changement de rotamère dans la séquence repliée courante et une mutation qui consiste à modifier le type de la chaîne latérale à une position i tirée au hasard. Le changement d'énergie ΔE_{m_1} associé à un changement de rotamère r_i vers r'_i est :

$$\Delta E_{m_1} = E^f(\dots, t_i, r'_i, \dots) - E^f(\dots, t_i, r_i, \dots) \quad (2.2)$$

avec E^f l'énergie de l'état replié. Pour le changement d'énergie ΔE_{m_2} d'une mutation de t_i vers t'_i , nous appliquons l'équation 2.1 à la séquence-conformation avant et après la mutation :

$$\Delta E_{m_2} = (E^f(\dots, t'_i, r'_i, \dots) - E^f(\dots, t_i, r_i, \dots)) - (E^u(t'_i) - E^u(t_i)) \quad (2.3)$$

avec E^f l'énergie de la l'état replié et E^u l'énergie de l'état déplié. Cela peut s'interpréter comme le changement à la position i de t_i vers t'_i dans l'état replié et du changement inverse t'_i vers t_i dans l'état déplié. Comme nos simulations Monte Carlo respectent le schéma de Métropolis, les séquences-conformations sont visitées suivant leur probabilité de Boltzmann. Ainsi le ratio des populations de deux séquences-conformations obtenues est fonction de leur stabilité relative.

2.2 Organisation générale

Proteus est constitué de trois parties :

- une version modifiée de Xplor, qui fournit plusieurs modèles de solvant et plusieurs fonctionnalités spécifiques au CPD. Xplor est un programme de simulation moléculaire [47], disponible en téléchargement sur le site de l'université de Yale. Les modifications CPD sont disponibles sur notre site Web.
- un ensemble de scripts Xplor, qui pilote le calcul de la matrice d'énergie.
- un programme écrit en C, que nous appelons « proteus » qui explore l'espace des séquences-conformations

Proteus est flexible et largement configurable. Il permet l'utilisation de plusieurs champs de force, de plusieurs modèles de solvant, et de plusieurs librairies de rotamères. La partie en C, proteus, propose plusieurs algorithmes d'exploration comme le MSD, MC ou REMC. Le programme proteus permet de diviser le système en plusieurs groupes ou d'en dupliquer une partie. Ceux-ci peuvent alors être combinés dans une fonction de score basée sur la fonction d'énergie physique afin de favoriser la stabilité de certains sous-ensembles du système ou certaines affinités. Plusieurs succès ont déjà été obtenus avec Proteus par exemple le redesign de la tyrosyl-ARNt synthétase [48], les calculs de pK_a [49] ou la création de nouveaux domaines PDZ [50].

Proteus autorise le choix de plusieurs fonctions d'énergie. La plus simple « MMCASA » combine la mécanique moléculaire pour la protéine et un modèle de solvant implicite CASA (voir 1.3.3 page 9). Deux types « MMGBSA » sont également possibles, dans lesquels le traitement du solvant est effectué par un modèle de Born Généralisé. Nous détaillons les points importants de Proteus dans la suite.

2.3 Décomposition par paires de la fonction d'énergie

Une fonction d'énergie décomposable par paires permet de réduire le calcul de l'énergie d'une séquence-conformation à une somme d'énergies précalculées. Cependant, les termes GB et SA ne sont pas rigoureusement décomposables par paires. Il faut alors introduire de nouvelles approximations pour permettre cette décomposition. Voyons dans la suite comment ces problèmes sont résolus dans Proteus.

2.3.1 Décomposition du terme de surface

Le terme surfacique présenté en 1.12 page 9, se définit comme :

$$E_{solv}^{surf} = \sum_i \sigma_{t_i} A_i \quad (2.4)$$

avec A_i la surface accessible au solvant de l'atome i , et σ_{t_i} un coefficient d'hydrophobicité de l'atome i . Ce terme n'est pas décomposable par paires de résidus, parce que la surface d'une première chaîne latérale enfouie par une deuxième peut aussi être enfouie par une troisième, voir la figure 2.2.

Pour rendre ce terme décomposable, nous utilisons la méthode de Street et al. [51] dans laquelle la surface enfouie d'une chaîne latérale est calculée à partir d'une somme sur les groupes voisins. Puis pour chaque groupe voisin, la surface de contact avec notre chaîne est calculée indépendamment des autres groupes. Ces surfaces de contact sont sommées et un facteur de réduction est appliqué mimant l'élimination des doubles comptages. Des travaux précédents effectués dans notre laboratoire ont montré qu'un facteur de 0,65 fonctionne bien [52, 31].

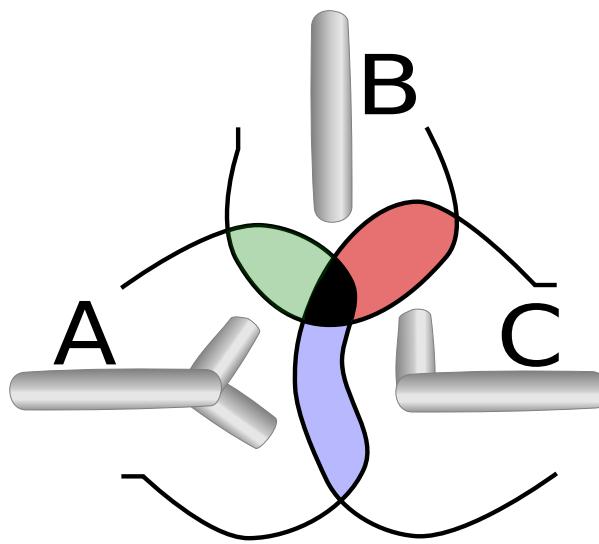


Figure 2.2 – Une représentation de la surface accessible de trois résidus. Les résidus A et B réduisent mutuellement leur surface exposée au solvant : c'est la zone verte. De même pour B, C : la zone rouge et A, C, la zone bleue. Un calcul par paires de résidus naïf surestime la surface accessible en comptant deux fois la zone noire.

2.3.2 « Native Environnement Approximation » (NEA)

Dans le terme GB de l'énergie de solvatation, le rayon de solvatation b_i approxime la distance de l'atome i à la surface de la protéine. C'est une fonction de la position de tous les atomes de la protéine. Pour que ce rayon soit décomposable par paire, Proteus implémente l'approximation NEA qui est présentée à la section 1.3.5 page 13. Dans cette approche, le rayon de solvatation b_i de chaque groupe (backbone, chaîne latérale ou ligand) est calculé une fois pour toutes en utilisant pour chacun de ses rotamères la structure native pour les autres groupes.

2.3.3 « Fluctuating Dielectric Boundary » (FDB)

Une nouvelle approximation GB a récemment été introduite dans Proteus [49], toujours avec l'objectif de rendre ce terme décomposable par paires. Elle exploite le fait que dans le GB, l'environnement diélectrique d'une paire de résidus est complètement caractérisé pour un petit ensemble de rayons de solvatation d'atomes. Ces rayons sont eux-mêmes sommes de paires sur les atomes de la protéine [53], [54]. La méthode s'appelle Fluctuating Dielectric Boundary » ou FDB et comporte deux étapes.

La première consiste à définir un rayon de solvatation B_I pour chaque résidu I de la protéine. On définit une énergie propre à chaque paire de résidus I, J par la somme suivante :

$$E_{IJ}^{self} \stackrel{def}{=} \sum_{i \in I, j \in J} E_{ij}^{self} \quad (2.5)$$

puis l'énergie propre d'un résidu I :

$$E_I^{self} \stackrel{def}{=} \sum_J E_{IJ}^{self} \quad (2.6)$$

Alors le rayon de solvatation moyen B_I est défini par :

$$E_I^{self} \stackrel{def}{=} \tau \sum_{i \in I} \frac{q_i^2}{2B_I} \quad (2.7)$$

avec $\tau = \frac{1}{\epsilon_p} - \frac{1}{\epsilon_s}$, q_i la charge de l'atome i . Nous avons

$$\left(\sum_{i \in I} q_i^2 \right) \frac{1}{B_I} = \sum_{i \in I} \frac{q_i^2}{b_i} \quad (2.8)$$

B_I est donc la moyenne harmonique des b_i pondérés par les charges au carré. Il est alors possible de définir la contribution g_{IJ} de la paire de résidus I et J à l'énergie ΔG_{elec} , par :

$$g_{IJ} = \sum_{i \in I, j \in J} \tau q_i q_j \left(r_{ij}^2 + B_I B_J \exp[-r_{ij}^2/4B_I B_J] \right)^{-1/2} \quad (2.9)$$

Pour $I = J$, on exclut les termes $i = j$. On peut noter qu'aux distances fixes r_{ij} , et avec $B = B_I B_J$, $g_{IJ}(B)$ varie faiblement en fonction de B . Archontis et Simonson [55], proposent d'approximer cette fonction par :

$$g_{IJ}(B) \approx c_1^{IJ} + c_2^{IJ} B + c_3^{IJ} B^2 + c_4^{IJ} B^{-1/2} + c_5^{IJ} B^{-3/2} \quad (2.10)$$

Les coefficients c_n^{IJ} peuvent être précalculés et stockés dans la matrice d'énergie, puisque les distances r_{ij} ne sont pas des variables pour un couple de chaînes latérales I, J et un choix de rotamères donnés. Pour une conformation donnée, le calcul de l'énergie GB à l'aide de l'approximation 2.10 est maintenant décomposable par paires.

2.4 La matrice d'énergie

Proteus utilise un backbone fixe, un espace discret de rotamères et une fonction d'énergie décomposable par paires. Ces trois éléments permettent de précalculer toutes les énergies d'interactions possibles. À cet ensemble d'interactions, il faut ajouter les interactions des résidus avec le backbone, pour chacun des rotamères possibles, pour constituer un ensemble complet de valeurs énergétiques. Cet ensemble permet d'obtenir rapidement la valeur de la fonction d'énergie pour chaque séquence-conformation. Cet ensemble peut être organisé sous la forme d'une matrice symétrique dans laquelle chaque couple de positions dans la chaîne polypeptidique apparaît avec sa multiplicité de couples de rotamères possibles, voir la figure 2.3.

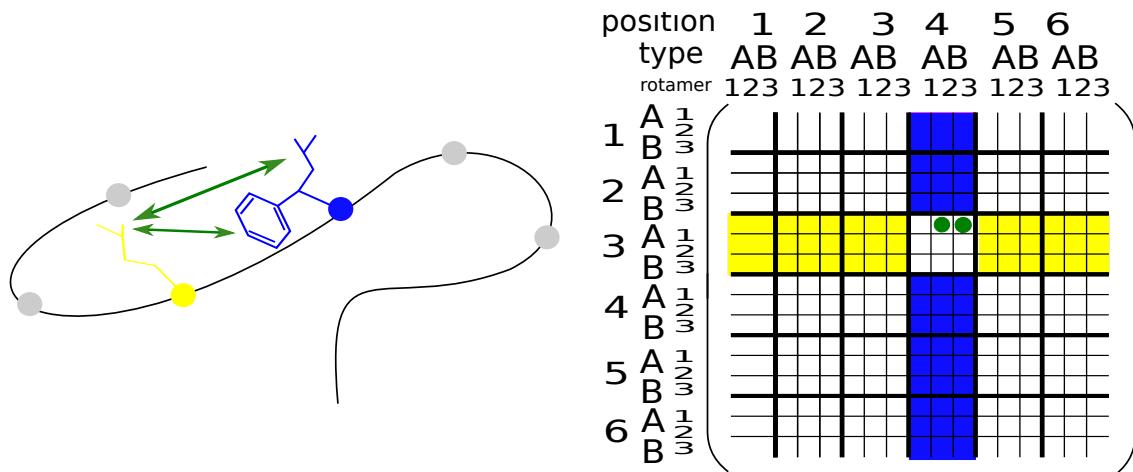


Figure 2.3 – **La matrice d'énergie.** Cet exemple montre un polypeptide de 6 résidus, chaque position possède 2 types d'acides aminés possibles et 3 rotamères possibles (2 pour le type A et 1 pour le type B). La matrice organise toutes les interactions de paires de chaînes latérales possibles. Les interactions de la bande jaune de la matrice impliquent le résidu numéro 3, celles de la bande bleue impliquent le résidu numéro 4. Les points rouge et vert correspondent aux interactions notées respectivement par les flèches rouge et verte à gauche.

2.4.1 Les énergies de l'état déplié

Notre expression de la stabilité relative de deux séquences-conformations se base sur la différence d'énergies $\Delta\Delta G$ entre l'état replié de la protéine et un état déplié. Nous avons donc besoin d'attribuer une énergie à l'état déplié. Proteus utilise une définition indépendante de la structure, telle que :

$$E^u(S) = \sum_i^N E_{t_i}^u \quad (2.11)$$

avec $E_{t_i}^u$ l'énergie du type d'acide aminé t_i . Ces énergies sont prises en entrée dans Proteus et leur détermination en amont doit être fonction du système étudié. Nous détaillerons dans le chapitre 5 plusieurs méthodes dont de nouvelles et plusieurs exemples de détermination qui seront exploités et évalués.

2.4.2 Déroulement de la construction de la matrice

Dans la suite, on appelle position active une position pour laquelle tous les types d'acides et tous les rotamères de chaque type d'acide aminé sont autorisés au cours de l'exploration. Lorsqu'une position n'est pas active, le type de l'acide aminé de la position est fixé. Si les rotamères de la chaîne latérale ne sont pas fixés, on dit que la position est inactive. Si la position n'est pas active et qu'en plus la conformation est fixée, on dit que la position est gelée.

Le calcul de la matrice d'énergie est exécuté à l'aide du programme Xplor [47]. À partir d'un fichier PDB, une série de scripts Xplor commence par préparer le système, qui peut contenir un ligand. Le déroulement est le suivant :

1. L'utilisateur configure l'exécution. Il détermine :
 - un champ de force (AMBER ff99SB, CHARMM19, CHARMM22, OPLS sont supportés.)
 - un traitement du solvant parmi CASA, GB/NEA, GB/FDB
 - un jeu de coefficients σ_i pour la pondération de la surface accessible au solvant
 - un jeu d'énergies dépliées $E_{t_i}^u$
 - les résidus autorisés à muter
 - les résidus autorisés à changer de rotamère
2. Les atomes du squelette sont fixés une fois pour toutes. On ne décrit pas ici le CPD multisquelette disponible dans Proteus [10].
3. Pour chaque résidu, les atomes des chaînes latérales sont placés avec les angles dièdres issus de la bibliothèque de rotamères. L'ensemble de conformations ainsi défini est sauvegardé dans un fichier PDB par position.
4. Les rayons de solvatation de Born sont calculés selon l'approximation GB demandée. Ces rayons sont sauvegardés dans un fichier dédié.
5. Pour chaque rotamère de chaque type, après une courte minimisation, où lui seul peut se déplacer, l'énergie d'interaction avec le squelette est calculée. Elle est stockée dans un fichier : ce sont les énergies de la diagonale de matrice. L'objectif de la minimisation est d'adapter le rotamère à son environnement natif.
6. Pour chaque paire de rotamères, comme pour la diagonale de la matrice, une courte minimisation est effectuée dans laquelle seule la paire courante peut se déplacer. Puis les termes d'énergie d'interaction du couple sont calculés et enregistrés dans des fichiers.

Pour chaque paire d'acides aminés, l'énergie d'interaction a été obtenue après 15 pas de minimisation, où seules les interactions entre la paire et avec le backbone sont prises en compte. Cette courte minimisation réduit l'impact de l'approximation de rotamères discrets. Les rotamères sont extraits de la librairie de Tuffery et al. [56] légèrement étendue, pour un total de 254 rotamères sur l'ensemble des types d'acides aminés. Cette extension comprend des orientations d'hydrogène supplémentaires pour les groupes OH et SH [31]. Cette bibliothèque de rotamères a été choisie pour sa simplicité et parce qu'elle a donné de très bonnes performances dans les tests de placement de chaînes latérales [57, 58]. Les scripts sont conçus pour pouvoir distribuer les calculs sur différentes architectures matérielles allant du PC monoprocesseur au cluster de calculs hétérogène.

2.4.3 Les fichiers d'énergies

Après le calcul de la matrice d'énergie, une étape de fusion des résultats permet d'obtenir deux fichiers d'énergies : un fichier d'énergies propres (diagonales) et un fichier d'énergies d'interactions. Le premier rassemble les énergies propres de chaque chaîne latérale possible et son interaction avec la partie fixe de la protéine (le backbone et les résidus gelés). Chaque ligne correspond à un rotamère possible de chaque type possible de chaque position non gelée. Les premiers champs identifient la position du résidu, son type, son rotamère. Les autres champs contiennent les termes de l'énergie. Les détails sont montrés aux figures 2.4 et 2.5.

Le second fichier rassemble les énergies entre les paires de rotamères. Il comporte deux types de lignes. Le premier donne le couple de positions et de types. Des lignes du second type suivent et donnent dans les deux premiers champs le couple de rotamères impliqué dans l'interaction et l'ensemble des termes énergétiques, figures 2.4 et 2.5. Ensuite, les différents fichiers d'énergies seront lus par proteus.

2.5 Configuration de l'exploration

L'étape suivante est l'exploration de l'espace. Elle s'effectue avec le programme proteus écrit en C. Ce programme se contrôle par un fichier de configuration de type XML sans imbrication de balise. Il y a actuellement près de quarante balises possibles, une partie est présentée table 2.1 page 36. Nous en détaillons quelques-unes ici. L'utilisateur peut choisir la méthode d'exploration entre MSD (« Multistart Steepest Descent heuristic »), MC/REMC ou le champ moyen. Pour le MSD, il peut paramétriser le nombre de cycles et le nombre de passages sur la séquence, voir section 1.4.4 page 19. Pour le MC/REMC, le fichier de configuration détermine le nombre de marcheurs, la taille de chaque trajectoire et la période de « swap », c'est-à-dire la période, en nombre de pas, à laquelle une tentative d'échange de répliques est effectuée, voir 1.4.0 page 25. Si elle est non nulle, l'exploration est de type REMC.

Pour contrôler les déplacements des marcheurs REMC, il existe

- cinq balises pour définir les déplacements
- une balise qui définit un voisinage de chaque position, utilisé en cas de double déplacement.

Il est possible de réduire l'espace des états que peut prendre une séquence-conformation. Les explications sont en 2.5.2. Il existe deux balises pour définir un système de groupes et déclarer une fonction de score. Elles permettent de définir une optimisation non plus uniquement du type de l'équation 2.1, mais répondant à un ensemble de problématiques plus large. Nous expliquons ce système en 2.7.7 page 47.

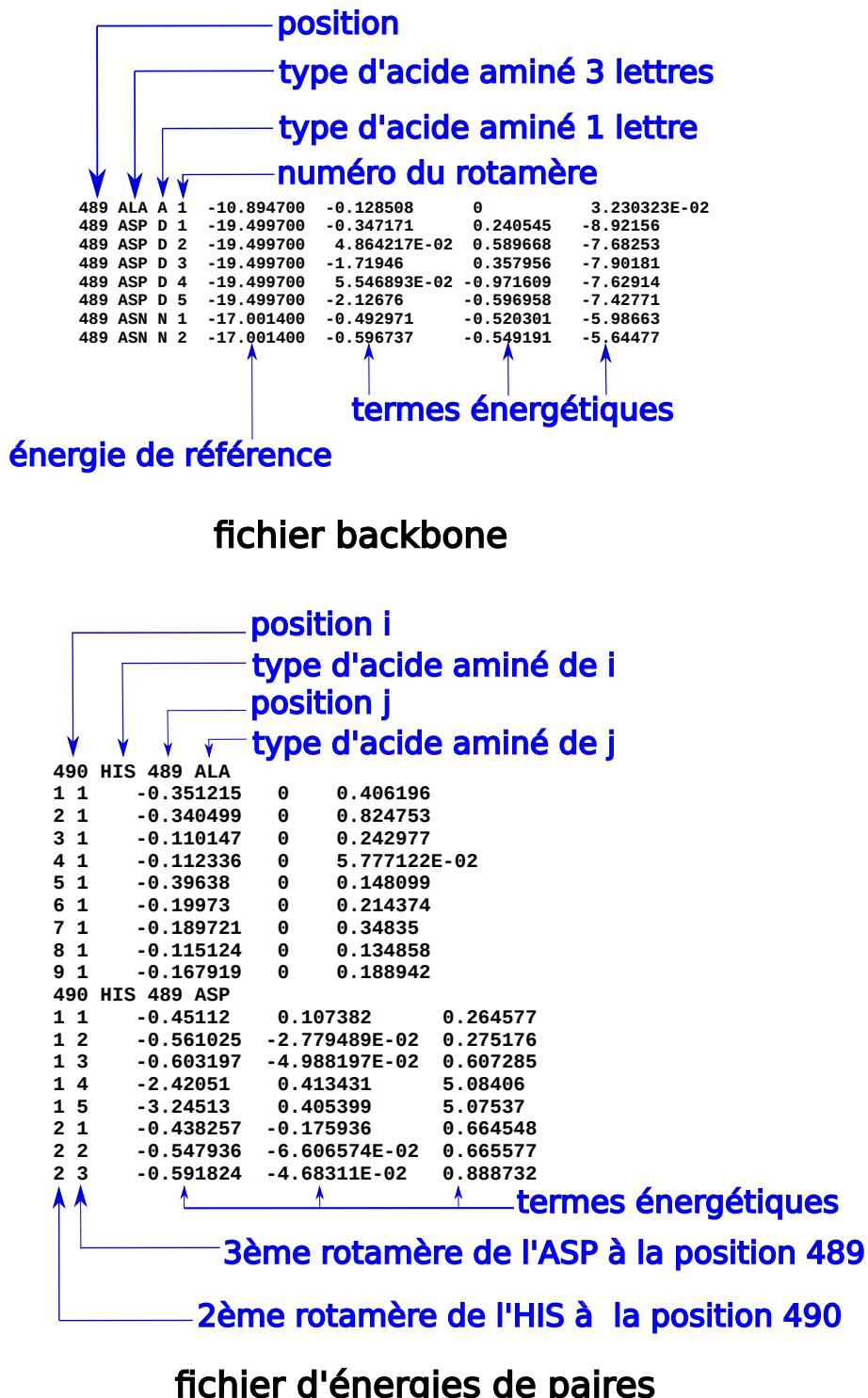


Figure 2.4 – Les fichiers d'énergies en mode CASA

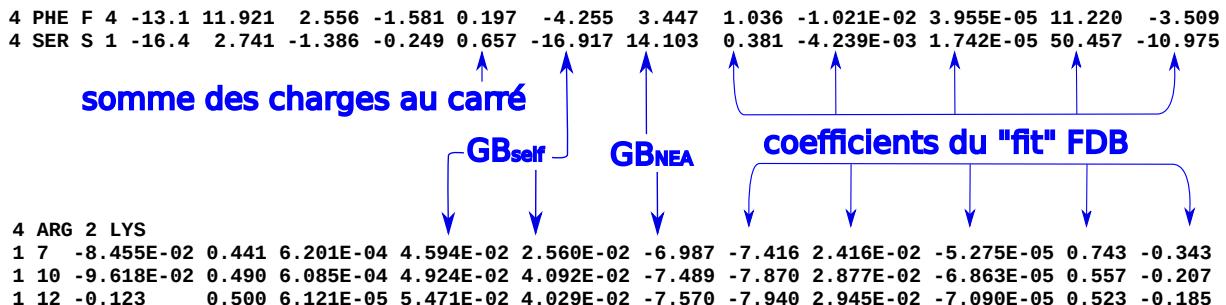


Figure 2.5 – Les fichiers d'énergies en mode GB/FDB

Type	nom	Description
méthode d'exploration	Mode	détermine le mode d'exécution : HEURISTIC(MSD), MONTECARLO, MEANFIELD et POSTPROCESS
nombre de pas	Trajectory_Length	la longueur de la trajectoire MC ou REMC
	Trajectory_Number	le nombre de trajectoires MC ou REMC
	Cycle_Number	le nombre de cycles en mode HEURISTIC
	Sequence_Pass_Number	le nombre maximum d'itérations sur la structure à chaque cycle HEURISTIC
fonction d'énergie	Optimization_Configuration	définition de la fonction d'énergie
	Group_Definition	groupes d'énergies et d'énergies d'interactions
restrictions de l'espace de séquences-rotamères	Space_Constraints	resteint les états pouvant être visités
paramètre	Temperature	attribue les températures aux marcheurs REMC
configuration	Random_Generator	le générateur de nombre aléatoire de la « GNU Scientific Library »
	Rot_Proba	probabilité d'avoir un changement de rotamère à chaque pas
	Rot_Rot_Proba	probabilité d'avoir un double changement de rotamères à chaque pas
	Mut_Proba	...
	Mut_Mut_Proba	...
	Mut_Rot_Proba	... (ancienne version de Proteus)
	Position_Weights	probabilité de tirage de chaque position, lors du premier choix
Monte Carlo	Step_Definition_Proba	probabilité de changer un rotamère ou un type d'acide aminé
	Neighbor_Threshold	Définit la taille des voisinages.
	Fasta_File	le nom du fichier produit par POSTPROCESS
entrées/sorties	Seq_Output_File	le nom du fichier de séquences produit par HEURISTIC ou MONTECARLO
	Energy_Output_File	le nom du fichier d'énergie produit par HEURISTIC ou MONTECARLO

Table 2.1 – Une partie des balises possibles du fichier de configuration de proteus

2.5.1 Les déplacements Monte Carlo

Dans l'algorithme de Metropolis, il est nécessaire de définir une probabilité de sélectionner un état B lorsque le système est dans l'état A . Pour Proteus, une transition se définit par des modifications à certaines positions de la séquence-conformation courante. Il y a deux classes de modifications élémentaires : une mutation et un changement de rotamère. Cinq balises permettent de définir des probabilités associées à ces modifications :

```
<Rot_Proba>
<Mut_Proba>
<Rot_Rot_Proba>
<Mut_Rot_Proba>
<Mut_Mut_Proba>
```

Chacune prend en paramètre une valeur comprise entre 0 et 1. Les deux premières fixent la probabilité des deux modifications élémentaires. Les trois autres fixent la probabilité de modifier deux positions simultanément, au cours d'un même pas Monte-Carlo. Le choix d'une première position à modifier se fait par tirage équitable sur l'ensemble des positions. Le choix de la seconde position se fait par tirage dans le voisinage de la première. Deux positions i et j sont voisines s'il existe un rotamère r_i de i et un rotamère r_j de j tels que :

$$|E(r_i, r_j)| > S$$

avec S le seuil donné par l'utilisateur dans la balise `<Neighbor_Threshold>`. Le voisinage d'une position i est l'ensemble de ses voisins. Ce système de déplacements MC a été revu pendant ma thèse, voir section 3.4 page 56.

2.5.2 Restriction de l'espace séquence-conformation

Dans Proteus, l'espace des états possibles se définit par le contenu du fichier « backbone ». Cependant, l'utilisateur peut restreindre cet espace de la façon suivante :

```
<Space_Constraints>
 489  LYS TRP
 490  ASN ARG{1,8,12}
</Space_Constraints>
```

Cela signifie qu'à la position 489, seuls les types LYS et TRP sont possibles et qu'à la position 490, les types ASN et ARG sont seuls possibles et que pour ARG seuls les rotamères 1, 8 et 12 sont autorisés (selon l'indexation fournie dans le fichier « backbone »). Cette balise permet aussi d'autres classes de restrictions exposées plus loin.

2.5.3 Définition de la fonction de score

Le cycle thermodynamique figure 2.1 peut être adapté pour exprimer d'autres problématiques que celle de la recherche de séquences stables. C'est le cas des calculs de pK_a qui dépendent d'une énergie libre de protonation. Le problème de la reconnaissance d'un

ligand peut également être traité, par un critère d'affinité ou de spécificité. Le critère d'affinité se base sur le cycle thermodynamique 2.6 qui permet d'écrire l'équation :

$$\Delta\Delta G = (G(P:L_2) - G(P:L_1)) - (G(L_2) - G(L_1)) \quad (2.12)$$

avec P la protéine, L_1 et L_2 deux ligands et $P:L_i$ un complexe protéine-ligand.

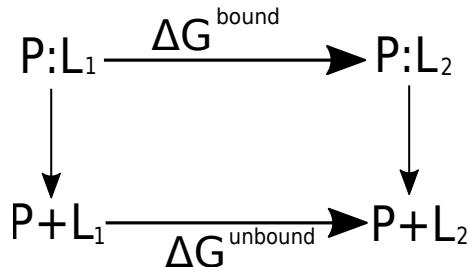


Figure 2.6 – Cycle thermodynamique qui définit l'affinité.

Le critère de spécificité combine une augmentation du poids d'un ligand et la réduction du poids d'un autre. En effet, Proteus permet la décomposition des équations 2.1 et 2.12 en contributions intramoléculaire et intermoléculaire. De plus, il autorise une pondération des différents termes et permet de dupliquer des parties du système. Cela permet également des optimisations simultanées de deux conformations à séquences identiques, mais non fixées. Un exemple est donné avec la déclaration de deux balises suivantes :

```

<Group_Definition>
  prot 5-611
  lig 901
</Group_Definition>
et
<Optimization_Configuration>
  m(0.6prot~lig+0.2lig+0.2prot)
</Optimization_Configuration>
  
```

Nous détaillons en annexe la façon dont ce dispositif fonctionne.

2.6 Les fichiers de sortie

Dans le mode HEURISTIC (algorithme MSD), proteus produit en sortie deux fichiers. Un fichier de conformations-séquences donne à chaque ligne, la meilleure séquence-conformation d'un cycle, son énergie au sens de la fonction de score, le nombre de passages sur la séquence nécessaire à son obtention, voir la section 1.4.4 page 19. Le second fichier donne la fonction de score et l'énergie de chaque groupe ou interaction de groupe utilisée, ceci pour chacune des séquences-conformations du premier fichier. Le lien entre les deux fichiers se fait par l'identifiant unique de résultats situé dans la première colonne des deux fichiers.

Dans le mode MONTECARLO (algorithmes MC et REMC), sont produits un fichier de conformations-séquences et un fichier d'énergies par marcheur. Le format de ces fichiers est le même que celui du mode HEURISTIC, excepté pour le champ contenant le nombre de passages sur la séquence : ici, il est occupé par le nombre de pas pendant lequel le marcheur est resté dans la séquence-conformation avant un déplacement. La température est dans l'entête.

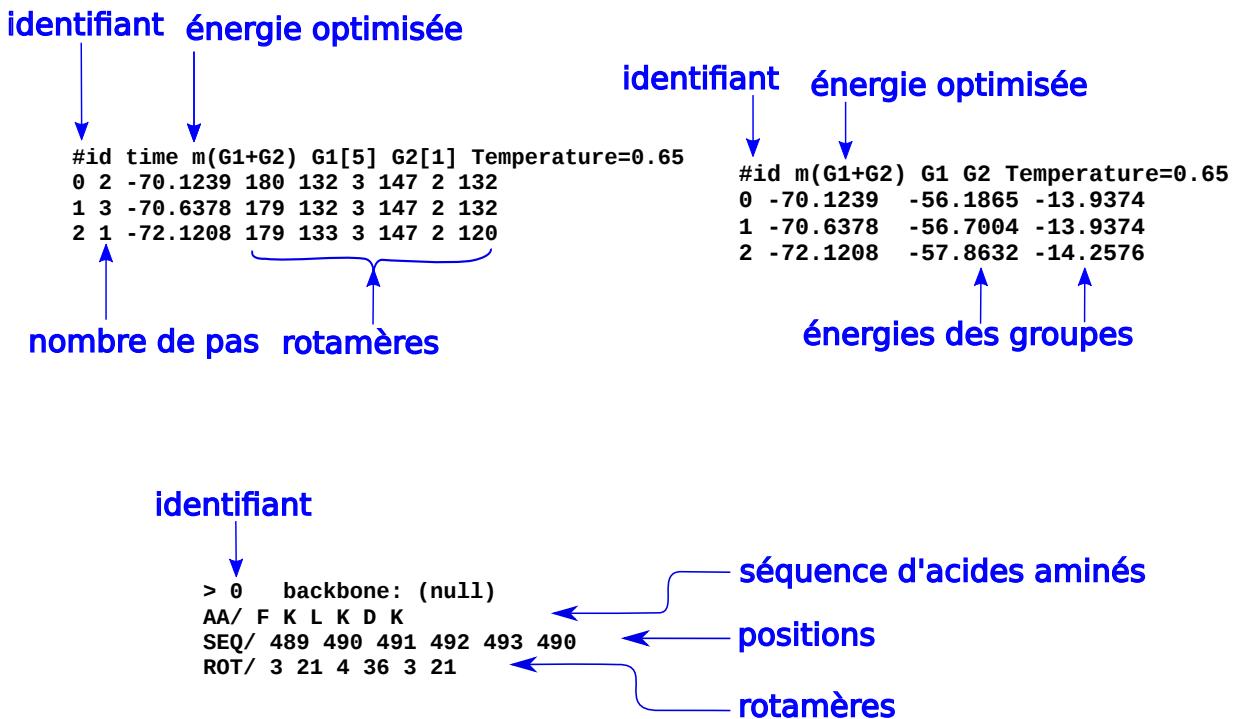


Figure 2.7 – Les fichiers en sortie de proteus en haut pour le mode MONTECARLO, en bas pour le mode POSTPROCESS

Dans les fichiers précédents, les séquences-conformations sont présentées sous forme de listes de nombres qui correspondent au nième rotamère de chaque position selon le fichier backbone. Le mode POSTPROCESS propose une conversion de ce format vers un format de type FASTA où apparaissent la séquence d'acides aminés, la liste des rotamères donnés par leur numéro et les positions PDB pour chaque séquence-conformation. Le format de ces fichiers est présenté à la figure 2.7.

2.7 Outils d'analyse de séquences

Nous présentons maintenant les outils d'analyse que nous utilisons pour examiner la qualité de nos résultats.

2.7.1 Superfamily/SCOP

Superfamily [59] comprend :

- une base de données de modèles de Markov cachés, où chaque modèle représente une structure 3D d'un domaine de la classification SCOP [60].
- une série de scripts qui annotent les séquences données en entrée. Ici, nous utilisons uniquement l'association de chaque séquence au modèle SCOP le plus vraisemblable.

Nous travaillons avec la version 1.75 et nous utilisons SAM (version 3.5) [61] et HMMER (version 3.0) [62], deux outils de manipulation de modèles de Markov cachés pour la biologie recommandés par l'équipe de Superfamily. La base SCOP contient 15 438 modèles.

2.7.2 Taux d'identité de séquences

Soient S et N deux séquences d'acides aminés de même longueur l . Le taux d'identité $Id(S,N)$ de S par rapport à N est égal au pourcentage de positions où l'acide aminé est identique dans S et N :

$$Id(S,N) = \frac{1}{l} \sum_{1 \leq i \leq l} \mathbb{1}(s_i, n_i) \times 100 \quad (2.13)$$

avec s_i et n_i l'acide animé en i de S et de N respectivement, et $\mathbb{1}(x,y)$ la fonction qui vaut 1 lorsque $x = y$ et 0 sinon.

2.7.3 Taux d'identité par position

Le taux d'identité d'un alignement A_S à la position i par rapport à une séquence N de même longueur se définit comme :

$$Id(A_S, i) = \frac{1}{m} \sum_{1 \leq j \leq m} \mathbb{1}(s_i^j, n_i) \times 100 \quad (2.14)$$

avec m le nombre de séquences de A_S . Ce taux d'identité donne une mesure de la ressemblance entre un alignement et une séquence. Cela nous permet de comparer nos séquences calculées aux séquences naturelles de la famille de la native.

2.7.4 Alignements Pfam

La base de données Pfam (Protein families database) [63, 64] regroupe les domaines protéiques en familles. Chaque famille est représentée par des alignements multiples de séquences et des modèles de Markov cachés. Dans la suite, nous utilisons l'alignement « seed » qui est un petit alignement de séquences naturelles représentatives. Il contient 45 séquences pour la famille PDZ. Nous utilisons également l'alignement « RP55 », qui se base sur l'alignement « seed » et augmente le nombre de séquences grâce à des modèles de Markov cachés construits à partir de « seed », jusqu'à contenir 12 255 séquences protéiques naturelles pour la famille PDZ.

2.7.5 Score BLOSUM

Pour tenir compte des ressemblances et des différences entre les acides aminés lors d'une substitution, nous avons besoin d'une matrice de coût. Nous utilisons les matrices BLOSUM40 et BLOSUM62 (BLOCKS SUbstitution Matrix) [65] qui sont construites à partir de blocs d'alignement très conservés (plus de 40% et 62% d'identités respectivement). Le score BLOSUM d'une substitution calculé à partir de la fréquence de la mutation correspondante. À cela est ajouté un score de pénalités pour l'insertion d'un gap, c'est-à-dire un saut dans l'alignement.

On définit alors un score de similarité de deux séquences de même longueur comme la somme des scores BLOSUM62 sur toutes les positions. De même, le score de similarité d'un alignement par rapport à une séquence sera défini comme la moyenne des scores de similarité sur ensemble des séquences de l'alignement. Enfin, un score de similarité de deux ensembles de séquences alignés sera la moyenne des scores de similarité du premier ensemble par rapport aux séquences du second.

2.7.6 Similarité d'un ensemble à un alignement Pfam

Afin de calculer un score de similarité d'un ensemble de séquences CPD par rapport à une famille Pfam, il faut commencer par aligner ces séquences avec l'alignement de la famille. Pour cela, nous utilisons le programme d'alignement BLAST [66, 67]. Il implémente une heuristique qui recherche puis étend les meilleurs alignements locaux. Nous procédons comme suit :

1. La commande `blastp` est utilisée avec comme base de données (paramètre `-db`) l'alignement Pfam et comme séquence en entrée (paramètre `-query`) la séquence native d'intérêt.
2. Dans la sortie `blast`, la séquence qui produit l'alignement le plus significatif avec la native est collectée, notons-la S_0 .
3. L'alignement `blast` est alors utilisé pour positionner la native par rapport à S_0 et les gaps nécessaires pour aligner la native à S_0 sont ajoutés.
4. Le positionnement et les gaps sont alors appliqués tels quels aux séquences CPD.

2.7.7 Entropie par position

Pour comparer la diversité des séquences CPD avec la diversité des séquences naturelles, nous utilisons l'entropie par position [68], à partir de la formule :

$$S_i = - \sum_{j=1}^6 f_j(i) \ln f_j(i) \quad (2.15)$$

avec $f_j(i)$ la fréquence du type de résidu j à la position i . Au lieu de distinguer les 20 types d'acides aminés, nous utilisons six classes de résidus, correspondant aux groupes suivants :

{L,V,I,M,C}, {F,Y,W}, {G}, {A,S,T,P}, {E,D,N,Q} et {K,R,H}. Cette classification a été obtenue par un clustering de matrice BLOSUM62 et une analyse des énergies de contact entre résidus dans les protéines [69]. Pour obtenir une mesure du nombre de types d'acides aminés apparaissant à une position, on utilise l'exponentielle de l'entropie par position $\exp(S)$ (qui varie de 1 à 6). Cela correspond à un nombre moyen de classes échantillonnées par position. Par exemple, une valeur de 2 à une position particulière indique que les acides aminés de deux des six classes sont présents à cette position en moyenne au sein des séquences analysées. Ensuite, on moyenne cette valeur sur l'ensemble des résidus de la chaîne protéique. Une valeur moyenne globale de 2 indique qu'en moyenne, deux classes d'acides aminés sont présentes à n'importe quelle position.

Annexe du chapitre 2 : Structures de données dans proteus

Cette annexe présente les principales structures de données utilisées dans le programme proteus. Un premier ensemble de structures regroupe les données physiques fournies en entrée à proteus, voir la figure 2.8. Le fichier backbone contient, d'une part, la description de tous les rotamères possibles à chaque position du système et d'autre part les énergies propres de chacun de ces rotamères. La structure `residues` contient la totalité de ces rotamères. Ils sont regroupés dans un tableau `rotamers` pour chaque type. L'ensemble des types est regroupé à son tour dans un tableau `types`. La matrice `ener` regroupe les énergies de paires dans un tableau à deux dimensions représentant les couples de positions (i,j) . Chaque élément de ce tableau contient l'ensemble des couples de rotamères de (i,j) sous forme de tableau à deux dimensions.

Le deuxième ensemble est constitué des structures « logiques ». Ce sont elles qui gèrent la duplication de parties du système comme expliqué à la section 2.7.7. La structure `posi_instance` contient un ensemble d'index sur un ensemble d'éléments des tableaux `rotamers` permettant la définition d'un espace d'état qui lui est propre. La structure `group` contient une liste de `posi_instance`. Cela est détaillé à la figure 2.9.

Le dernier ensemble de structures de données contient les éléments régulièrement modifiés au cours de l'exploration. Il y a un tableau `grp_ener` contenant les énergies de groupes ou interactions entre groupes utilisées dans la fonction de score. Il représente la partie utile de la matrice de la figure 2.12. La structure `conformation` contient la séquence-conformation courante. Une liste de structures `ins_modif` représente les modifications à effectuer sur la conformation courante, figure 2.10. Ceci permet la mise à jour des différentes énergies au cours de la trajectoire.

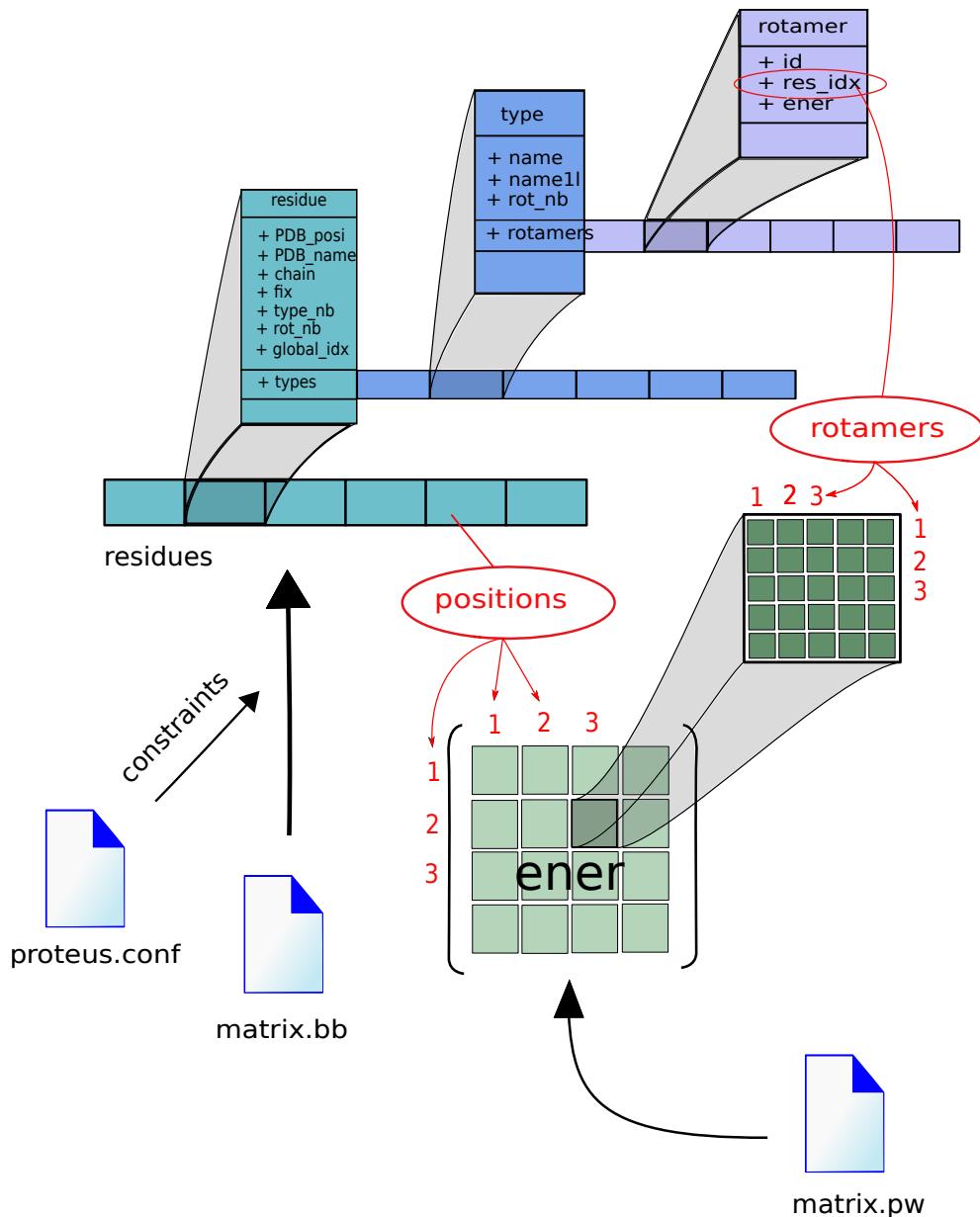


Figure 2.8 – Les principales structures « physiques » dans proteus

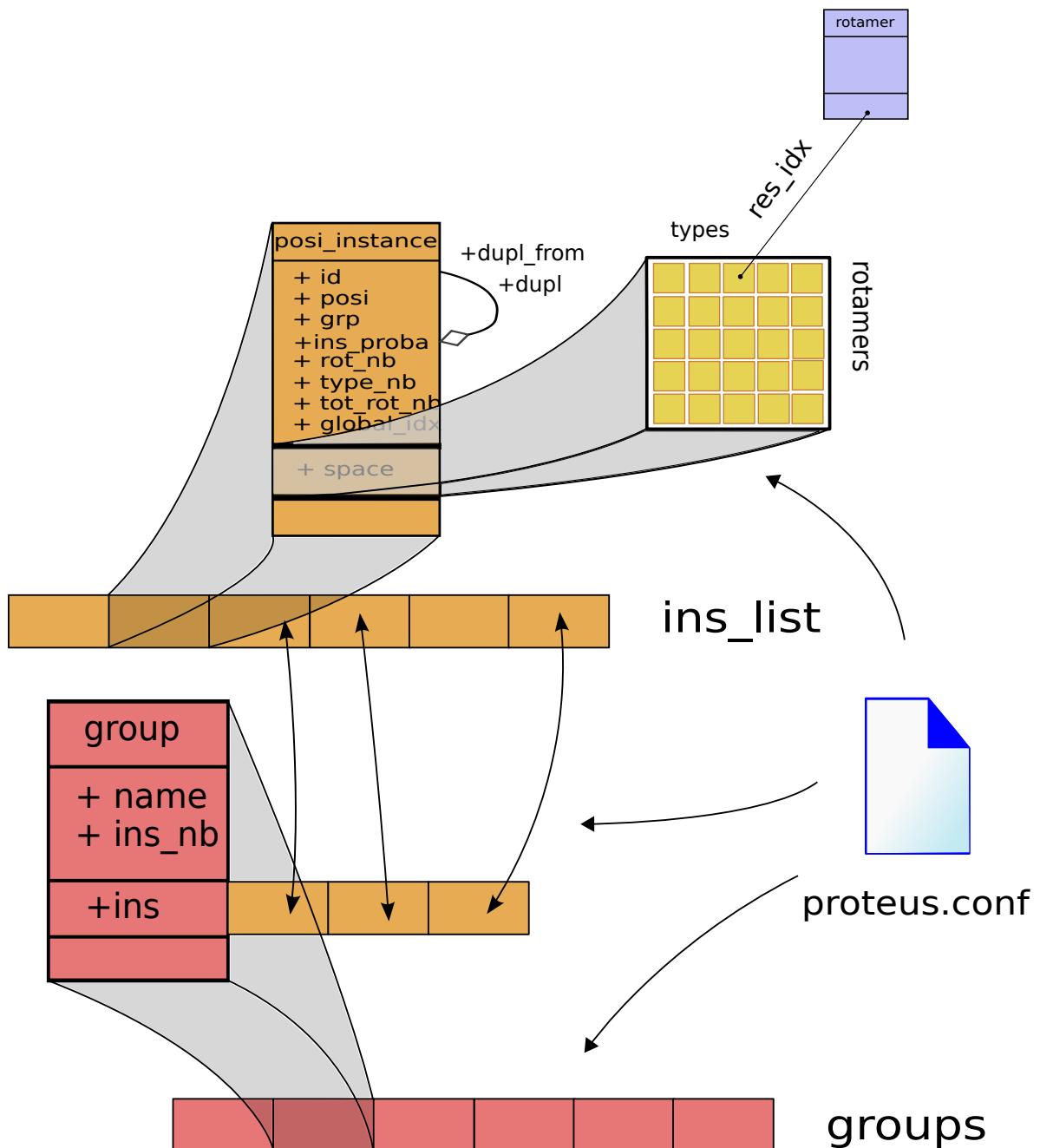


Figure 2.9 – Les principales structures « logiques » dans proteus

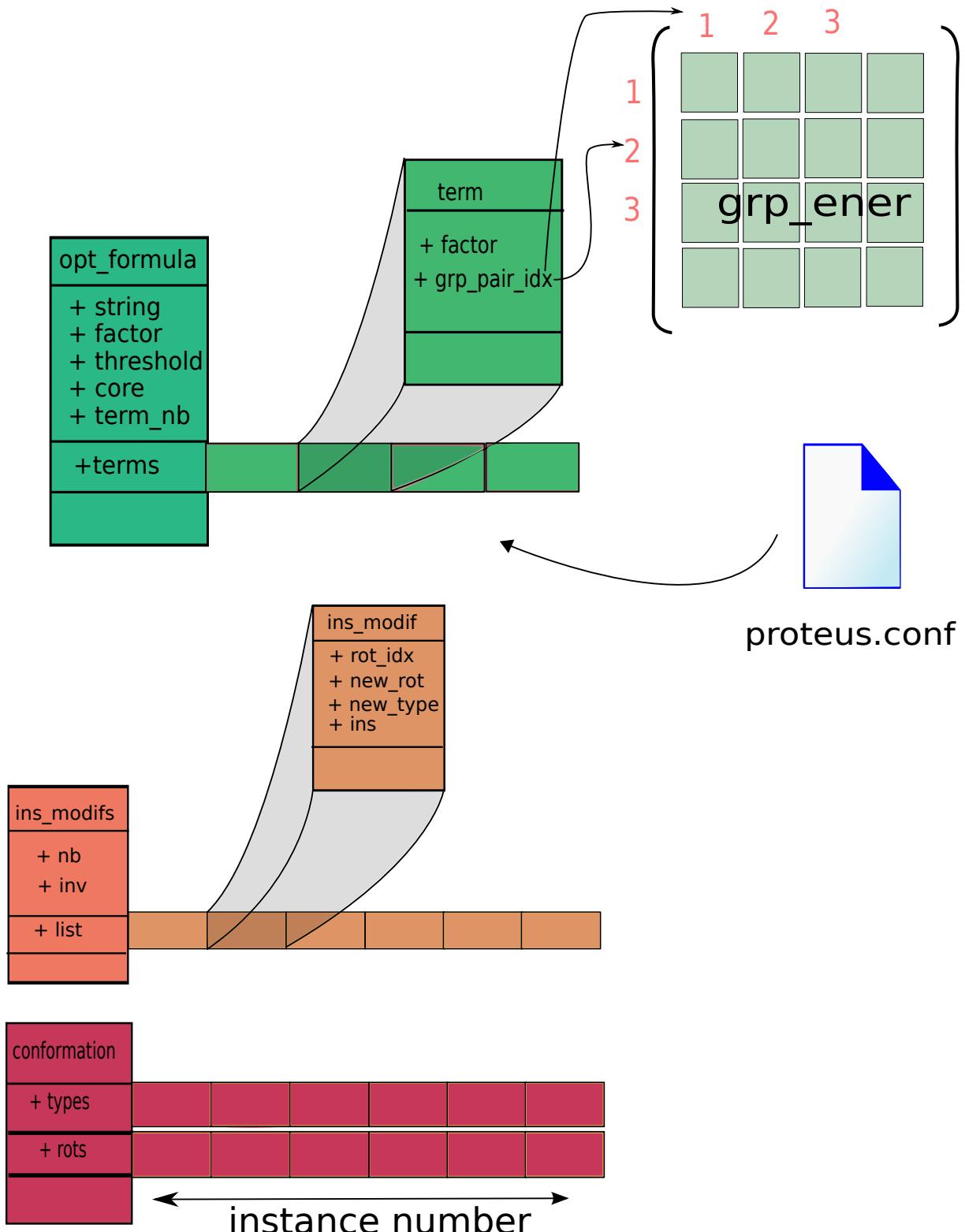


Figure 2.10 – Les principales structures « dynamiques » dans proteus

Annexe du chapitre 2 : Définition de la fonction de score

Nous détaillons ici, le dispositif de définition de la fonction de score.

Définition de groupes

L'utilisateur peut définir des groupes au sein du système. Un groupe se définit en donnant la liste des positions qu'il contient dans la balise <Group_Definition>. Par exemple :

```
<Group_Definition>
grp1 1-3
grp2 4 5
grp3 6-8
</Group_Definition>
```

On peut définir plusieurs groupes associés à un même ensemble de positions :

```
<Group_Definition>
...
grp3 6-8
grp4 6-8
</Group_Definition>
```

Dans l'exemple précédent, grp4 représente une seconde conformation-séquence des résidus 6 à 8. Les groupes grp3, grp4 ont tous deux leur propre espace d'états, qui peuvent être différents :

```
<Space_Constraints>
grp3.6 LEU TRP
grp4.6 ASN ARG
</Space_Constraints>
```

Il peut au contraire relier leurs états respectifs :

```
<Space_Constraints>
grp4.TYPE grp3
</Space_Constraints>
```

La déclaration précédente garantit que grp3 et grp4 auront des séquences identiques tout au long de l'exploration.

En tirant parti de la décomposition par paires de la fonction d'énergie, on peut exprimer simplement l'énergie d'un groupe et l'interaction entre deux groupes. On a :

$$E(grp_i) = \sum_{a \in grp_i} E_a + \sum_{a \in grp_i} \sum_{b \in grp_i, a < b} E_{a,b} \quad (2.16)$$

et

$$E(grp_i, grp_j) = \sum_{a \in grp_i} \sum_{b \in grp_j} E_{a,b} \quad (2.17)$$

avec $E(grp_i)$ l'énergie du groupe grp_i , $E(grp_i, grp_j)$ l'énergie d'interaction entre les groupes grp_i et grp_j , E_a l'énergie propre du résidu a et $E_{a,b}$ l'énergie d'interaction entre les rotamères aux positions a et b . On voit les énergies de groupes sur la matrice d'énergies, figure 2.11. Les contributions d'un groupe se situent dans un carré sur la diagonale et les contributions à l'énergie d'interaction entre deux groupes se situent hors de la diagonale.

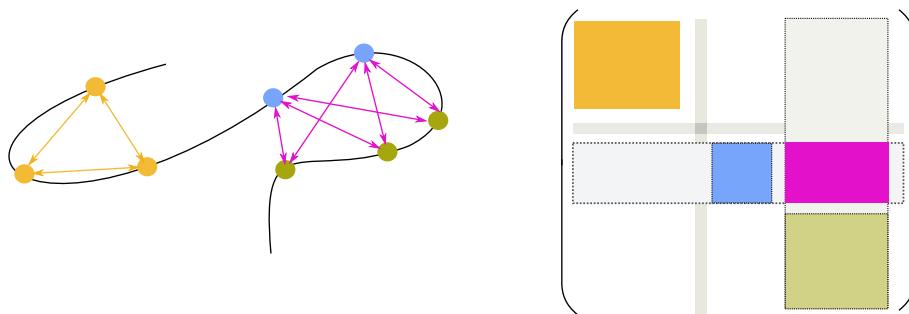


Figure 2.11 – **Les contributions aux énergies de groupes et d'interaction entre groupes dans la matrice d'énergie.** L'énergie des groupes orange, bleu et vert est une somme de termes situés des carrés de même couleur dans la matrice d'énergie. L'interaction entre le groupe bleu et le groupe vert est la somme de tous les termes du rectangle rouge.

Déclaration de la fonction de score

On peut introduire une nouvelle matrice qui rassemble les énergies des groupes et les énergies d'interaction entre les groupes, voir la figure 2.11. Dans cette matrice, l'énergie propre d'un groupe est un élément de la diagonale, celle d'une interaction entre groupes, un élément hors de la diagonale. La fonction de score peut se définir comme une combinaison linéaire des éléments de cette matrice :

```
<Optimization_Configuration>
  m(0.2grp1+0.2grp2+0.3grp3~grp3-0.3grp2~grp4)
</Optimization_Configuration>
```

Ici la notation de l'énergie d'interaction entre deux groupes $grp2$ et $grp3$ est $grp2\sim grp3$ et l'énergie propre du groupe $grp1$ est simplement $grp1$. Les nombres sont des pondérations des énergies. La notation $m(E)$ indique que la fonction de score demandée est une minimisation de la somme déclarée entre les parenthèses. Cela signifie que pour deux séquences-conformations A et B, A est meilleure que B si $E(A) < E(B)$. Cette convention intervient dans l'exploration MSD. On peut utiliser $t(...)$ pour indiquer une fonction de

seuil :

```
<Optimization_Configuration>
t(grp1)<125.8
</Optimization_Configuration>
```

A est meilleure que B si $E(A) < 125,8$. C'est utile dans le cas de l'algorithme MSD, cela permet d'obtenir un ensemble de conformations-séquences avec une énergie inférieure à un seuil.

$$E_G(C) = \begin{pmatrix} E_{1,1} & E_{1,2} & E_{1,3} & E_{1,4} \\ E_{1,2} & E_{2,2} & E_{2,3} & E_{2,4} \\ E_{1,3} & E_{2,3} & E_{3,3} & 0 \\ E_{1,4} & E_{2,4} & 0 & E_{4,4} \end{pmatrix}$$

Figure 2.12 – **La matrice des énergies de groupe.** Les énergies des groupes grp1, grp2, grp3 et grp4 et leurs interactions présentées sous forme de matrice. Il n'y a pas d'interaction entre grp3 et grp4 parce qu'ils sont associés à un même sous-système (dupliqué).

Chapitre 3

Développements algorithmiques

Ce chapitre expose les modifications faites dans le logiciel Proteus et dans Xplor pendant la préparation de cette thèse. Xplor est un logiciel conçu pour la biologie structurale développé à l'origine par Axel T. Brünger à l'université de Yale [47]. Il propose un langage de script permettant d'exploiter ses fonctionnalités. Pour adapter Xplor au CPD nous y ajoutons la gestion de jeux de coordonnées multiples (rotamères) pour chaque résidu. Plusieurs modèles de solvant implicite ont également été ajoutés.

Le REMC est par nature un algorithme parallèle, nous présentons l'implémentation de cet algorithme dans proteus avec notre gestion du parallélisme. Le schéma proposé par Metropolis et Hasting constitue un cadre général dans lequel il est possible d'adapter les probabilités utilisées au système à étudier. Nous expliquons comment le critère d'acceptation a été amélioré en tenant compte de la spécificité de notre modèle et de nos objectifs. Enfin sont présentées quelques nouvelles fonctionnalités dans proteus : un système de pondération pour le choix des positions à modifier, un contrôle du taux de mutations et de changements de rotamères, une gestion dynamique de la mémoire en fonction du fichier de configuration et un système de création de labels qui simplifie la configuration.

3.1 Les Modèles

Proteus, lors de la préparation du système, place chaque chaîne latérale possible aux différentes positions de la chaîne polypeptidique selon la librairie de rotamères de Tuffery. Ces placements sont utilisés à de nombreuses reprises pendant le calcul des énergies d'interactions. Xplor ne permettait qu'une gestion de quatre jeux de coordonnées pour une chaîne latérale simultanément. Cela oblige l'utilisation de nombreux fichiers pour stocker l'ensemble des jeux de coordonnées. Afin de remédier à ce problème, nous modifions le code source d'Xplor pour y introduire deux nouvelles notions. Une « resclass » identifie un résidu par un triplet composé du « resid », du « resname » et du « segid ». Ces trois notions existent dans le format PDB et Xplor. Ils représentent respectivement la position dans une chaîne polypeptidique, le type d'acide aminé, le segment ou molécule à laquelle appartient le résidu. Un « modèle » est un jeu de coordonnées d'une resclass. Une resclass peut avoir plusieurs modèles.

L'utilisateur ne manipule que les modèles ; les resclass n'apparaissent ni dans les fichiers d'entrée/sortie ni dans les commandes. Un modèle se déclare par des lignes ATOM d'un

ATOM	339	N	GLY	39	-3.933	5.444	16.117	1.00	1.00	13	A
ATOM	340	H	GLY	39	-3.661	4.518	16.366	0.00	1.00	13	A
ATOM	341	CA	GLY	39	-4.479	6.276	17.176	1.00	1.00	13	A
ATOM	342	C	GLY	39	-3.682	7.552	17.389	1.00	1.00	13	A
ATOM	343	O	GLY	39	-4.251	8.617	17.614	1.00	1.00	13	A
ATOM	1044	OD1	ASP	111	-13.801	-5.521	-3.500	1.00	0.00	14	A
ATOM	1045	OD2	ASP	111	-14.043	-4.328	-5.339	1.00	0.00	14	A
ATOM	1046	C	ASP	111	-12.244	-8.798	-5.753	1.00	0.00	14	A
ATOM	1047	O	ASP	111	-11.038	-9.008	-5.762	1.00	0.00	14	A
ATOM	1048	N	LYS	112	-13.097	-9.470	-6.515	1.00	0.00	14	A
ATOM	1049	H	LYS	112	-14.075	-9.280	-6.501	0.00	0.00	14	A
ATOM	1050	CA	LYS	112	-12.655	-10.531	-7.415	1.00	0.00	14	A
ATOM	1051	CB	LYS	112	-13.852	-11.135	-8.142	1.00	0.00	14	A
ATOM	1052	CG	LYS	112	-14.788	-11.857	-7.220	1.00	0.00	14	A

↑
index du modèle

Figure 3.1 – Exemple de fichier PDB avec déclaration de 3 modèles. L'avant-dernière colonne contient l'index des modèles. Par exemple, il y a un modèle pour GLY à la position 39 du segment A et un autre pour ASP, position 111, segment A.

fichier PDB. Il y a deux possibilités de lecture des modèles. La commande :

```
coor disp=model @file.pdb
```

ajoute chaque modèle de file.pdb dans un tableau en mémoire. Un nombre est lu à la colonne 67-71, il représente l'indice du modèle pour une resclass. Le lien avec la resclass se fait par la liste des atomes contenant l'indice, voir un exemple à la figure 3.1. La commande :

```
coor disp=model @file.pdb push=true
```

ajoute un seul modèle par resclass. L'indice d'un modèle n'est pas lu, mais calculé comme le plus grand indice existant plus un. La copie de modèle se fait par la commande :

```
coor copy from=A to=B idx=i=j end
```

avec A et B pouvant prendre les valeurs main, comp, xref ou model. Le mot idx=i=j n'est pas obligatoire. Par défaut, si from=model alors idx=1 et si to=model le nouvel indice créé sera le plus grand indice plus un. L'ancienne syntaxe est toujours supportée. La commande :

```
write coor sele=(resid $1 and resn $aa1) from=model output=new.pdb end
```

imprime les modèles de chacune des resclass définies par la sélection dans un fichier PDB. On écrit une ligne par atome de la resclass avec l'indice du modèle, pour tous les modèles. Il est possible de limiter l'impression à un seul modèle *i* par une commande du type :

```
write coor from=model idx=i output=new.pdb end
```

3.2 OpenMP pour le REMC

3.2.1 Présentation d'OpenMP

Pour l'implémentation de l'algorithme « Replica Exchange Monte Carlo » nous devons paralléliser la partie Monte-Carlo de proteus. Nous nous orientons vers une programmation à mémoire partagé. Dans ce domaine, l'interface de programmation OpenMP pour « Open

Multi-Processing » offre un standard mature, bien supporté par les compilateurs C/C++ et simple à mettre en œuvre. Il s'agit d'une spécification qui décrit une collection de directives au compilateur, une bibliothèque de routines et un ensemble de variables d'environnement. Le principe est d'ajouter à un code existant des directives pour définir :

- les instructions à exécuter en parallèle (création de fils d'exécution) ; On parle de région parallèle dans le code.
- les situations de synchronisations entre fils d'exécution
- le statut des variables (partagé entre fils, privé, etc.)

La bibliothèque OpenMP permet la configuration de l'exécution et son contrôle par un ensemble de fonctions et de macros. Il existe des fonctions qui gèrent le nombre de fils d'exécution, qui fixent la politique de l'ordonnancement, qui retournent un identifiant du fil d'exécution, etc. La définition de la macro `_OPENMP` garantit la compatibilité de l'exécutable.

Les variables d'environnement contrôlent les mêmes informations que les fonctions OpenMP. Elles doivent être définies avant l'exécution. Elles sont prises en compte avec une priorité plus faible que les fonctions de la bibliothèque, elles-mêmes de priorité plus faible que les directives en cas de conflit.

En entrant dans une région parallèle, le fil courant crée les autres fils. Il possède un statut particulier, il devient le fil maître. Les autres fils se terminent avec la région parallèle ; le maître continue son exécution. Tous les fils ont accès à la même mémoire partagée, notamment aux variables définies avant la région parallèle. La déclaration d'une variable dans une région parallèle engendre la création d'une variable pour chaque fil accessible uniquement par lui.

3.2.2 REMC dans proteus

Comme nous l'avons vu à la section 1.4.7, l'algorithme REMC considère plusieurs simulations indépendantes d'un même système. Cela fait de lui un algorithme bien adapté à la programmation parallèle. Deux points demandent une attention particulière : l'échange de répliques et la création de l'identifiant unique d'impression, voir 2.6. Le schéma général de REMC dans proteus est présenté à la figure 3.2. Une région parallèle est initiée par la directive « `#pragma omp parallel` » : les différents marcheurs sont créés. Chacun réalise une trajectoire de type MC jusqu'à un nombre de pas multiple de la période d'échange. A priori l'avancée des marcheurs n'est pas simultanée. Donc une directive « `#pragma omp barrier` » est placée avant le test d'échange, elle garantit que chacun a effectué le bon nombre de pas. Une fois tous les marcheurs bloqués par cette directive, ils sont libérés. La directive « `#pragma omp master` » dédie l'exécution du test et de l'échange de répliques au seul marcheur maître. Pour empêcher les autres de marcher avant un échange éventuel, une seconde directive « `barrier` » est placée après les instructions d'échanges.

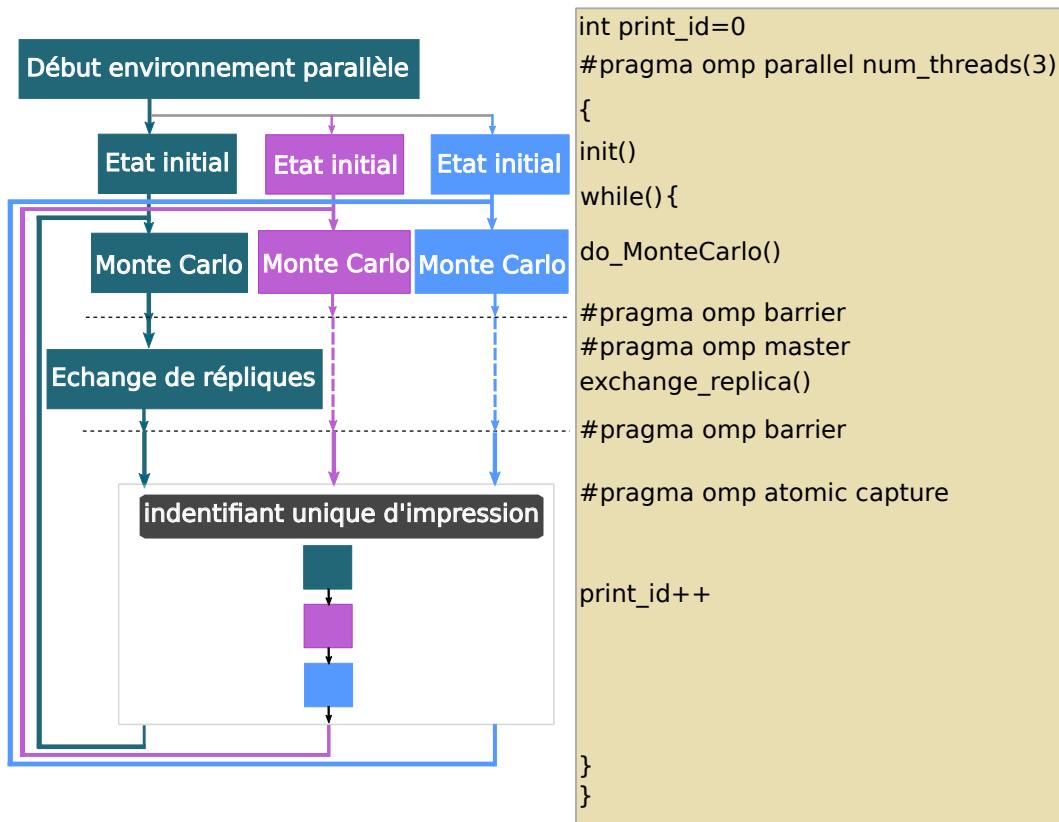


Figure 3.2 – La correspondance entre les directives OpenMP à droite et le comportement des fils d'exécution à gauche sur un REMC simplifié. Sont schématisés, la création d'une région parallèle, deux synchronisations (les lignes pointillées à gauche), une région exécutée uniquement par un fil maître, une affectation séquentielle.

Tout au long d'une trajectoire, des séquences-conformations sont imprimées. Pour faciliter le post-traitement, un identifiant unique sur toute l'exécution est attribué à chacune. Pour que tous les marcheurs puissent l'incrémenter, l'index qui sert d'identifiant est déclaré comme variable partagée. Pour garantir que chaque passage sur cette instruction retourne une valeur unique, une directive « `#pragma omp atomic capture` » est utilisée. Pour terminer cette section, nous donnons deux représentations graphiques du comportement des marcheurs REMC dans proteus à la figure 3.3.

3.3 Amélioration de la fonction d'acceptation MC

Une amélioration a été apportée à la fonction d'acceptation MC de proteus. Pour réaliser une mutation du type t vers t' , nous effectuons un changement de type d'acide aminé dans la structure repliée et le changement inverse dans la structure dépliée, voir la section 2.1. La probabilité d'une mutation est donc le produit de la probabilité de choisir t' par la probabilité de choisir une structure repliée particulière avec t' et une structure dépliée particulière avec t . Dans proteus, ces changements sont sélectionnés par tirage aléatoire uniforme sur l'ensemble des états possibles.

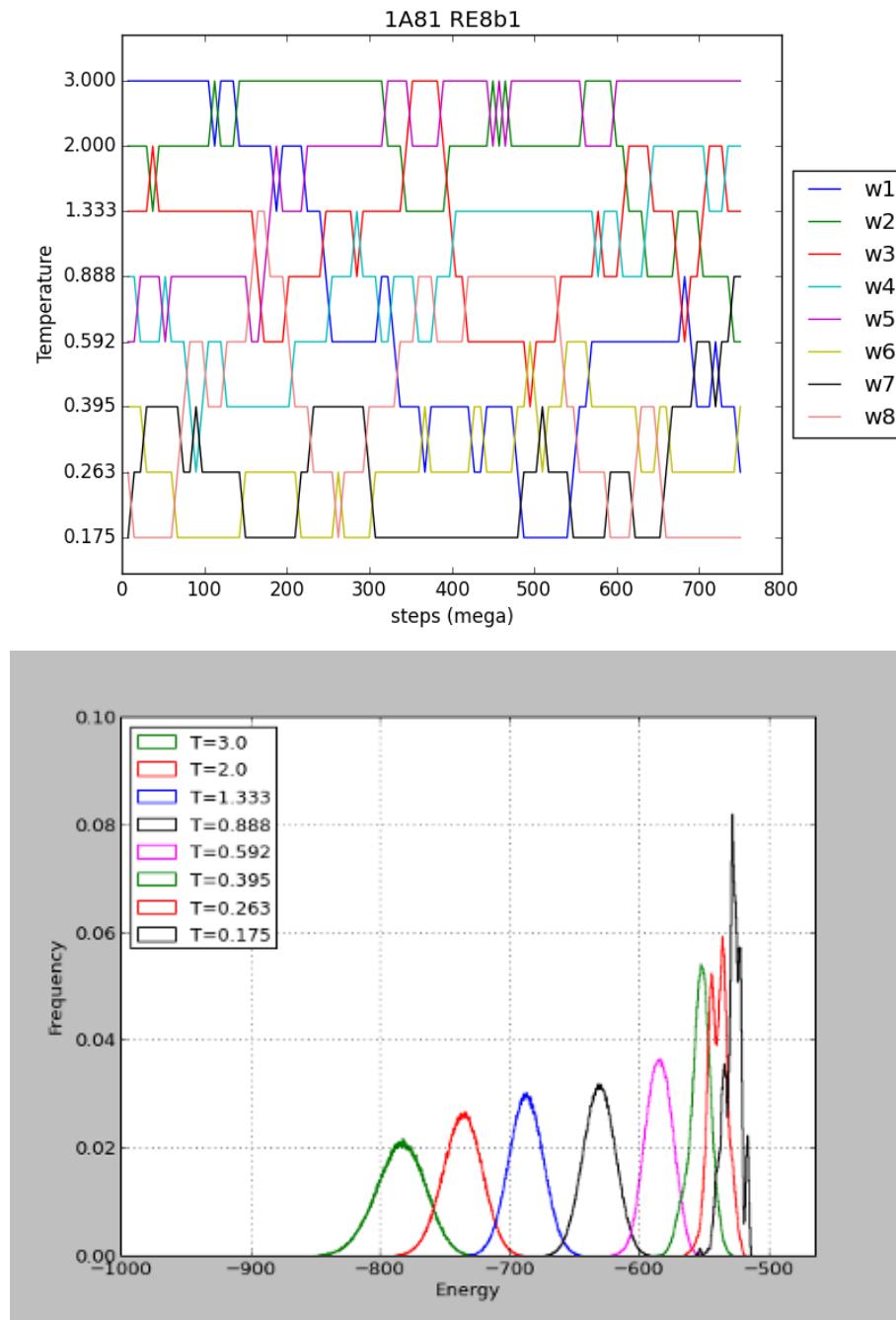


Figure 3.3 – Comportement de proteus pour un REMC à huit marcheurs. En haut, la trajectoire des marcheurs dans l'ensemble des huit températures. En bas, la distribution des énergies par marcheurs. On peut imaginer la distribution cumulée des marcheurs et remarquer que son graphe serait proche de celle d'une distribution de Boltzmann

Se pose alors la question des états possibles des chaînes latérales à l'état déplié. Plusieurs points de vue sont possibles. On peut considérer qu'il existe autant de rotamères à l'état déplié qu'à état replié et qu'ils ont tous la même énergie. Cette possibilité a l'avantage de rendre la probabilité de mutation symétrique, c'est-à-dire que le passage du type t vers t' est aussi probable que le passage de t' vers t . Mais elle a l'inconvénient d'être peu réaliste parce que les rotamères sont les positionnements préférentiels à l'état replié et non à l'état déplié. Nous avons fait évoluer proteus vers un autre point de vue, en considérant qu'une chaîne latérale à l'état déplié n'a qu'un seul rotamère dominant. Cela induit une dissymétrie dans la probabilité de sélection qui est corrigée par une probabilité d'acceptation du type de l'équation 1.39. Cette nouvelle probabilité d'acceptation est décrite en détail en 4.6 page 63.

3.4 Nouveau système de déplacements

Le système de déplacements expliqué en 2.5.1 souffre de certaines rigidités. Il ne permet pas de distribuer les modifications selon les positions dans la chaîne polypeptidique. Il n'est pas possible de répartir alternativement les mouvements entre mutations et changement de rotamères.

Alors, nous introduisons deux nouvelles balises. La première permet la définition de deux poids à chaque position (dupliquée ou non). Le premier poids pondère la sélection d'une position lors d'une mutation, l'autre pondère la sélection d'une position lors d'un changement de rotamères :

```
<Position_Weights>
Rot 489 0.50
Rot 495 490-493 0.05
Mut 489-491 G2.492 G3.489-491 0.052
</Position_Weights>
```

Les poids sont ensuite normalisés avant d'être utilisés lors de la sélection de positions. Ainsi, lorsque proteus prépare un changement de rotamère pendant la modification de la première position, la position 489 a dix fois plus de chance d'être modifiée que 495.

La balise `<Step_Definition_Proba>` permet la construction en termes de probabilité du mouvement effectué à chaque pas. La syntaxe est la suivante :

```
<Step_Definition_Proba>
Rot 1.0
Mut Mut 0.1
</Step_Definition_Proba>
```

Chaque ligne à l'intérieur de la balise définit une forme de pas. Le nombre à la fin de la ligne fixe la probabilité du choix par proteus de cette forme. Les probabilités sont normalisées avant de début de la simulation. Ici, pour un pas quelconque d'une trajectoire, le changement de rotamère est choisi avec la probabilité $\frac{10}{11}$ et une double mutation est

choisi avec la probabilité à $\frac{1}{11}$. Le choix de la seconde position à modifier se fait, comme précédemment, par un tirage uniforme sur l'ensemble des voisins de la première position.

3.5 Label

Dans certaines situations, le fichier de configuration de proteus peut devenir particulièrement grand. C'est le cas par exemple lorsque les énergies de références y sont définies. En effet, cela nécessite la déclaration d'une énergie pour chaque type d'acide aminé à chaque position de la chaîne polypeptidique du système étudié. Pour simplifier certaines déclarations, la nouvelle balise `<Label>` permet de définir un alias sur un ensemble de mots. La syntaxe est la suivante :

```
<Label>
exposed 11 15 17 19
buried 12 13 14 16 18 20 21
</Label>
```

les labels « exposed » et « buried » peuvent alors être utilisés de la façon suivante :

```
<Ref_Ener>
CYS exposed -1.09
CYS exposed 2.60
TYR buried -4.34
TYR buried -1.92
</Ref_Ener>
```

cela définit les énergies de références pour CYS et TYR pour toutes les positions des deux ensembles.

3.6 Allocation variable de la matrice

Jusqu'à présent, proteus chargeait en mémoire la matrice d'énergie au début de son exécution. Avec l'introduction du GB/FDB la taille des données d'énergie peut être multiplié par un facteur huit environ. Afin de réduire l'usage de la mémoire, la dernière version du programme prend en compte les restrictions de l'espace de recherche déclarées dans le fichier de configuration pour limiter le chargement des énergies à celles nécessaires à l'exploration.

Chapitre 4

Comparing three stochastic search algorithms for CPD

Ce chapitre reprend l'article publié dans « Journal of Chemical Theory and Computation » [70].

Abstract

Computational protein design depends on an energy function and an algorithm to search the sequence/conformation space. We compare three stochastic search algorithms : a heuristic, Monte Carlo (MC), and a Replica Exchange Monte Carlo method (REMC). The heuristic performs a steepest-descent minimization starting from thousands of random starting points. The methods are applied to nine test proteins from three structural families, with a fixed backbone structure, a molecular mechanics energy function, and with 1, 5, 10, 20, 30, or all amino acids allowed to mutate. Results are compared to an exact, “Cost Function Network” method that identifies the global minimum energy conformation (GMEC) in favorable cases. The designed sequences accurately reproduce experimental sequences in the hydrophobic core. The heuristic and REMC agree closely and reproduce the GMEC when it is known, with a few exceptions. Plain MC performs well for most cases, occasionally departing from the GMEC by 3–4 kcal/mol. With REMC, the diversity of the sequences sampled agrees with exact enumeration where the latter is possible : up to 2 kcal/mol above the GMEC. Beyond, room temperature replicas sample sequences up to 10 kcal/mol above the GMEC, providing thermal averages and a solution to the inverse protein folding problem.

4.1 Introduction

Computational protein design (CPD) has developed into an important tool for biotechnology [71–76]. Starting from a 3D structural model, CPD explores a large space of possible sequences and conformations, to identify protein variants that have certain predefined properties, such as stability or ligand binding. Conformational space is usually defined by a library of sidechain rotamers, which can be discrete or continuous, and by a finite set of backbone conformations or a specific repertoire of allowed backbone deformations. The energy function usually combines physical and empirical terms [77–79]. Both solvent and the unfolded protein state are described implicitly.

The number of amino acid positions that are allowed to mutate can vary, depending on the problem of interest, from 2 or 3 to several dozen, or even more (especially if a very simple and approximate energy function is used). Thus, the combinatorial complexity can be enormous, so that speed is important, as well as accuracy. In addition, it is usually important to identify not one but several high-scoring sequences, for at least three reasons. First, if the typical error in the energy function is σ , we should enumerate all the possible sequences/structures within 1 or 2 σ of the optimal one. Second, it may be of interest to characterize the diversity of a sequence family, by enumerating sets of sequences compatible with its backbone fold (the “inverse folding problem”) [2, 80–83]. Third, we may want to compute properties that are averaged over structural and possibly sequence fluctuations at a given temperature, which requires that we explore solutions within the thermal range. An example is the calculation of ligand binding constants, following a method introduced recently [10, 46]. Calculation of acid/base constants by constant-pH Monte Carlo (MC) is another example, which can be seen as a subproblem of CPD, with sidechain protonation state changes treated as mutations [84–86].

Thus, the complexity and cost of a CPD calculation will depend on several factors. While energy calculations usually represent the bulk of the cost, the power and efficiency of the exploration method are also very important. Several exploration methods exist that can identify exactly the global minimum energy sequence and conformation (GMEC). These include “dead end elimination” methods, or DEE [87, 14], branch-and-bound methods [88, 89], and cost function network methods [40, 39]. While some of these methods can handle large problems, they usually cannot enumerate suboptimal solutions within a large interval σ_E above the GMEC (more than a few kcal/mol). Partly for this reason, stochastic methods remain popular, such MC [90, 78]. MC has two advantages. First, in the limit of a long simulation and with an appropriate move set and move acceptance scheme, we expect this method to sample sequence/conformation from a Boltzmann distribution. Second, MC can be readily combined with enhanced sampling methods developed in the broader field of molecular simulations, such as Replica Exchange or umbrella sampling [91, 92].

Our goal here is to assess three stochastic exploration methods for a series of CPD problems of increasing complexity. The first method is a heuristic method that is not guaranteed to find the global minimum energy conformation, or GMEC, but has been

effective in applications [41, 82, 83]. It performs steepest-descent minimizations starting from thousands of different random configurations, yielding a large number of low energy sequences. The second method is a MC exploration. To obtain Boltzmann sampling with MC, rather weak conditions are required. Specifically, if the sequence/conformation space is finite (as here), if any two states in this space can be connected by moves from the allowed move set (as here), if the move scheme is time-independent (as here), if we assume there are no periodic series of states that can trap the system, and if the move probability has a simple reversibility property (the Kolmogorov condition [93], verified here), then a very long trajectory is guaranteed to converge to a stationary state and obey the so-called detailed balance condition [91] (see below). This in turn makes it easy to design a move acceptance scheme that leads to Boltzmann statistics, such as the classic Metropolis scheme [94, 95]. The third method is an enhanced, multiwalker MC, which performs “replica exchange” [96–98]. Several walkers, or replicas are propagated at different temperatures, and exchange conformations at regular intervals according to a MC test. We refer to it as Replica Exchange Monte Carlo (REMC). These methods are also compared to a fourth method that is exact, in the sense that it can provably identify the GMEC in favorable cases [40, 39]. It is based on ‘cost function networks’, or CFN, where the cost function is the energy, and the network refers to the set of interacting amino acids. The CFN method uses a depth-first branch-and-bound search through a tree of rotamer assignments, with fast integer arithmetic for the energy evaluations and sophisticated tree pruning operations. It can also enumerate all the sequence/conformation combinations within a given energy range δE (not too large) above the GMEC. It is implemented in the Toulbar2 program, by Schiex and coworkers. Other exact methods exist, some of which appear to be even faster than CFN [89]. Our goal, however, is not to “rank” the stochastic and exact methods, but rather to compare our three stochastic methods to each other, and this is facilitated if an exact enumeration of low energy states has also been done.

We use a CPD model that is rather simple but representative of a large class of applications. We use a fixed backbone structure, a discrete set of sidechain rotamers and we assume that the energy function is pairwise additive; that is, the energy has the form of a sum over residue pairs [99–101]. With these simplifications, all possible residue pair interactions can be computed ahead of time and stored in a lookup table [102]; exploration is then done in a second stage. Thus, the cost of energy calculations and sequence/structure exploration are well-separated. The method is implemented in the Proteus CPD package [100, 101] (except for the CFN sequence exploration, done with Toulbar2). Our MC framework is presented in some detail below; the other methods are recalled more briefly.

We considered nine test proteins from three structural classes : SH3, SH2, and PDZ domains. The sequence identity between representatives of the same family is in the range 25–38%, with two exceptions (SH2 domains 1A81, 1M61 have 57% identity, PDZ domains 2BYG, 1R6J have 19% identity). For each protein, we chose different numbers and sets of residues to mutate and we applied the different exploration methods, using

several possible parameterizations for each one. To characterize the different methods, we compared their speed, their ability to identify the GMEC, and their sampling of suboptimal sequences/conformations above the GMEC. The designed sequences were characterized by computing their similarity to natural sequences, their classification by the Superfamily fold recognition tool [103, 104], and their sequence entropies ; they are shown to agree rather well with natural sequences of the corresponding families. Overall, the heuristic method is the most successful in identifying low energy solutions, while REMC is almost as successful but has the advantage of sampling from a Boltzmann distribution over a large energy range, yielding thermal averages.

4.2 Methods

4.2.1 Monte Carlo : general framework

We consider a polypeptide of n amino acids. Its sequence S is written $S = t_1 t_2 \cdots t_n$, where t_i is the sidechain type of amino acid i . We assume that each amino acid i can take on a few different types t, t', \dots that form a set T_i . For each sequence, there are two classes of structures : folded and unfolded. For the folded form, all the sequences S share the same, precise geometry for the polypeptide backbone ; only the sidechain positions can vary. Specifically, the sidechain of each amino acid i can explore a few discrete conformations or rotamers r, r', \dots (around 10 per type t_i). The folded energy is E_f . The structure of the unfolded form is not specified ; the energy is assumed to be independent of the particular unfolded structure, and to have the additive form :

$$E_u(S) = \sum_{i=1}^n E_u(t_i) = \sum_{i=1}^n (e_u(t_i) - kT \log n_u(t_i)), \quad (4.1)$$

where $E_u(t_i)$ is a free energy associated with sidechain type t_i in the unfolded state, and the rightmost term separates it into an energy component $e_u(t_i)$ and a conformational entropy term, where kT is the thermal energy and $n_u(t_i)$ is the number of conformations or rotamers available to sidechain type t_i in the unfolded state.

We perform a Monte Carlo simulation [94, 95] where one copy of the folded protein is explicitly represented. The unfolded state is included implicitly, by propagating the simulation with the energy function $E_M = E_f - E_u$ (the folding energy). One possible elementary MC move is to change a rotamer r_i in the current folded sequence ; the energy change is $\Delta E_M = \Delta E_f = E(\dots t_i, r'_i \dots) - E(\dots t_i, r_i \dots)$. Another possible move is a mutation : we modify the sidechain type $t_i \rightarrow t'_i$ at a chosen position i in the folded protein, assigning a particular rotamer r'_i to the new sidechain. The energy change is

$$\Delta E_M = \Delta E_f - \Delta E_u = (E_f(\dots t'_i, r'_i \dots) - E_f(\dots t_i, r_i \dots)) - (E_u(t'_i) - E_u(t_i)) \quad (4.2)$$

ΔE_M measures the stability change due to the mutation (for the given set of rotamers) ; it is as if we performed the reverse mutation $t'_i \rightarrow t_i$ in the unfolded form.

If the moves are generated and accepted with an appropriate Metropolis-like scheme, the Markov chain will visit states according to their Boltzmann probability :

$$p_M(S,c) = \frac{e^{-\beta(E_f(S,c)-E_u(S))}}{\sum_{S'} \sum_{c'} e^{-\beta(E_f(S',c')-E_u(S'))}} \quad (4.3)$$

where $\beta = 1/kT$ and the subscript M indicates probabilities sampled by the Markov chain. For two conformations c, c' of sequence S, the Markov probability ratio is $p_M(S,c)/p_M(S,c') = e^{-\beta(E_f(S,c)-E_f(S,c'))}$. For two sequences S, S', the probability ratio is

$$\frac{p_M(S)}{p_M(S')} = \frac{\sum_c e^{-\beta(E_f(S,c)-E_u(S))}}{\sum_{c'} e^{-\beta(E_f(S',c')-E_u(S'))}} = \frac{e^{-\beta\Delta G_{\text{fold}}(S)}}{e^{-\beta\Delta G_{\text{fold}}(S')}} \quad (4.4)$$

In the ratio of Markov probabilities, we recognize the ratio of Boltzmann factors for S and S' folding, so that we have the second equality, where $\Delta G_{\text{fold}}(S)$ denotes the folding free energy of sequence S (respectively, S').

Eq. (4.4) has a simple interpretation : the Markov chain, with the chosen energy function $E_M = E_f - E_u$ and appropriate move probabilities, leads to the same distribution of states as a macroscopic, equilibrium, physical system where all sequences S, S', ... are present at equal concentrations, and are distributed between their folded and unfolded states according to their relative stabilities. This is exactly the experimental system we want our Markov chain to mimic. In this interpretation, a MC mutation move $S \rightarrow S'$ amounts to unfolding one copy of S and refolding one copy of S'.

It remains to specify the move generation probabilities and choose an appropriate acceptance scheme [94, 95]. Let $\alpha(o \rightarrow n)$ be the probability to select a trial move between two states o and n and $acc(o \rightarrow n)$ the probability to accept it. Under weak assumptions, the simulation obeys the so-called detailed balance condition :

$$N(o)\pi(o \rightarrow n) = N(n)\pi(n \rightarrow o), \quad (4.5)$$

where $N(o), N(n)$ are the equilibrium populations of states o and n and $\pi(o \rightarrow n)$ is the probability to make the $o \rightarrow n$ move. Detailed balance is guaranteed in the limit of a very long simulation if the move set allows us to connect any two states in the sequence/conformation space (as here), if the system is “aperiodic” (there are no series of states that can form “periodic orbits”, trapping the system indefinitely), and if the so-called Kolmogorov reversibility condition is verified (as here) : the products of probabilities around closed loops of states are the same in both directions. We cannot prove the aperiodic condition, but it appears very likely and is treated as an assumption. To produce Boltzmann statistics, we choose the acceptance probabilities [94, 95] :

$$acc(o \rightarrow n) = \exp(-\beta\Delta E_M) \frac{\alpha(n \rightarrow o)}{\alpha(o \rightarrow n)} \quad \text{if } \Delta E_M > 0; 1 \text{ otherwise} \quad (4.6)$$

where $\Delta E_M = E_M(n) - E_M(o)$ is the $o \rightarrow n$ energy difference. Notice that this scheme obeys the Kolmogorov reversibility condition.

For a rotamer move at a particular position in the polypeptide chain, of type t , we define the move generation probability as $\alpha(o \rightarrow n) = \frac{1}{n_f(t)} = \alpha(o \rightarrow n)$; all possible choices for the new rotamer are equiprobable, forward and backward rotamer moves have the same generation probability, and eq. (4.6) reduces to the simple Metropolis test [94].

For a mutation move at a particular position, we define $\alpha(o \rightarrow n)$ as follows :

- (a) select a new type t' with equal probabilities $\alpha_t(o \rightarrow n) = \frac{1}{N}$ for all N possible types ;
- (b) choose a rotamer r' for the new sidechain with equal probabilities $\alpha_r(o \rightarrow n) = \frac{1}{n_f(t')}$ for all $n_f(t')$ possible folded state rotamers.

The overall probability is therefore

$$\alpha(o \rightarrow n) = \alpha_t(o \rightarrow n)\alpha_r(o \rightarrow n) = \frac{1}{Nn_f(t')} \quad (4.7)$$

The $o \rightarrow n$ and $n \rightarrow o$ probabilities are different whenever the old and new sidechain types have different numbers of possible rotamers. With these move probabilities, the mutation acceptance probability can be written :

$$acc(t \rightarrow t') = e^{-\beta(\Delta E_f - \Delta E_u)} \frac{n_f(t)}{n_f(t')} = e^{-\beta(\Delta E_f - \Delta E_u)} \frac{n_f(t)n_u(t')}{n_u(t)n_f(t')} \text{ if } \Delta E_M > 0 \quad (4.8)$$

$$= 1 \text{ otherwise} \quad (4.9)$$

If the number of rotamers in the folded and unfolded states are the same, $n_u = n_f$, the fraction on the right will cancel out. However, the rotamer numbers also appear in the energy change that determines whether the move is uphill, ΔE_M .

With REMC, several simulations (“replicas” or “walkers”) are propagated in parallel, at different temperatures ; periodic swaps are attempted between two walkers’s conformations. The swap is accepted with probability

$$acc(t \rightarrow t') = \text{Min} \left[1, e^{(\beta_i - \beta_j)(\Delta E_i - \Delta E_j)} \right] \quad (4.10)$$

where β_i, β_j are the inverse temperatures of the two walkers and $\Delta E_i, \Delta E_j$ are the changes in their folding energies, due to the conformation change [97, 98].

4.2.2 MC and REMC : implementation details

For plain MC, we use one- and two-position moves, where either rotamers, types, or both are changed. For two-position moves, the second position is selected among those that have a significant (unsigned) interaction energy with the first one, meaning that there is at least one rotamer combination where their interaction is 10 kcal/mol or more. For REMC, we use four or eight walkers, with thermal energies kT_i that range from 0.125 to 2 or 3 kcal/mol, and are spaced in a geometric progression : $T_{i+1}/T_i = \text{constant}$, following

Kofke [97]. Conformation swaps are attempted at regular intervals, between walkers at adjacent temperatures. The precise parameter settings are given in table 4.1.

Table 4.1 – Selected MC and REMC protocols

name	walker temperature(s) kT (kcal/mol)	trajectory length (steps)	move probabilities ^a rot ; mut ; mut+rot ; mut+mut ;	walker swap periodicity ^b
MC	0.2	$6 \cdot 10^9$	0 ; 1 ; 0.1 ; 0	-
REMCa	0.125 ; 0.25 ; 0.5 ; 1	$1.5 \cdot 10^9$	1 ; 0 ; 0.1 ; 0	$7.5 \cdot 10^6$
REMCb	0.25 ; 0.5 ; 1 ; 2	$1.5 \cdot 10^9$	1 ; 0 ; 0.1 ; 0	$7.5 \cdot 10^6$
REMCC	0.175 ; 0.263 ; 0.395 ; 0.592 ; 0.888 ; 1.333 ; 2 ; 3	$0.75 \cdot 10^9$	0 ; 1 ; 0.1 ; 0	$7.5 \cdot 10^6$
REMCD	0.175 ; 0.263 ; 0.395 ; 0.592 ; 0.888 ; 1.333 ; 2 ; 3	$0.75 \cdot 10^9$	1 ; 0 ; 0.1 ; 0	$7.5 \cdot 10^6$
REMCE	0.175 ; 0.263 ; 0.395 ; 0.592 ; 0.888 ; 1.333 ; 2 ; 3	$0.75 \cdot 10^9$	1 ; 0 ; 0.1 ; 0	10^6

^aProbabilities at each MC step to change, respectively, a rotamer ; a sidechain type ; a type at one position and a rotamer at another ; a type at two positions. ^bThe interval between attempts to exchange states between two walkers (using a Metropolis test).

4.2.3 Heuristic sequence optimization

The heuristic sequence optimization uses a multistart, steepest-descent minimization, where thousands of random starting configurations are minimized, leading to a large collection of local energy minima [41, 100]. One “heuristic cycle” proceeds as follows : an initial amino acid sequence and set of sidechain rotamers are chosen randomly. These are improved in a stepwise way. At a given amino acid position i , the best amino acid type and rotamer are selected, with the rest of the sequence held fixed. The same is done for the following position $i + 1$, and so on, performing multiple passes over the amino acid sequence until the energy no longer improves or a set, large number of passes is reached (500 passes). The final sequence, rotamer set, and energy are output, ending the cycle. Below, we typically perform ~ 100.000 heuristic cycles for each protein.

4.2.4 Cost function network method

The CFN method is implemented in the Toulbar2 program [40, 39]. The Proteus energy matrices are converted to the Toulbar format with a perl script. With this format, all the interaction energies are approximated as positive integers, without loss of generality. We used Toulbar2 version 0.9.7.0 with a recommended parameterization (options -l=3 -m -d : -s) ; for the unsuccessful cases (GMEC not identified) we systematically repeated calculations with version 0.9.6.0 and a more aggressive protocol (options -l=1 -dee=1 -m -d : -s). To enumerate sequence/conformation pairs that have energies higher than the

GMEC, Toulbar2 is run with the “suboptimal” option and an energy threshold. Available memory was limited to 30 gigabytes. For a few of the unsuccessful CFN tests, we tried a third protocol, described recently [105], using the most recent version 0.9.8. Results were similar to those obtained with the second protocol ; that is, only a few new successes were obtained with the third protocol.

4.2.5 Energy function

The energy function has the form :

$$E = E_{\text{bonds}} + E_{\text{angles}} + E_{\text{dihe}} + E_{\text{impr}} + E_{\text{vdw}} + E_{\text{Coul}} + E_{\text{solv}} \quad (4.11)$$

The first six terms represent the protein internal energy. They were taken from the Charmm19 empirical energy function [18]. The last term represents the contribution of solvent. We used a “Coulomb + Accessible Surface Area”, or CASA implicit solvent model [52, 99] :

$$E_{\text{solv}} = \left(\frac{1}{\epsilon} - 1 \right) E_{\text{Coul}} + \sum_i \sigma_i A_i \quad (4.12)$$

Here, ϵ is a dielectric constant that scales the Coulomb energy, set to 23, following earlier tests [99]. A_i is the solvent accessible surface area of atom i ; σ_i is a parameter that reflects each atom’s preference to be exposed or hidden from solvent. The solute atoms were divided into four groups with the following σ_i values (cal/mol/Å²) : unpolar (-5), aromatic (-40), polar (-8) and ionic (-10). Hydrogen atoms were assigned a surface coefficient of zero. Surface areas were computed by the Lee and Richards algorithm [106], using a 1.5 Å probe radius. Pairwise additivity errors for the surface energy term were corrected by applying a reduction factor of 0.5 to buried pairs [51, 52]. Energy calculations were done with a modified version of the Xplor program [47, 101].

The energies $E_u(t)$ associated with the unfolded state were determined empirically to give reasonable amino acid compositions for the protein families considered here [100] ; they are reported in table 4.2.

Table 4.2 – Reference energies for unfolded state (kcal/mol).

Ala	-8.519	Ile	-10.530
Arg	-22.107	Leu	-12.227
Asn	-16.715	Lys	-20.999
Asp	-19.699	Met	-11.985
Cys	-9.426	Phe	-18.017
Gln	-17.718	Ser	-11.881
Glu	-20.048	Thr	-11.413
His(d)	-17.824	Trp	-19.267
His(e)	-17.824	Tyr	-20.893
His+	-21.704	Val	-9.5654

Three His variants are given protonated on ND1 (d), NE2 (e), or both (+).

4.2.6 Test systems and preparation

We considered nine protein domains, from the SH3, SH2, and PDZ families, listed in table 4.3. Each domain is known to fold stably and has an associated crystal structure used for our calculations. Systems were prepared and energy matrices computed using procedures described previously [82, 83]. Briefly, each PDB structure was minimized through 200 steps of conjugate gradient minimization. For each residue pair, interaction energies were computed after 15 steps of energy minimization, with the backbone fixed and only the interactions of the pair with each other and the backbone included. Sidechain rotamers were described by a slightly expanded version of the library of Tuffery et al [56], which has a total of 254 rotamers (sum over all amino acid types). The expansion consists in allowing additional hydrogen orientations for OH and SH groups [31]. This rotamer library was chosen for its simplicity and because it gives very good performance in sidechain placement tests, comparable to the specialized Scwrl4 program (which uses a much larger library) [57, 58].

For each protein, we ran tests with 1, 5, 10, 20, 30 or all positions allowed to mutate. The positions selected are listed in 4.4. For each protein, the sets of 5–30 designed positions were chosen randomly, in five different ways. The choice was constrained so that all the designed positions have a significant interaction with all others. Specifically, for each pair of designed positions, there should be at least one rotamer combination where the (unsigned) interaction energy is above a chosen threshold E_{min} . The threshold was 10 kcal/mol for the 5- and 10-position tests. It was 1 kcal/mol for the 20- and 30-position tests. With this criterion, the sets of designed positions are all quite compact, and highly coupled.

Table 4.3 – Test proteins

type	PDB	length	acronym	type	PDB	length	acronym
PDZ	1G9O	91	NHERF	SH2	1A81	108	Syk kinase
PDZ	1R6J	82	syntenin	SH2	1BM2	98	Grb2
PDZ	2BYG	97	DLH2	SH2	1M61	109	Zap70
SH3	1ABO	58	Abl	SH2	1O4C	104	Src kinase
SH3	1CKA	57	c-Crk				

4.2.7 Sequence characterization

Designed sequences were compared to the Pfam alignment for the corresponding family, using the Blosum40 scoring matrix and a gap penalty of -6. Each Pfam sequence was also compared to its own Pfam alignment. For these Pfam/Pfam comparisons, if a test protein T was part of the Pfam alignment, the T/T self comparison was left out, to be more consistent with the designed/Pfam comparisons. If the test protein T was not part of the Pfam alignment, we used Blast to identify its closest Pfam homologue H and left the T/H

comparison out, for consistency. The Pfam alignments were either the “seed” alignment for each family (around 50 sequences) or much larger, “full” alignments, with 6287, 3052, and 14944 sequences, respectively, for the SH3, SH2, and PDZ families. Similarities were computed for protein core residues, defined by their near-complete burial, and listed in Results.

Designed sequences were submitted to the Superfamily library of Hidden Markov Models [103, 104], which attempts to classify sequences according to the SCOP classification [60]. Classification was based on SCOP version 1.75 and version 3.5 of the Superfamily tools. Superfamily executes the hmmscan program, which implements a Hidden Markov model for each SCOP family and superfamily; here hmmscan was executed with an E-value threshold of 10^{-10} , using a total of 15438 models to represent the SCOP database.

To compare the diversity in the designed sequences with the diversity in natural sequences, we used a standard, position-dependent sequence entropy [68], computed as follows :

$$S_i = - \sum_{i=1}^6 f_j(i) \ln f_j(i) \quad (4.13)$$

where $f_j(i)$ is the frequency of residue type j at position i , either in the designed sequences or in the natural sequences (organized into a multiple alignment). Instead of the usual, 20 amino acid types, we employ six residue types, corresponding to the following groups : {LVIMC}, {FYW}, {G}, {ASTP}, {EDNQ}, and {KRH}. This classification was obtained by a cluster analysis of the BLOSUM62 matrix [107], and also by analyzing residue-residue contact energies in proteins [69]. To get a sense of how many amino acid types appear at a specific position i , we usually report the residue entropy in its exponentiated form, $\exp(S_i)$, which ranges from 1 to 6.

4.3 Results

Our main focus is to characterize sequence/structure exploration methods and their ability to sample low energy sequences. We begin, however, by showing that the sequences we sample are similar to experimental ones. Indeed, the performance of exploration methods depends on the shape and ruggedness of the energy surface, and should be tested in situations where the energy function is sufficiently realistic, as judged by the quality of the designed sequences. After that, we compare the ability of the four exploration methods to identify low energy sequences, including the GMEC. Finally, we consider the diversity of sequence sets, or density of states sampled by each method.

4.3.1 Quality of the designed sequences

We first report information on the quality of our designed sequences. We use sets of REMC sequences to illustrate the main features. The best REMC protocol, REMCd (table 4.1) was used. Results with the other exploration methods are expected to be

similar. Indeed, while the methods sometimes exhibit differences of up to a few kcal/mol between their best sequences, the average sequence quality of the 100-1000 best sequences is typically similar between methods. Table 4.4 summarizes results for our nine test proteins in design calculations where all positions were allowed to change types. All mutations were allowed, except mutations to/from Gly and Pro, since these are likely to change the backbone structure.



Figure 4.1 – Sequence logos for the core region of two designed proteins : 1CKA (SH3) and 2BYG (PDZ). The low energy CPD sequences are compared to the sequences of the full Pfam collection of experimental sequences. Positions shown correspond to the hydrophobic core of each protein ; residue numbers are indicated (PDB numbering).

REMC was done with height replicas at temperatures between 0.175 and $3 kT$ units. Simulation lengths were 750 million steps (per replica). The top 10000 sequence/conformation combinations were retained, corresponding to 200–400 unique sequences. For eight of the nine test proteins, all the retained sequences were recognized as members of the correct superfamily and family, with match lengths and E-values given in table 4.4. Thus, our designed sequences are mostly similar to experimental ones. Sequence identities to wildtype are 31% on average (table 4.4), similar to earlier studies with the same energy function [82, 83]. The good agreement with experiment is also illustrated by computing similarity scores between the designed sequences and sequences from the Pfam database. For the protein core region, the similarity is similar to that between experimental sequences, as

shown in figure 4.1. Notice that we use here a simple CASA solvent model, since our focus is not the quality of the designed sequences, but the performance of our search algorithms. With a more sophisticated Generalized Born solvent model and a more highly optimized unfolded state model, sequence quality would be even better (manuscript in preparation).

Representative sequence logos for the protein core are shown in Fig. 4.1, illustrating the agreement between designed and experimental sequences. For the SH3 protein 1CKA, the design mostly recovers the native sidechain type, or a homologous type that is common in other natural sequences from Pfam. Exceptions include position 170, where we obtain I or V, whereas the native type is W ; however, L and V are also occasionally found in Pfam. At position 139, we sample R instead of the native A or V, found in Pfam ; however, the hydrophobic part of the designed Arg sidechain is homologous to A and V, while its ionized tip actually sits outside the hydrophobic core, at the protein surface. Finally, at position 143, we sample W, which is homologous to Y and F found in Pfam. For the PDZ domain 2BYG, agreement with the native and Pfam sequences is similar, with a few departures : at positions 207 and 245, we sample W instead of the natural types I, L, V, P ; at position 253, we sample H, compared to mostly I, L, V, and sometimes F in Pfam ; at position 265,we always sample Y, whereas the Pfam types are diverse (and mostly hydrophobic).

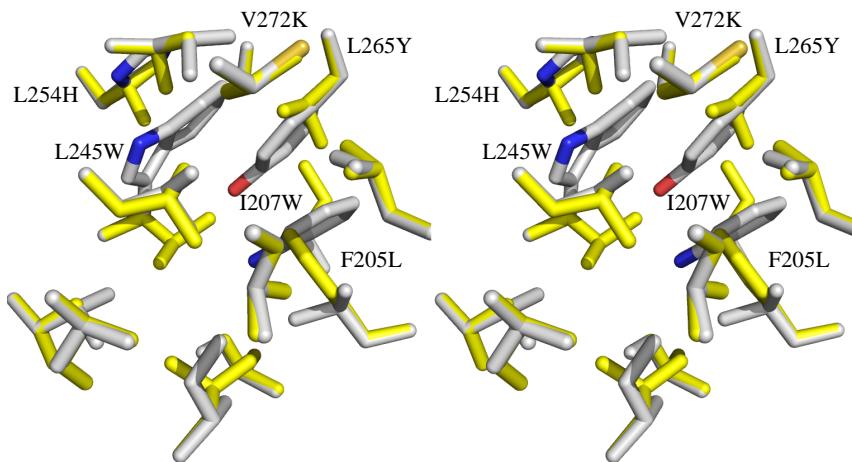


Figure 4.2 – The lowest energy sequence/structure combination is shown for the 2BYG PDZ domain (stereo view ; selected sidechains only ; white sticks), superimposed on the Xray structure (yellow sticks) ; the sidechains shown are the ones in the sequence logo, Fig.2. The main mutated positions are labelled ; other positions are either not mutated or mutated within the ILV group. Figure produced with pymol [1]

For the same PDZ domain, the lowest energy sequence/structure combination is shown in figure 4.2, superimposed on the Xray structure ; only the hydrophobic core sidechains are shown, for clarity. Most of the native/designed sidechains overlap very well, although the double mutation (F205L, L245W) leads to some local repacking.

For the 1A81 SH2 domain, the lowest energy sequences are not recognized by Superfamily, but if we assay sequences that are 8-12 kcal/mol above the GMEC, about 10% are correctly recognized by Superfamily as SH3 sequences. Similarly, if the top 10,000 sequences produced by the heuristic algorithm are tested, 57% of them are recognized by Superfamily as SH3 family members. The heuristic sequences are also 8-12 kcal/mol above the GMEC (see below). This protein illustrates the imperfect parameterization of our energy function, with the consequence that higher energy sequences can actually be more realistic, in some cases, than lower energy ones. Thus, the significance of the GMEC is only as good as the energy function.

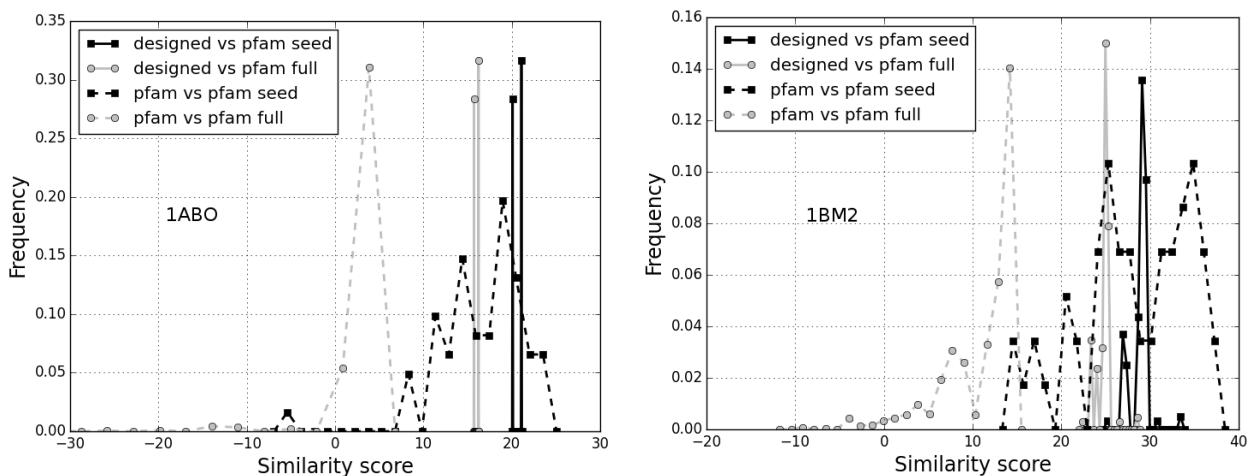


Figure 4.3 – Histogram of Blosum40 similarity scores to Pfam sequences for the core region of two designed proteins : 1ABO (SH3) and 1BM2 (SH2). The similarity between Pfam sequences is also shown, considering either the Pfam seed alignment or the much larger full alignment.

Table 4.4 – Designed sequence quality measures

Protein	Number of sequences tested	Identity % to wildtype	Superfamily tests				
			Match length	Superfamily E-value	Superfamily success rate	Family E-value	Family success rate
1A81	236	27	none				
1ABO	203	32	51/58	4.4e-4	100%	2.8e-3	100%
1BM2	209	27	78/98	4.2e-5	100%	2.6e-3	100%
1CKA	416	33	40/57	1.1e-5	100%	3.4e-3	100%
1G9O	338	36	79/91	7.0e-7	100%	2.5e-3	100%
1M61	405	42	97/109	7.2e-7	100%	2.6e-4	100%
1O4C	274	21	95/104	2.1e-4	100%	4.5e-3	100%
1R6J	270	34	74/82	9.8e-6	100%	4.6e-3	100%
2BYG	426	28	59/97	1.4e-5	100%	7.1e-3	100%

4.3.2 Finding the GMEC

CPU and memory limits for each method

The ability of an exploration method to sample low energy sequences depends on the CPU and memory ressources available, as well as on detailed parameterization choices. Here, we set somewhat arbitrary limits, to remain within a practical run situation. For CFN, we set a maximum time limit of 24 h and a memory limit of 30 gbytes (gb). For the heuristic method, we used 110,000 heuristic cycles, increased to 330,000 or 990,000 cycles in a few cases; even for these cases, run times did not exceed 24 h. For MC, we ran up to 10^9 simulation steps, which corresponded to CPU times of 9 h at most. For REMC, we ran $0.75 \cdot 10^9$ simulation steps per replica, with a few exceptions. We used an OpenMP, shared memory parallelization on a single processor, with one replica per core. Total CPU time per core was never more than 3 hours, for a total CPU use of less than 24 h. For the heuristic, MC, and REMC methods, memory requirements are modest ; about 2 gb for the largest calculations. Run times are shown in Fig. 4.4 as a function of the number of designed positions, which varies from one position to the entire protein (about 90 positions). For comparison, the CPU time needed to compute the energy matrix for a single pair of designed positions, using an advanced energy function (Generalized Born solvent plus a sophisticated surface area term) and a single core of a recent Intel processor is about 5 hours.

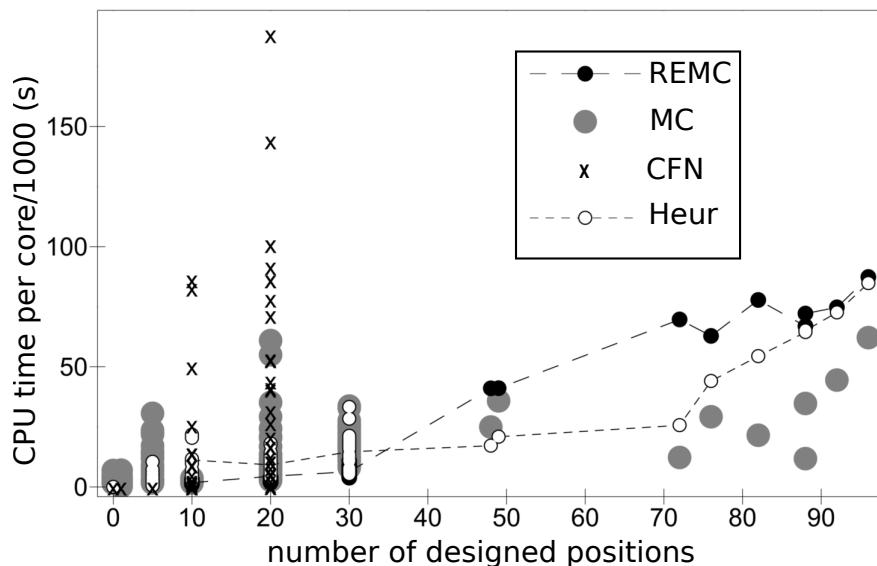


Figure 4.4 – Run times for different test calculations and search methods. CPU times per core are shown ; only the REMC calculations use multiple cores (OpenMP parallelization). CFN results are only given for 20 designed positions or fewer. REMC was only done for the larger calculations. Results are shown for a large, representative subset of the test calculations. A few CFN calculations were allowed to run beyond the 24h CPU limit. For clarity, the average Heur and REMC values are included as dashed lines.

The MC and REMC methods require choices of move probabilities and temperatures, which affect the sampling in ways that vary from protein to protein. Fig. 4.5 shows the lowest energy sampled with a small collection of protocols : one heuristic, one MC, and five REMC protocols, applied to our nine proteins, with all positions allowed to mutate (except Gly/Pro). The protocol details are given in table 4.1. For these large design problems, the GMEC is not known. Instead, for each protein, the overall best energy (the best of the seven protocols) is taken as the reference, or zero value.

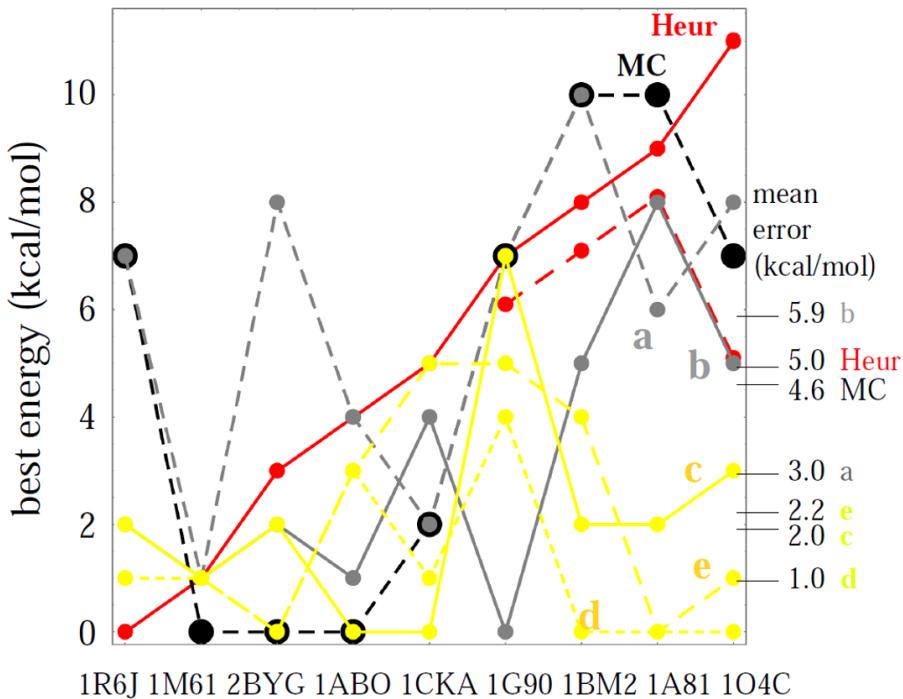


Figure 4.5 – Comparison between selected heuristic, MC, and REMC protocols for whole protein design. The best energy obtained with each protocol is shown for the nine test proteins. Zero represents the overall best energy for each protein. The mean “error” is indicated on the right for each protocol (the mean difference from the overall best energy). The protocols are a heuristic method (Heur), an MC method, two REMC protocols with 4 walkers (REMC a, b), and three REMC protocols with 8 walkers (yellow lines and dots, REMC c, d, e). Details of the protocols are in table 4.1 ; each curve is labelled according to the protocol name. The dashed red line shows heuristic results with a more aggressive protocol (330,000 heuristic cycles) for the four rightmost proteins.

For a given protein, the best energy varies by up to 12 kcal/mol from one protocol to another (compare the 1BM2 REMCa and REMCc energies or the 1CKA MC and REMCd energies). For each protocol, the best energy obtained was averaged over the nine proteins, giving a mean “error”; these values are also reported in Fig. 4.5. The lowest mean error is 1.0 kcal/mol, with REMCd. In other words, REMCd gives a best energy that is, on average, 1.0 kcal/mol above the overall best energy. Based on these and other similar tests, for the rest of this work, we used the specific MC protocol in table 4.1 and the REMCd replica exchange protocol, which are generally good but not necessarily optimal for every situation.

Optimal sequences/structures with up to 10 designed positions

As our first series of tests, we did calculations for each test protein with zero, one, or five designed positions. Results are summarized in Fig. 4.5. With zero positions, only rotamers are optimized (at all positions in the protein). With one, we systematically designed each position of each protein in turn (plus rotamers at all positions). With 5, we picked the positions randomly, close together in the structure, in five different ways, for a total of 45 tests. Two positions were considered close by if they have at least one rotamer combination that gives an interaction energy of 10 kcal/mol or more (in absolute magnitude; eg, steric overlap). In all these cases, CFN found the GMEC very rapidly (seconds); the heuristic also found the GMEC, with much longer run times (an hour). MC found the GMEC in all but a few cases (Fig. 4.6), with run times of a few minutes.

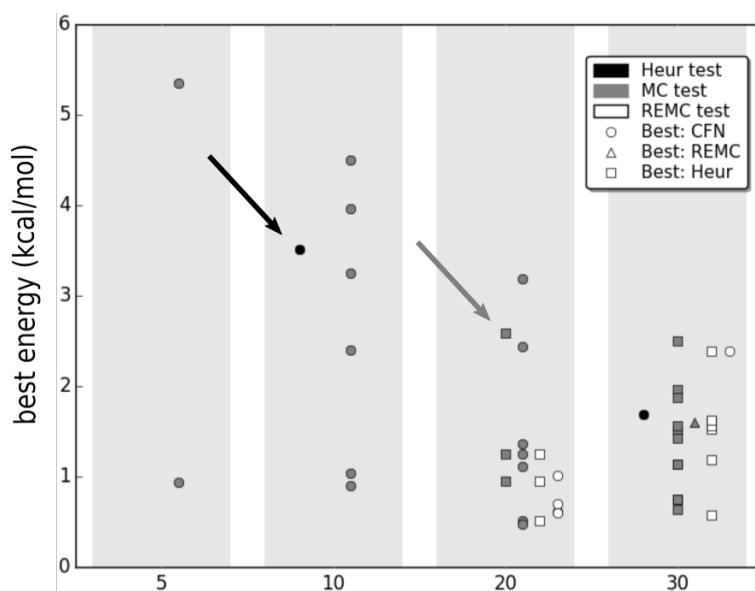


Figure 4.6 – The lowest energies obtained with the different exploration methods for 5-, 10-, 20-, and 30-position design. Differences with respect to the GMEC or the overall best energy are shown, excluding the smallest values (less than 0.4 kcal/mol above the best energy). The color of each point indicates the nature of the test; its shape indicates which method gave the best energy. Two examples are highlighted by arrows: the black arrow shows the heuristic result for a 10-position test where the best energy was given by CFN; the gray arrow shows the MC result for a 20-position test where the best energy was given by the heuristic.

As a second series of tests, we chose randomly for each protein a set of 10 positions to design; the other positions had fixed types but explored all possible rotamers. The selected positions were close by in the protein structure. For each protein, we made five separate choices of positions to design, for a total of 45 test cases. The CFN, heuristic, and MC methods were run for all 45 cases; REMC was run only when MC gave a poor result (6 cases, involving 5 proteins). Results are summarized in Fig. 4.6 and table 4.5. Twenty cases where all methods found the GMEC are not listed in the table, leaving 25 where at least one method did not find the GMEC. CFN performed very well: only in one case

did it not find the GMEC. The lowest energy was sampled in this case with the heuristic, and the best CFN energy was 5.7 kcal/mol higher (despite using the more aggressive CFN protocol). With a third, more recent CFN protocol [105], this result did not change.

The heuristic performed about as well as CFN for 10-position design. In one case, CFN did not find the GMEC and the heuristic gave the lowest energy (2BYG-1). In 39 cases, the heuristic found the GMEC. In three cases, it was within 0.15 kcal/mol of the GMEC, with no mutations (only rotamer differences). In one case (1CKA-5), it was 0.29 kcal/mol above the GMEC, with no mutations. Tripling the number of heuristic cycles allowed the GMEC to be reached (within 0.07 kcal/mol) in all these cases, with run times below 6 h. There was only one real failure, 1M61-2, where the best heuristic solution was 3.5 kcal/mol above the GMEC, with three mutations relative to the GMEC. For this case, the GMEC was recovered (within 0.01 kcal/mol) if the number of cycles was increased to 990,000, for a run time of 7 h. Switching from the heuristic structure (after 330,000 cycles) to the GMEC requires concerted changes in 3 adjacent sidechain positions. This is only possible during a heuristic cycle if there is a downhill, connecting pathway made of single position changes, which is evidently very rare for this particular test. Thus, the heuristic method can only find the GMEC if it draws the right combination of types/rotamers at the very beginning of a cycle; hence the need for 990,000 cycles.

Plain MC did slightly less well for 10-position design. In 21 cases, it found the GMEC. In 18 cases, its best sequence was within 0.2 kcal/mol of the GMEC, with 0–3 mutations (one on average). Notice that 0.2 kcal/mol is the thermal energy for the MC protocol used. In six cases, its best sequence was between 0.9 and 4.5 kcal/mol above the GMEC, with 2–7 mutations (3 on average). For these six cases, REMC was run, and sampled sequences within 0.40 kcal/mol of the GMEC, except for one case where its best sequence was 0.80 kcal/mol above the GMEC. Overall, MC or REMC reached the GMEC to within 0.40 kcal/mol in all but one case. A 0.40 kcal/mol energy difference is actually less than the average pairwise additivity errors in the energy function [52, 99, 31], and so one might consider this performance to be about as good as the CFN and heuristic methods. In terms of speed, for 10-position design, all the methods were comparable (a few hours per run on average).

Optimal sequences with 20 or 30 designed positions

We did similar tests with 20 designed positions, selected randomly in 5 different ways for each protein, as above. Results are given in Fig. 4.6 and table 4.6. CFN found the GMEC in 28 out of 45 cases; in two others, it found the best energy of the four methods. For 6 of these 30 cases, the more aggressive protocol was necessary, and run times were 2–22 h (11 on average). For 14 of the other 15 cases, the best CFN energy was 0.1–7.5 kcal/mol above the best solution found by the other methods, and 2.8 kcal/mol on average, despite using the more aggressive protocol. For the worst case, the CFN energy was 13.9 kcal/mol above the best solution. Of the 17 cases where the CFN was not identified,

Table 4.5 – Tests with 10 designed positions

rotamers ^a	length ^b	Protein	CFN ^c	Heur. ^d	MC	REMC
2991	108(17)	1A81 3	gmec	0.001	0.1595	
		1A81 4	gmec	0.	0.0317	
		1A81 5	gmec	0.	0.0563	
2520	58(8)	1ABO 1	gmec	0.0675	0.9054	0.8041
		1ABO 4	gmec	0.	0.0128	
2957	98(10)	1BM2 1	gmec	0.	0.0950	
		1BM2 5	gmec	0.	0.1082	
		1CKA 5	gmec	0.2859	3.2525	0.
2508	57(8)	1G9O 3	gmec	0.1366	0.1366	
		1G9O 5	gmec	0.	3.9599	0.
		1M61 1	gmec	0.	0.0776	
2957	109(21)	1M61 2	gmec	3.5105	4.5062	0.3215
		1M61 5	gmec	0.	0.0432	
		1O4C 1	gmec	0.	0.1121	
		1O4C 2	gmec	0.	0.1046	
		1O4C 3	gmec	0.	0.1519	
3037	104(8)	1O4C 4	gmec	0.	0.1545	
		1O4C 5	gmec	0.	0.1753	
		1R6J 1	gmec	0.	2.4022	0.3986
		1R6J 2	gmec	0.	1.0398	0.3049
		1R6J 3	gmec	0.	0.0106	
2773	82(10)	1R6J 5	gmec	0.	0.0162	
		2BYG 1	5.7485	0.	0.0337	
		2BYG 3	gmec	0.	0.0833	
		2BYG 4	gmec	0.	0.2149	

^aTotal number of rotamers available to the system. Each designed position can explore 206 rotamers ; the others explore about 10 rotamers each. ^bTotal protein length (number of Gly+Pro in parentheses). ^cgmec indicates the GMEC was successfully identified. ^dFor all four exploration methods and each test, we report the difference between the best energy obtained and the overall best energy (the best over all methods, which may or not be the GMEC). 10-position tests where all four methods found the GMEC are not listed.

half were rerun with a third, more recent CFN protocol [105] ; in just one of these cases (1BM2-4, a very mild failure) did the newer protocol identify the GMEC.

The heuristic method found the GMEC in 22 of the 28 cases where it is known. For the other six cases, it was within 0.40 kcal/mol of the GMEC, with 0–4 mutations (2.7 on average). For the 17 cases where the GMEC was not identified by CFN, the heuristic produced the lowest energy of all methods, except one case (104C-1) where it was 0.35 kcal/mol above CFN. Overall, the heuristic either found the best energy of the four methods or was within 0.40 kcal/mol of the best energy.

MC converged to the best energy in 11 cases ; in 25 other cases, it was within 0.50 kcal/mol of the best energy. In the other nine cases, its best energy was at most 3.2 kcal/mol above the best energy (sampled by the heuristic and/or CFN). Finally, REMC was done for all the test cases. In six cases, its best energy was more than 0.50 kcal/mol from the best energy. However, the differences were notably smaller than for plain MC,

with an average of just 0.8 kcal/mol for the 6 worst cases and a maximum (for 1G90-1) of 1.25 kcal/mol.

The same tests were done with 30 designed positions ; see Fig. 4.6 and table 4.6. CFN found the GMEC in just one case ; in 5 others, it did not find the GMEC but gave the lowest energy overall. In four other cases, it was within 0.50 kcal/mol of the best energy sampled by the other methods. For the other 35 cases, its best energy was higher than the best method, with differences of 10 kcal/mol or more in 20 cases.

The heuristic produced the lowest energy in all but four cases, with differences in those cases of 0.01, 0.10, 0.70, and 1.69 kcal/mol from the best energy. In the last two cases, CFN produced the best energy. Plain MC found the best energy in only 12 cases, but gave only moderate energy errors : in just four cases was its best sequence more than 2 kcal/mol above the overall best energy (differences of 2.2, 2.5, 2.8, and 7.7 kcal/mol). REMC was applied to the 14 cases where the MC errors were largest ; in four of these it reduced the error to 0.6 kcal/mol or less. The largest MC error was reduced from 7.7 to 2.4 kcal/mol. Doubling the REMC trajectory length (to 1.5 billion steps, 1G9O-5) reduced the largest remaining errors to 1.1 and 1.8 kcal/mol.

4.3.3 Density of states above the GMEC

The exact CFN method can enumerate exhaustively sequence/conformation states above the GMEC, up to a given energy threshold, if the threshold is not too large. MC and REMC explore states randomly, within a typical energy range that depends on temperature. To characterize the diversity of the sequence ensembles, we focus on the CFN and REMC methods, and we consider both the sequence entropy and the total number of states.

The mean, exponentiated sequence entropies $\langle e^{S_i} \rangle$ are reported in table 4.7 for each test protein. The sequence entropies for the corresponding Pfam alignments (both seed and full) are also shown. The values are averaged over the designed positions in the protein chain, and can be interpreted as a mean number of amino acid classes sampled at each position. There are six classes (see Methods), one of which (Gly) is not available to the designed positions but is present in Pfam. The entropies are much smaller in the designed sets than in the Pfam sets.

Retaining the top 10,000 designed sequences, CPD samples 1.3 to 1.7 amino acid classes at each position on average, compared to 3–4 in the Pfam alignments, or 2–3 Pfam classes if we exclude the Gly class. Thus the CPD sets are much less diverse than Pfam, as observed earlier for these and other protein families [82, 83]. However, we showed earlier that if we did CPD for around ten backbone conformations, corresponding to ten representatives of a particular domain class (SH3, SH2, PDZ), then collected the sampled sequences, the overall entropy was similar to Pfam [82, 83].

The sequence entropy $S(E)$ is shown in figure 4.7 for the 1CKA SH3 protein as a function of the energy threshold E . Exact CFN results are compared to REMC. For this small protein, complete enumeration was feasible up to an energy threshold of $E = 2$

Table 4.6 – Tests with 20 and 30 designed positions

Protein	20 positions					30 positions			
	CFN	Heur.	MC	REMC	mutations ^a	CFN	Heur.	MC	REMC
1A81 1	gmec*	0.	0.3275	0.3851	0	1.2074	0.	0.6353	
1A81 2	gmec*	0.1705	2.4355	1.0069	3	2.5520	0.	0.0578	
1A81 3	gmec	0.	0.4640	0.6186	0	43.5263	0.	2.4996	1.2025
1A81 4	gmec	0.3878	0.5748	0.6991	4	5.1300	0.	0.0305	
1A81 5	gmec	0.0068	0.5088	0.1541	4	3.2417	0.	1.9586	0.5791
1ABO 1	gmec	0.1205	1.1159	0.2153	2	44.5504	0.	0.	
1ABO 2	13.8563	0.	0.	0.	8	12.7303	0.	0.	
1ABO 3	1.2190	0.	0.	0.	9	9.3870	0.	0.2630	
1ABO 4	1.9940	0.	0.0076	0.	5	10.7691	0.	0.	
1ABO 5	3.5418	0.	0.9483	0.9483	9	4.3907	0.	0.	
1BM2 1	gmec	0.	0.0619	0.1584	0	22.5876	0.	1.7290	1.6013
1BM2 2	7.5304	0.	0.0725	0.0143	8	22.1386	0.	1.9856	1.5876
1BM2 3	gmec	0.0229	0.4762	0.2897	0	22.5410	0.	1.9990	1.1541
1BM2 4	0.1186	0.	2.5883	0.0789	2	15.2639	0.	2.2127	2.3854
1BM2 5	gmec	0.2396	0.3746	0.3746	3	15.9890	0.	2.8354	1.1937
1CKA 1	gmec*	0.	0.	0.	0	6.2700	0.	0.	
1CKA 2	gmec	0.	0.	0.	0	2.0995	0.	0.	
1CKA 3	gmec	0.	0.	0.	0	47.0217	0.	0.	
1CKA 4	4.3122	0.	0.	0.	4	44.0830	0.	0.	
1CKA 5	4.2849	0.	0.	0.	3	8.8608	0.	0.	
1G9O 1	2.0574	0.	1.2525	1.2525	5	2.0816	0.	1.5942	0.
1G9O 2	3.2106	0.	0.2177	0.1915	1	0.3270	0.	0.3126	
1G9O 3	1.9008	0.	0.4417	0.1019	1	17.7150	0.	1.5667	1.5667
1G9O 4	0.5030	0.	0.3855	0.1455	5	2.9758	0.	1.4284	1.6202
1G9O 5	0.4298	0.	0.1495	0.5114	5	0.	1.6890	7.6985	2.3857
1M61 1	gmec	0.	0.	0.	0	14.4935	0.0097	0.	0.
1M61 2	gmec	0.	0.	0.	0	5.0899	0.	1.8749	0.008
1M61 3	gmec	0.	0.	0.	0	3.5795	0.	0.0154	
1M61 4	gmec	0.	0.	0.	0	16.1511	0.	0.	
1M61 5	gmec	0.	0.2521	0.1345	0	23.0927	0.	0.	
1O4C 1	0.	0.3465	0.0690	0.0587	6	14.9064	0.	0.3435	
1O4C 2	6.4214	0.	0.1963	0.3175	4	58.1558	0.	0.0795	
1O4C 3	gmec	0.	0.3461	0.0997	0	9.9221	0.	0.1789	
1O4C 4	gmec	0.	0.3640	0.1382	0	5.7790	0.	0.0423	
1O4C 5	0.	0.	0.1131	0.2206	0	9.9221	0.	0.1789	
1R6J 1	gmec	0.	0.2604	0.2002	0	gmec*	0.	0.0246	
1R6J 2	gmec	0.	0.0071	0.0183	0	14.9800	0.	0.0957	
1R6J 3	gmec	0.	0.0537	0.0732	0	0.	0.	0.0440	
1R6J 4	gmec	0.	0.0639	0.0601	0	0.	0.	0.0957	
1R6J 5	gmec	0.	0.0735	0.0244	0	0.	0.7036	1.8823	0.0781
2BYG 1	gmec	0.	3.1878	0.0257	0	17.9752	0.	0.1592	
2BYG 2	gmec	0.	0.0524	0.0831	0	0.3832	0.	0.1502	
2BYG 3	gmec*	0.	1.3564	0.0826	0	0.1442	0.	0.1593	
2BYG 4	gmec	0.	0.1968	0.6022	0	0.	0.0958	0.0050	
2BYG 5	1.8604	0.	0.0933	0.0386	2	0.5003	0.	0.6876	

Format as in table 4.5. gmec* indicates the more aggressive protocol. ^aBetween CFN/Heur.

Table 4.7 – Designed and Pfam sequence entropies

Protein	Top 10,000 structures	Top 10,000 sequences	Pfam seed	Pfam full
1ABO	1.36	1.58	2.79	3.01
1CKA	1.20	1.41	2.84	3.03
1R6J	1.33	1.48	3.11	3.66
1G9O	1.21	1.53	3.29	3.81
2BYG	1.57	1.63	3.31	3.67
1BM2	1.08	1.26	2.90	3.50
1O4C	1.36	1.68	2.94	3.47
1M61	1.31	1.41	2.91	3.51
1A81	1.13	1.29	2.91	3.51

The entropies are exponentiated, then averaged over all positions. The designed entropies correspond to REMC runs where all positions are designed (except Gly/Pro).

kcal/mol above the GMEC. REMC samples energies up to about 14 kcal/mol above the GMEC. The REMC sampling is essentially complete up to about 0.75 kcal/mol above the GMEC, at which point the REMC curve (gray) begins to depart from the exact, CFN curve (black).

However, the REMC diversity at each position agrees very well with the CFN result up to about 1.5 kcal/mol above the GMEC. At $E = 2$ kcal/mol above the GMEC, the REMC entropy is still 93% of the exact value. Thus, REMC samples the full sequence diversity at each position in this range, even though it does not sample exhaustively all the combinations of mutations (let alone rotamers) at all positions. Thus, for any pair of positions, REMC may not sample all combinations of allowed amino acid classes (up to 25 combinations, since the allowed sidechain types are grouped into five classes ; see Methods) but it effectively samples, at either position, the same types as the exact method.

As we consider higher energy threshold values, $E \geq 3$ kcal/mol, the number of states sampled by REMC increases exponentially and the entropy increases in a quasilinear way. Different replicas sample different energy ranges, as expected ; for example, the $kT=0.592$ and $kT=0.888$ kcal/mol replicas sample the 4–10 and 11–14 kcal/mol ranges, respectively.

4.4 Concluding Discussion

Stochastic search methods are very common in CPD [71, 41, 30, 108]. They often take the form of MC-based simulated annealing (SA) [71, 109] ; various forms of biased or quenched Monte Carlo are also popular [110–113]. For example, MC moves can be chosen to preferentially affect residues that have a high local energy [112] and/or selected degrees of freedom can be subjected to energy minimization (quenched) during or after an MC move. The steepest-descent heuristic used here [41] has similarities to quenched MC. These particular quenched or biased stochastic methods do not sample a Boltzmann

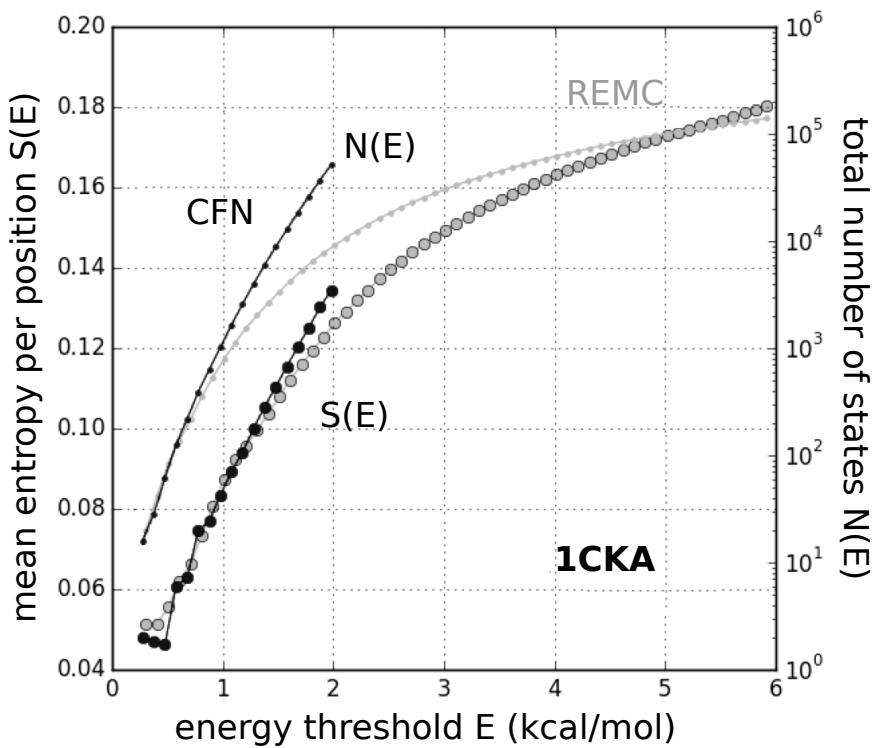


Figure 4.7 – Sequence entropy $S(E)$ and number of states $N(E)$ within a given energy range E above the GMEC for the 1CKA SH3 domain. Ten positions were allowed to vary. The entropy (large dots) is a single position sequence entropy, Eq. (4.13), averaged over all the variable positions. CFN results (black) are based on a complete enumeration of all states within the energy range (at most E kcal/mol above the GMEC). REMC results (gray) are based on the states sampled by all eight walkers during a single trajectory. The number of states (small dots) corresponds to all the different combinations of sequences and rotamers.

distribution of states, but aim to be as effective as possible at identifying low energy sequences, including the GMEC. They can be extended to very large search spaces that include backbone flexibility [114–116], and they do not directly depend on additive energy functions (although additivity can dramatically increase efficiency). In contrast, while exact methods like CFN have been extended beyond discrete rotamer sets [14, 117], and can handle slight departures from a purely additive energy function (ones including 3- or 4-body energy terms) [118, 119], they cannot readily handle arbitrary energy functions or a flexible backbone. Other stochastic methods preserve the Boltzmann character of the statistical ensemble, including plain Monte Carlo, Replica Exchange MC, and configuration bias MC methods [91, 108]. At the same time, REMC benefits from high temperature sampling, like simulated annealing, with energy ranges of 10 kcal/mol or more above the GMEC readily accessible in this work. Here, we tested three stochastic search methods, and compared their ability to identify low energy sequences in problems of increasing size/complexity. While the capability to identify the low energy sequences is important, we should keep in mind that their significance is only as good as the energy function, as illustrated above for 1A81. Direct comparison to the exact GMEC was possible routinely for problems involving up to 10 designed positions, and for 28 of 45 tests with 20 designed

positions. The 10-position designs involved total rotamer numbers of 2500–3000 ; for the 20- and 30-position designs, there are about 2000 and 4000 more rotamers, respectively. The larger tests are relevant for whole protein design projects, as well as projects that redesign one or more large protein surfaces (eg, protein crystal design). For these tests, the GMEC was usually not available, despite using a very recent Toulbar2 protocol [105] . Therefore, the stochastic methods could only be evaluated indirectly. The indirect quality indicators were (a) consistency between the methods and (b) general sequence quality compared to experiment. Agreement between the heuristic and REMC solutions was very good in general, and agreement with experimental Pfam sequences was excellent for core residues, as observed previously with the same energy function (but a heuristic exploration method) [82, 83]. Results with a more advanced protein force field and Generalized Born solvent are expected to be even better [29, 120]. Designed surface residues were less similar to experiment, which can be partly explained because the experimental sequences are subject to additional constraints and selective pressures (such as domain-domain interactions), not included in the design.

Overall, the heuristic and REMC methods gave very good agreement with each other and with the GMEC when available, except for some of the whole-protein design tests. CFN, in its Toulbar2 implementation was very effective for up to 10 designed positions. Exploration speed was similar for all methods for 10-position design, and similar for the stochastic methods applied to the larger problems. With 8-walker REMC and 0.75 billion steps per walker, the three largest departures from the overall best energy, among 67 large, difficult tests, were 4, 3, and 2.4 kcal/mol. Sequence diversity was also recapitulated accurately, compared to exact enumeration, in the very limited range where the latter was possible. In addition, REMC sampled suboptimal solutions as much as 10 kcal/mol above the GMEC, providing thermal averages and an approximate solution to the inverse folding problem. We have recently extended the REMC method to allow backbone moves, with the help of a hybrid move scheme to be described elsewhere. Overall, both the heuristic and REMC method appear to be effective search and sampling methods for most problems and problem sizes.

Annexe du chapitre 4 : Sélection des positions

Cette annexe détaille la méthode de sélection des positions actives employée dans le chapitre 4. Pour les tests avec une seule position active, comme les temps de calculs le permettent toutes les positions sont testées. Il y a huit cent quatre tests par algorithme. Pour les autres groupes de tests (cinq, dix, vingt et trente positions actives), cinq sélections sont effectuées pour chacune des neuf protéines, c'est-à-dire quarante-cinq tests par algorithme.

Pour définir complètement les tests, il faut décrire le choix des positions actives. Il y a peu d'intérêt à tester des situations dans lesquelles les positions actives n'interagissent pas ou faiblement entre elles. En effet, s'il existe une position active i dont chaque résidu est sans interaction avec tous les résidus possibles aux autres positions actives, déterminer le meilleur état pour i est proche du test dans lequel i est la seule position active de la sélection. Cela nous ramène vers les tests à une seule position active. Ainsi le choix des positions actives ne se fait pas par tirage aléatoire, car le risque d'obtenir des positions avec peu d'interactions est trop grand. Il se fait sous contraintes d'interaction. Pour cela, nous utilisons la notion de voisinage de proteus, introduite en 2.5.1. On dit qu'un ensemble de N positions \mathcal{V} est un « voisinage fort » pour le seuil s_{vois} si toutes les paires de positions (i et j) de \mathcal{V} sont voisines pour s_{vois} .

Pour sélectionner les positions actives, nous nous basons sur la liste des voisins de chaque position à un seuil donné. Le programme proteus peut fournir ces listes en mode verbeux, sans effectuer d'optimisation. Pour cela, nous utilisons le mode MONTECARLO avec une trajectoire de zéro pas. La recherche de voisinages forts se fait alors de la façon suivante :

1. s_{vois} est initialisé à 10.
2. construction des listes de voisins au seuil s_{vois}
3. recherche de cinq voisinages forts, par extension progressive d'une paire de positions voisines
4. si succès pour les neuf protéines arrêt, sinon diminution de s_{vois} de 1 kcal/mol
5. si $s_{vois} \geq 1$, retour à l'étape 2, sinon arrêt.

Nous obtenons les 45 voisinages forts, qui constitue notre sélection pour le groupe à 5 et 10 positions actives en utilisant un seuil égal à 10, ils sont dans les tableaux 4.8 et 4.9. Pour les sélections 20-actives et 30-actives, il a fallu descendre le seuil à 1 pour obtenir les 45 voisins forts proches de l'objectif, il manque une ou deux positions dans quelques cas, voir les tables 4.10 et 4.11.

Table 4.8 – La sélection des positions pour tests 5-actives

Nom	positions actives
1A81 1	10 13 16 84 86
1A81 2	20 21 24 27 116
1A81 3	35 38 56 105 107
1A81 4	44 47 52 65 67
1A81 5	82 84 86 87 90
1ABO 1	64 66 90 93 100
1ABO 2	72 74 80 104 111
1ABO 3	79 82 102 111 115
1ABO 4	83 86 104 105 106
1ABO 5	93 100 102 113 116
1BM2 1	101 106 140 141 146
1BM2 2	120 128 131 132 135
1BM2 3	58 61 127 128 129
1BM2 4	74 75 98 100 105
1BM2 5	85 87 95 110 128
1CKA 1	136 138 158 175 190
1CKA 2	149 166 169 171 181
1CKA 3	151 153 157 159 172
1CKA 4	164 170 172 184 187
1CKA 5	172 174 182 186 187
1G9O 1	10 13 54 57 92
1G9O 2	15 39 42 54 57
1G9O 3	24 26 28 39 42
1G9O 4	48 53 57 59 88
1G9O 5	75 78 79 86 88
1M61 1	12 20 23 24 27
1M61 2	17 20 24 37 49
1M61 3	27 33 51 100 102
1M61 4	5 8 10 11 36
1M61 5	59 71 84 87 94
1O4C 1	20 21 32 34 46
1O4C 2	2 71 79 81 82
1O4C 3	33 45 63 71 73
1O4C 4	43 45 63 71 85
1O4C 5	8 33 82 83 86
1R6J 1	194 237 239 270 272
1R6J 2	199 201 211 218 232
1R6J 3	213 218 227 232 238
1R6J 4	221 227 232 267 269
1R6J 5	241 254 258 267 269
2BYG 1	189 191 221 244 246
2BYG 2	205 224 239 245 248
2BYG 3	232 233 265 272 274
2BYG 4	238 240 243 276 278
2BYG 5	253 261 264 265 274

Table 4.9 – La sélection des positions pour tests 10-actives

Nom	positions actives
1A81 1	13 15 39 41 53 86 89 90 93 103
1A81 2	39 41 53 55 64 66 76 89 92 103
1A81 3	51 53 64 66 68 74 76 82 88 92
1A81 4	76 82 87 88 90 91 92 95 97 99
1A81 5	9 10 11 16 41 51 53 66 88 89
1ABO 1	64 72 74 79 89 91 101 103 108 111
1ABO 2	66 68 80 82 88 90 100 102 104 111
1ABO 3	69 70 72 74 80 81 106 113 114 115
1ABO 4	71 78 83 84 94 99 101 104 105 106
1ABO 5	72 79 82 94 99 102 104 106 111 115
1BM2 1	119 120 121 122 123 125 131 134 135 140
1BM2 2	125 126 127 129 130 133 134 136 137 147
1BM2 3	83 99 101 106 108 135 140 141 146 148
1BM2 4	85 95 97 110 118 120 125 128 131 132
1BM2 5	99 101 106 139 140 141 142 143 144 146
1CKA 1	134 135 160 161 162 173 174 175 176 179
1CKA 2	137 139 143 151 153 157 159 172 182 186
1CKA 3	138 140 147 149 150 155 166 169 181 188
1CKA 4	140 141 153 154 155 157 174 175 184 186
1CKA 5	151 153 157 166 168 173 174 176 178 179
1G9O 1	10 11 13 14 15 16 53 54 57 92
1G9O 2	15 17 24 26 39 42 48 51 53 88
1G9O 3	26 28 39 42 48 53 57 59 88 90
1G9O 4	34 35 58 60 68 70 74 75 89 91
1G9O 5	71 73 74 77 80 81 82 83 84 85
1M61 1	10 12 20 23 24 27 35 49 102 104
1M61 2	17 20 21 24 37 39 40 47 49 58
1M61 3	34 36 46 48 59 61 71 83 84 87
1M61 4	5 6 11 36 46 48 61 69 83 84
1M61 5	59 61 70 71 75 77 83 86 87 92
1O4C 1	31 33 45 47 61 63 73 86 89 100
1O4C 2	50 51 52 53 63 72 73 77 85 89
1O4C 3	61 62 63 71 72 73 79 85 88 89
1O4C 4	73 74 75 76 77 89 92 94 96 101
1O4C 5	90 91 93 96 98 99 101 102 103 104
1R6J 1	193 194 195 197 199 218 232 236 267 269
1R6J 2	199 209 211 213 218 227 232 238 265 267
1R6J 3	201 204 205 209 211 218 241 258 265 267
1R6J 4	209 211 213 218 227 238 241 258 265 267
1R6J 5	238 240 241 242 246 257 258 261 265 267
2BYG 1	194 196 203 205 224 233 239 245 274 276
2BYG 2	203 205 207 224 227 233 239 243 245 276
2BYG 3	206 207 222 245 248 253 256 261 264 265
2BYG 4	221 222 245 248 251 253 256 261 264 265
2BYG 5	247 248 249 250 251 252 259 262 263 275

Table 4.10 – La sélection des positions pour tests 20-actives

Nom	positions actives
1A81 1	9 11 12 13 15 16 17 19 20 25 28 39 40 41 42 43 44 45 114 117
1A81 2	9 11 12 13 15 16 17 19 20 25 28 39 40 41 42 43 44 45 47 117
1A81 3	9 11 12 13 15 16 17 19 41 43 48 51 68 74 84 86 109 114 117
1A81 4	12 13 15 16 17 19 20 25 28 39 40 41 42 43 44 45 47 86 114 117
1A81 5	13 15 16 19 41 43 48 51 60 64 68 70 71 74 84 86 87 88 109 114 117
1ABO 1	64 66 67 68 82 86 87 88 89 90 91 101 102 103 108 111 113 116
1ABO 2	64 65 66 67 84 87 88 89 90 91 93 100 101 102 103 108 111 113 116
1ABO 3	65 66 67 87 88 89 90 91 93 94 95 100 101 102 103 106 108 111 113 116
1ABO 4	64 65 66 67 69 87 88 89 90 91 93 100 101 102 103 106 108 111 113 116
1ABO 5	66 67 68 82 86 87 88 89 90 91 93 100 101 102 103 106 108 111 113 116
1BM2 1	55 56 60 61 62 83 84 85 86 87 95 97 99 110 118 125 127 133 150 152
1BM2 2	55 56 60 61 62 83 84 85 86 87 95 97 99 110 118 125 127 128 129 152
1BM2 3	55 56 58 60 61 62 64 67 69 73 83 84 85 86 87 129 132 133 150 152
1BM2 4	55 56 60 61 62 69 83 84 85 86 87 95 97 99 110 129 132 133 150 152
1BM2 5	58 60 60 61 61 62 64 67 69 73 75 83 84 85 86 129 132 133 150 152
1CKA 1	134 135 136 137 138 139 150 151 160 161 162 163 164 170 171 172 173 179 189 190
1CKA 2	134 135 136 137 139 150 151 153 160 161 162 163 164 170 171 172 173 179 189 190
1CKA 3	134 136 137 139 150 151 157 158 160 161 162 163 164 170 171 172 173 179 189 190
1CKA 4	136 137 139 150 151 153 158 159 160 161 162 163 164 170 171 172 173 179 189 190
1CKA 5	137 139 150 151 153 158 160 161 162 163 164 170 171 172 173 174 175 179 189 190
1G9O 1	9 10 11 13 14 15 31 34 38 54 57 58 60 68 90 91 92 94 95 96
1G9O 2	9 11 13 14 15 16 31 34 38 54 57 58 60 68 90 91 92 94 95 96
1G9O 3	9 11 13 14 15 31 34 38 54 55 57 58 60 68 90 91 92 94 95 96
1G9O 4	9 11 13 15 16 17 54 57 58 59 60 61 68 89 90 91 92 94 95 96
1G9O 5	10 11 13 15 16 17 54 57 58 60 61 68 89 90 91 92 94 95 96
1M61 1	34 35 36 37 38 39 42 46 47 48 49 50 61 63 69 71 77 78 81 82
1M61 2	35 36 37 38 39 42 46 47 48 49 50 61 63 69 71 77 78 81 82 83
1M61 3	38 39 42 46 47 48 49 50 61 63 69 71 77 78 81 82 83 84 85 87
1M61 4	42 46 47 48 49 50 61 63 69 71 77 78 81 82 83 84 85 87 88 98
1M61 5	5 7 50 61 63 69 71 77 78 81 82 83 84 85 87 88 98 103 104 109
1O4C 1	32 33 34 35 43 45 63 65 71 73 79 80 81 82 83 84 85 86 87 89
1O4C 2	3 4 5 7 8 9 11 17 31 32 33 34 35 43 45 63 65 71 73 79
1O4C 3	1 3 4 5 6 7 8 9 11 17 31 32 33 35 43 45 65 81 82 83
1O4C 4	1 2 3 4 5 6 7 8 9 11 12 13 14 17 19 35 65 81 82 83
1O4C 5	1 3 4 5 6 7 8 9 11 12 17 31 32 33 34 35 65 81 82 83
1R6J 1	193 194 195 197 214 215 217 218 233 235 236 237 239 240 241 242 247 269 270 273
1R6J 2	193 194 197 198 199 217 233 235 236 237 238 239 240 241 242 247 268 270 272 273
1R6J 3	193 195 197 217 233 235 236 239 240 241 242 244 245 247 268 269 270 272 273
1R6J 4	193 195 197 217 233 235 236 237 239 241 242 244 245 247 268 269 270 272 273
1R6J 5	193 194 197 198 199 233 236 237 239 240 241 247 268 269 270 272 273
2BYG 1	186 187 188 189 190 191 192 215 216 219 244 246 270 271 273 274 278 280 281 282
2BYG 2	186 187 188 189 190 215 216 219 221 223 240 243 270 271 273 274 278 280 281 282
2BYG 3	186 187 188 189 190 215 216 219 221 223 240 243 244 270 271 273 278 280 281 282
2BYG 4	186 187 188 189 190 215 216 219 221 223 240 243 244 270 274 276 278 280 281 282
2BYG 5	187 189 190 191 192 215 216 219 221 223 240 243 244 270 274 276 278 280 281 282

Table 4.11 – La sélection des positions pour tests 30-actives

Nom	positions actives
1A81 1	9 11 12 13 15 16 17 19 20 25 26 27 28 29 36 38 39 40 41 42 43 48 51 68 74 84 86 109 114 117
1A81 2	9 10 11 12 13 15 16 17 19 20 25 28 39 41 43 48 51 68 74 83 84 86 87 88 90 91 93 109 114 117
1A81 3	9 11 12 13 15 16 17 19 20 25 27 28 36 38 39 40 41 42 43 48 51 68 74 84 86 109 114 117
1A81 4	9 10 11 12 13 15 16 17 19 20 25 28 36 39 40 41 42 43 44 45 48 51 68 74 84 86 109 114 117
1A81 5	9 10 11 12 13 15 16 17 19 20 25 28 39 40 41 42 43 44 45 47 48 51 52 68 74 84 86 109 114 117
1ABO 1	64 65 66 67 68 70 71 72 75 78 79 80 81 82 83 86 87 88 89 90 91 93 100 101 102 103 108 111 113 116
1ABO 2	64 65 66 67 68 72 75 78 80 81 82 83 84 86 87 88 89 90 91 93 94 100 101 102 103 104 108 111 113 116
1ABO 3	64 66 67 68 70 71 72 78 82 86 87 88 89 90 91 93 94 95 96 99 100 101 102 103 104 105 108 111 113 116
1ABO 4	64 65 66 67 70 71 72 68 82 86 87 88 89 90 91 93 94 95 96 99 100 101 102 103 104 105 108 111 113 116
1ABO 5	65 66 67 70 71 72 75 78 80 82 86 87 88 89 90 91 93 94 95 96 99 100 101 102 103 108 111 113 116
1BM2 1	55 56 58 60 61 62 83 84 85 86 87 95 97 99 110 118 119 120 121 122 125 127 128 129 130 131 132 133 150 152
1BM2 2	56 58 60 61 62 83 84 85 86 87 95 97 99 110 118 119 120 121 122 123 125 127 128 129 130 131 132 133 150 152
1BM2 3	58 60 61 62 83 84 85 86 87 95 97 99 108 109 110 118 120 121 122 123 125 127 128 129 132 133 134 135 150 152
1BM2 4	55 56 58 60 61 62 83 84 85 86 87 95 97 99 108 109 110 118 120 121 125 127 128 129 130 131 132 133 150 152
1BM2 5	56 58 60 61 62 67 83 84 85 86 87 95 97 99 110 111 112 113 115 118 125 127 128 129 130 131 132 133 150 152
1CKA 1	134 135 136 137 139 140 141 142 143 144 146 147 148 149 150 151 158 159 160 161 162 163 164 170 171 172 173 179 189 190
1CKA 2	134 135 136 137 139 143 144 146 147 148 149 150 151 158 159 160 161 162 163 164 170 171 172 173 179 186 187 188 189 190
1CKA 3	135 136 137 139 144 146 147 148 149 150 151 157 158 159 160 161 162 163 164 170 171 172 173 179 186 187 188 189 190
1CKA 4	136 137 139 140 141 142 143 144 146 147 148 151 158 159 160 161 162 163 164 170 171 172 173 179 184 186 187 188 189 190
1CKA 5	134 136 137 139 140 141 142 143 144 146 147 148 151 158 159 160 161 162 163 164 170 171 172 173 179 182 187 188 189 190
1G90 1	9 10 11 13 15 24 31 34 38 40 41 42 43 46 48 49 50 51 54 57 58 60 68 89 90 91 92 94 95 96
1G90 2	9 11 13 15 31 32 34 38 40 41 42 43 46 48 49 50 51 54 57 58 60 68 89 90 91 92 94 95 96
1G90 3	9 10 11 13 15 31 32 34 38 40 41 42 43 46 48 49 50 51 54 57 58 60 68 89 90 91 92 94 95 96
1G90 4	10 11 13 14 15 31 32 34 38 40 41 42 43 46 48 49 50 51 54 57 58 60 61 68 87 89 90 91 92 94 95 96
1G90 5	10 11 13 14 15 31 32 40 41 42 43 46 48 49 50 51 54 57 58 60 61 62 68 87 89 90 91 92 94 95 96
1M61 1	12 14 15 20 34 35 36 37 38 39 42 46 47 48 49 50 61 63 69 71 77 78 81 82 83 84 85 87 88 98
1M61 2	6 7 8 10 11 12 14 15 20 21 34 35 36 37 38 39 42 46 47 48 49 50 61 63 69 71 77 78 81 82
1M61 3	5 7 35 36 37 38 39 42 46 47 48 49 50 61 63 69 71 77 78 81 82 83 84 85 87 88 98 103 104 109
1M61 4	7 8 10 11 12 14 15 20 34 35 36 37 38 39 42 46 47 48 49 50 61 63 69 71 77 78 81 82 83 84
1M61 5	8 10 11 12 14 15 20 34 35 36 37 38 39 42 46 47 48 49 50 61 63 69 71 77 78 81 82 83 84 85
1O4C 1	1 2 3 4 5 6 7 8 9 11 17 31 32 33 34 35 43 45 63 65 71 81 82 83 90 91 92 93 94 96
1O4C 2	1 3 4 5 7 8 9 11 17 31 32 33 34 35 43 45 63 65 71 73 79 80 81 82 83 90 91 92 93 96
1O4C 3	1 3 4 5 6 7 8 9 11 17 31 32 33 34 35 43 45 63 65 71 73 79 80 81 82 83 90 91 92 93 96
1O4C 4	1 3 4 5 7 8 9 11 17 31 32 33 34 35 43 45 63 65 71 73 79 80 81 82 83 84 91 92 93 96
1O4C 5	1 3 4 5 6 7 8 9 11 17 31 32 33 34 35 43 45 63 65 71 73 79 80 81 82 83 84 92 93 96
1R6J 1	193 194 195 197 198 199 217 218 219 220 221 222 223 224 225 226 227 228 229 233 235 236 237 239 247 268 269 270 272 273
1R6J 2	193 194 195 197 198 199 217 220 221 222 223 224 225 226 227 228 229 233 235 236 237 239 240 247 268 269 270 272 273
1R6J 3	193 194 195 197 198 199 208 217 220 221 222 223 224 225 226 227 228 229 230 233 235 236 237 239 247 268 269 270 272 273
1R6J 4	193 194 195 197 198 199 217 220 221 222 223 224 225 226 227 228 229 233 235 236 237 239 240 247 249 268 269 270 272 273
1R6J 5	194 195 197 198 199 217 220 221 222 223 224 225 226 227 228 229 233 235 236 237 239 240 247 249 250 268 269 270 272 273
2BYG 1	186 187 188 189 190 191 192 215 216 219 221 223 240 243 244 246 250 251 252 253 254 255 256 257 259 260 278 280 281 282
2BYG 2	186 187 188 189 190 191 192 198 215 216 219 221 223 240 243 244 251 252 253 254 255 256 257 259 260 278 280 281 282
2BYG 3	186 187 188 189 190 191 192 215 216 219 221 223 240 243 244 251 252 253 254 255 256 257 259 260 278 246 280 281 282
2BYG 4	186 187 188 189 190 191 192 215 216 219 221 223 240 243 244 245 246 251 252 253 254 255 256 257 259 260 278 281 282
2BYG 5	186 187 188 189 190 191 192 215 216 219 221 223 240 243 244 246 251 252 253 254 255 256 257 259 260 278 280 281 282

Chapitre 5

Dessin computationnel de domaines PDZ

5.1 Introduction

Nous cherchons maintenant à évaluer la performance de notre modèle CPD sur un ensemble de protéines PDZ. Les domaines PDZ (« Postsynaptic density-95 / Discs large / Zonula occludens-1 ») sont de petits domaines globulaires qui établissent des réseaux d’interactions entre protéines dans la cellule [121–125]. Ils forment des interactions spécifiques avec des protéines cibles, généralement en reconnaissant quelques acides aminés à l’extrémité C-terminale. En raison de leur importance biologique, les domaines PDZ et leur interaction avec les protéines cibles ont été largement étudiés et utilisés en conception *in silico*. Des ligands ont été conçus pour moduler l’activité de domaines PDZ impliqués dans diverses pathologies [126, 127]. Des domaines PDZ et leurs ligands redessinés ont été utilisés pour élucider les principes du repliement des protéines et de l’évolution [128–130]. Ainsi, ces domaines avec leurs ligands peptidiques fournissent d’excellents « benchmarks » pour tester les méthodes informatiques elles-mêmes [131, 132].

À partir d’une sélection de domaines PDZ, nous optimisons les énergies de référence, paramètres essentiels dans notre modèle, grâce à la méthode du maximum de vraisemblance. Puis, la performance du modèle est testée en générant des séquences par Proteus pour chaque protéine de la sélection. Pour cela, nous nous basons sur les résultats précédents, en particulier ceux de la section 4.3.2, pour définir les paramètres du Monte Carlo. Nous confrontons nos résultats à ceux de la fonction d’énergie de Rosetta, fonction qui a connu le plus de succès et qui est la plus souvent citée [133]. Elle comprend un terme de répulsion de Lennard-Jones, un terme Coulomb, un terme de liaison hydrogène, un terme de solvatation Lazaridis-Karplus et des énergies de référence d’état dépliées, mais est plus empirique que la nôtre. Il y a un grand nombre de paramètres spécifiquement optimisés pour le CPD, qui offrent des performances optimales, mais une interprétation physique moins transparente que Proteus.

La production de nos séquences calculées est effectuée par des simulations Monte Carlo où toutes les positions de la chaîne polypeptidique sont autorisées à muter librement, excepté celles occupées par une glycine ou une proline qui conservent leur type sauvage.

Ces exceptions découlent de la contrainte du backbone fixe, puisque ces deux types d'acides aminés influent sur la structure du squelette. Nous obtenons des milliers de séquences pour chaque domaine PDZ. Nos tests comprennent une validation directe et une validation croisée dans laquelle les énergies de référence optimisées sur un sous-ensemble de domaines sont utilisées sur un autre sous-ensemble.

Nous réalisons également une série de simulations Monte Carlo de deux domaines PDZ où le potentiel chimique hydrophobe des types d'acides aminés est progressivement augmenté, polarisant artificiellement la composition de la protéine. Comme le biais hydrophobe augmente, les acides aminés hydrophobes envahissent progressivement la protéine de l'intérieur. La propension de chaque position du noyau à devenir hydrophobe à un niveau de biais plus ou moins élevé peut être considérée comme un indice d'hydrophobicité. Il dépend de la structure et nous renseigne sur la capacité de la protéine à supporter des mutations.

5.2 Le modèle d'état déplié

5.2.1 Les énergies de référence

Les simulations MC sont gouvernées par l'énergie de repliement de la protéine, c'est-à-dire la différence entre son énergie à l'état replié et déplié. Un mouvement élémentaire possible est une « mutation ». Il s'agit d'une modification du type de chaîne latérale à une position choisie i dans la protéine pliée, en assignant un rotamère particulier à la nouvelle chaîne latérale. On effectue en même temps la modification inverse dans l'état déplié, voir les détails en 2.1. Pour une séquence S particulière, nous avons introduit à la section 1.1.1 l'énergie de l'état déplié comme :

$$E^u = \sum_{i \in S} E_{t_i}^r \quad (5.1)$$

Ici, i est la position dans la séquence et t_i le type en i , la somme se faisant sur tous les résidus de S . Les grandeurs E_t^r sont les « énergies de référence ». Ce sont les seuils paramètres empiriques dans notre modèle de simulation.

5.2.2 La vraisemblance des énergies de référence

Notre objectif est de déterminer les énergies de référence empiriquement afin que les simulations produisent des fréquences d'acides aminés qui correspondent à un ensemble de valeurs cibles expérimentales. Pour cela, nous choisissons les E_t^r qui maximisent la probabilité de Boltzmann de l'ensemble de séquences expérimentales. Nous retenons donc les E_t^r les plus vraisemblables étant donnée l'observation des séquences expérimentales. Soit S une séquence particulière. Sa probabilité de Boltzmann est :

$$p(S) = \frac{1}{Z} \exp(-\beta \Delta G_S) \quad (5.2)$$

avec

$$\Delta G_S = G_S^f - E_S^u \quad (5.3)$$

ΔG_S est l'énergie libre de repliement de S , G_S^f est l'énergie libre de l'état replié, $\beta = \frac{1}{kT}$ est la température inverse et Z une constante de normalisation (la fonction de partition). En injectant 5.3 dans 5.2, nous avons alors :

$$kT \ln p(S) = \sum_{i \in S} E^r(t_i) - G_S^f - kT \ln Z = \sum_{t \in aa} n_S(t) E_t^r - G_S^f - kT \ln Z \quad (5.4)$$

La somme à droite se fait sur l'ensemble des types d'acides aminés et $n_S(t)$ est le nombre d'acides aminés de type t dans S .

Nous considérons maintenant un ensemble \mathcal{S} de N séquences cibles ; on appelle \mathcal{L} la probabilité d'observer l'ensemble \mathcal{S} en entier. \mathcal{L} est fonction des paramètres du modèle E_t^r . Nous appelons \mathcal{L} la vraisemblance des E_t^r [134]. Nous avons :

$$kT \ln \mathcal{L} = \sum_S \sum_{i \in aa} n_S(t) E_t^r - \sum_S G_S^f - NkT \ln Z = \sum_{t \in aa} N(t) E_t^r - \sum_S G_S^f - NkT \ln Z \quad (5.5)$$

avec $N(t)$ le nombre d'acides aminés de type t dans l'ensemble \mathcal{S} . Le facteur de normalisation Z est une somme sur l'ensemble des séquences possibles \mathcal{R} :

$$Z = \sum_{\mathcal{R}} \exp(-\beta \Delta G_{\mathcal{R}}) = \sum_{\mathcal{R}} \exp(-\beta \Delta G_{\mathcal{R}}^f) \Pi_{t \in aa} \exp(\beta n_{\mathcal{R}}(t) E_t^r) \quad (5.6)$$

Pour maximiser \mathcal{L} , nous considérons la dérivée de Z selon chacune des E_t :

$$\frac{\partial Z}{\partial E_t^r} = \sum_{\mathcal{R}} \beta n_{\mathcal{R}}(t) \exp(-\beta \Delta G_{\mathcal{R}}^f) \Pi_{s \in aa} \exp(\beta n_{\mathcal{R}}(s) E_s^r) \quad (5.7)$$

Nous avons alors :

$$\frac{kT}{Z} \frac{\partial Z}{\partial E_t^r} = \frac{\sum_{\mathcal{R}} n_{\mathcal{R}}(t) \exp(-\beta \Delta G_{\mathcal{R}})}{\sum_{\mathcal{R}} \exp(-\beta \Delta G_{\mathcal{R}})} = \langle n(t) \rangle. \quad (5.8)$$

La quantité à droite est la moyenne de Boltzmann du nombre $n(t)$ des acides aminés de type t . Comme une simulation Monte Carlo converge vers la distribution de Boltzmann, la population moyenne de t que nous obtenons dans nos simulations converge vers $\langle n(t) \rangle$. En pratique, nous obtenons donc cette quantité comme la population moyenne de t dans une simulation Monte Carlo assez longue.

Pour que $\ln \mathcal{L}$ soit maximal il faut que ses dérivées par rapport à E_t^r soient nulles :

$$\frac{1}{N} \frac{\partial}{\partial E_t^r} \ln \mathcal{L} = \frac{1}{N} \sum_S n_S(t) - \langle n(t) \rangle = \frac{N(t)}{N} - \langle n(t) \rangle \quad (5.9)$$

et donc

$$\mathcal{L} \text{ maximum} \implies \frac{N(t)}{N} = \langle n(t) \rangle, \forall t \in aa$$

Ainsi, pour maximiser \mathcal{L} , nous choisissons les E_t^r telles qu'une longue simulation donne les mêmes fréquences d'acides aminés que l'ensemble cible.

5.2.3 Recherche du maximum de vraisemblance

Nous utilisons trois méthodes pour approcher les valeurs E_t^r .

- La première consiste à avancer dans la direction du gradient de $\ln(\mathcal{L})$ en utilisant la règle itérative suivante [134] :

$$E_t^r(n+1) = E_t^r(n) + \alpha \frac{\partial}{\partial E_t^r} \ln(\mathcal{L}) = E_t^r(n) + \delta E (n_t^{exp} - \langle n(t) \rangle_n) \quad (5.10)$$

avec α une constante, $n_t^{exp} = \frac{N(t)}{N}$ la population moyenne d'acide aminé de type t dans l'ensemble ciblé, $\langle \cdot \rangle_n$ indique une moyenne sur une simulation effectuée en utilisant les énergies de références courantes $E_t^r(n)$, et δE une constante empirique avec la dimension d'une énergie, correspondant à l'amplitude de mise à jour. Cette procédure de mise à jour est répétée jusqu'à convergence. Nous l'appelons **méthode linéaire**.

- La deuxième méthode est une variante de la première dans laquelle le δE n'est pas constant, mais ajusté au cours de la simulation de la façon suivante. On introduit une fonction proxy C comme outil de mesure de l'état de l'optimisation :

$$C = \sum_{t \in aa} (n_t^{exp} - \langle n(t) \rangle_n)^2 \quad (5.11)$$

À chaque itération on applique trois fois la règle 5.10 avec trois valeurs différentes de δE , mais des énergies de références identiques. Une interpolation parabolique est effectuée sur les trois valeurs de la fonction C obtenues, le minimum de la parabole est calculé. Ce minimum est utilisé comme δE pour un quatrième cycle, au terme duquel les énergies sont mises à jour. Nous appelons cette méthode la **méthode parabolique**.

- La troisième méthode, utilisée précédemment [100, 101], applique une règle de mise à jour logarithmique :

$$E_t^r(n+1) = E_t^r(n) + kT \ln \frac{\langle n(t) \rangle_n}{n_t^{exp}} \quad (5.12)$$

avec kT l'énergie thermique, fixée empiriquement à 0,5 kcal/mol. Nous l'appelons la **méthode logarithmique**. Dans les dernières itérations, certaines valeurs ont tendance à converger lentement, avec des oscillations. Par conséquent, une règle modifiée est ajoutée dans laquelle l'énergie au cycle n et l'énergie au cycle $n-1$ sont moyennées avec un poids respectif de 2/3 et 1/3 afin d'atténuer les vibrations.

5.3 Méthodes de calcul

5.3.1 Énergie de l'état replié

La matrice énergétique d'une protéine est calculée avec la fonction d'énergie suivante :

$$E = E_{MM} + E_{GB} + \sum_i \sigma_i A_i \quad (5.13)$$

E_{MM} représente l'énergie interne de la protéine et se compose de six termes détaillés en 1.2.1. Les autres termes de l'équation 5.13 représentent la contribution du solvant. Nous utilisons un modèle de solvant implicite « Generalized Born + Surface Area » ou GBSA (1.3.5 et 1.12). Ici A_i est la surface accessible au solvant de l'atome i , σ_i est un paramètre qui représente la préférence de chaque atome à être exposé ou non au solvant. Les atomes du soluté sont divisés en quatre groupes avec pour chacun une valeur σ_i spécifique en cal/mol/Å². Nous travaillons avec deux jeux différents pour les σ_i . Le premier est un jeu déjà utilisé dans différentes études [70, 135] : non polaire -50, aromatique -120, polaire -80, ionique -90. Le second provient d'une étude plus récente faite dans le notre laboratoire, dans laquelle il a été optimisé sur un ensemble de protéines PDZ [58] : non polaire -30, aromatique -10, polaire -90, ionique -90. Dans les deux cas, on attribue aux atomes d'hydrogène un coefficient de surface de 0.

Les surfaces sont calculées par l'algorithme de Lee et Richards [106], qui est implémenté dans le programme Xplor [47] et expliqué en 2.3.1. Les simulations MC utilisent une constante diélectrique pour la protéine $\epsilon_p = 4$ ou 8. Nous utilisons deux variantes du modèle GB, la méthode NEA pour « Native Environment Approximation » et la méthode FDB ou « Fluctuating Dielectric Boundary » [49], plus rigoureuse. Elles sont présentées respectivement en 1.3.5 et 2.3.3.

Le champ de force utilisé, Amber ff99SB [17], est légèrement modifié pour le CPD, en remplaçant les charges du backbone par un ensemble unifié, obtenu en faisant la moyenne sur l'ensemble des types d'acides aminés et ajustant légèrement pour rendre la partie backbone de chaque acide aminé neutre [84].

5.3.2 Les énergies de référence de l'état déplié

Dans le modèle CPD, l'énergie de l'état déplié dépend de la composition de la séquence par l'ensemble des énergies de référence E_t^r (équation 5.1). Ici, les énergies de référence sont attribuées en fonction des types d'acides aminés t , mais aussi de la position de chaque acide aminé dans la structure repliée à travers son caractère enfoui ou exposé au solvant. Ainsi, pour un type donné, il y a deux valeurs distinctes de E_t^r , une enfouie et une exposée. Cette approche se justifie par les trois éléments suivants :

- Nous supposons que la structure résiduelle est présente dans l'état déplié, de sorte que les acides aminés conservent en partie leur caractère enfoui/exposé.

- Nous supposons que le modèle d'état déplié compense de manière systématique des erreurs dans la fonction d'énergie de l'état plié, de sorte que la structure pliée contribue indirectement aux énergies de référence.
- Cette stratégie rend le modèle moins sensible aux variations de la longueur des boucles de surface et au ratio du nombre de résidus de surface sur celui des enfouis, qui peut varier considérablement selon les homologues.

Par conséquent, ce modèle devrait être plus transférable à l'intérieur d'une famille de protéines. Distinguer les positions enfouies/exposées double le nombre de paramètres E_t^r à ajuster. Inversement, pour réduire le nombre de paramètres, nous groupons les acides aminés en classes homologues, voir table 5.1. Dans chaque classe c, et pour chaque type de position (enfoui ou exposé), les énergies de référence ont la forme :

$$E_t^r = E_c^r + \delta E_t^r \quad (5.14)$$

E_c^r est un paramètre ajustable, tandis que δE_t^r est une constante, calculée comme la différence d'énergie de mécanique moléculaire entre les types d'acides aminés de classe c, pour une conformation dépliée.

Groupe	acides aminés	propriétés
1	Ala, Cys, Thr	petit
2	Ser	
3	Glu, Asp	chargé négativement
4	Gln, Asn	polaire
5	Ile, Leu, Val	apolaire
6	Met	non polaire
7	Hip, Hid, Hie	chargé positivement
8	Arg	
9	Lys	
10	Phe, Trp	aromatique
11	Tyr	
12	Gly, Pro	non mutable

Table 5.1 – Les groupes d'acides aminés utilisés pour l'optimisation des énergies de référence.

Plus précisément, nous effectuons des simulations MC d'un peptide étendu (le peptide *Syndecan1*) et calculons les énergies moyennes pour chaque type d'acide aminé à chaque position peptidique (à l'exclusion des positions terminales). Nous prenons les différences entre les types d'acides aminés et les moyennons sur les positions peptidiques. Pour optimiser les valeurs E_t^r , nous utilisons une des trois méthodes de la section 5.2.3 avec comme fréquences cibles les fréquences expérimentales des classes d'acides aminés ou

des types d'acides aminés. Le début d'optimisation se fait sur les classes, puis lorsque la convergence de la fonction proxy C est correctement établie, nous relâchons cette contrainte pour optimiser sur l'ensemble des types d'acide aminé mutables. Typiquement, vingt cycles d'optimisation sont effectués sur les classes, puis encore vingt cycles sur les types.

5.4 Séquences expérimentales et modèles structuraux

5.4.1 L'ensemble des protéines PDZ

Nous sélectionnons huit protéines de la famille PDZ dont les structures cristallographiques sont connues. Aux trois présentes au chapitre précédent, NHREF, Syntenin et DLG2, sont ajoutées les protéines INAD, GRIP, PSD95, Cask et Tiam1. Leur séquence est présentée à la figure 5.1. Le nombre de positions actives, c'est-à-dire les positions qui vont être mutées, est du même ordre pour chaque protéine (voir le tableau 5.2).

Table 5.2 – La sélection de domaines protéiques PDZ

nom	Code PDB	résidus	nombre de positions actives
NHREF	1G9O	9-99	76
INAD	1IHJ	13-105	82
GRIP	1N7E	668-761	79
Syntenin	1R6J	193-273	72
DLG2	2BYG	186-282	82
PSD95	3K82	305-402	80
Cask	1KWA	487-568	74
Tiam1	4GVD	838-930	84

5.4.2 Alignements Blast croisés

Pour caractériser les homologies dans cet ensemble de huit domaines PDZ, une série de requêtes BLAST est effectuée sur chaque paire de séquences en utilisant le programme `blastp` avec les options comme indiqué en 2.7.6. Il apparaît que Syntenin et Tiam1 sont atypiques dans l'ensemble : il n'y a pas d'homologues avec une E-value inférieure à 10^{-7} et plusieurs E-value supérieures à 10. PSD95 est la protéine la plus consensuelle, ayant d'une part une homologie avec toutes les autres à au plus $6 \cdot 10^{-4}$, et d'autre part ayant quatre homologues à moins de $2 \cdot 10^{-10}$, pour un pourcentage d'identité compris entre 30 et 46. Globalement, il n'y a que peu d'homologies, la plus forte n'étant que de $3 \cdot 10^{-15}$ entre PSD95 et DLG2 pour un pourcentage d'identité de 37. Les détails sont dans le tableau 5.3.

5.4.3 Sélection des homologues

Pour définir les fréquences d'acides aminés cibles nécessaires pour maximiser nos vraisemblances, nous sélectionnons un ensemble de séquences homologues pour chacune

Table 5.3 – E-value et pourcentage d’identité des alignements Blast native versus native pour nos séquences PDZ

Protéine	NHREF	INAD	GRIP	Syntenin	DLG2	PSD95	CASK	TIAM1
NHREF	2 10 ⁻⁶⁶ (100)	5 10 ⁻¹⁰ (40)	2 10 ⁻³ (25)	3 10 ⁻⁷ (25)	2 10 ⁻¹¹ (35)	1 10 ⁻¹² (30)	5 10 ⁻⁵ (25)	9 10 ⁻⁷ (35)
INAD	5 10 ⁻¹⁰ (40)	3 10 ⁻⁶⁸ (100)	2 10 ⁻⁷ (27)	[18]	2 10 ⁻⁸ (27)	9 10 ⁻¹⁴ (46)	4 10 ⁻⁶ (35)	[16]
GRIP	2 10 ⁻³ (25)	2 10 ⁻⁷ (27)	3 10 ⁻⁶⁷ (100)	[21]	3 10 ⁻¹⁴ (36)	2 10 ⁻¹⁰ (37)	9 10 ⁻¹² (30)	5 10 ⁻⁵ (35)
Syntenin	3 10 ⁻⁷ (25)	[18]	[21]	1 10 ⁻⁵⁹ (100)	[17]	1 10 ⁻⁶ (32)	7 10 ⁻³ (32)	[18]
DLG2	2 10 ⁻¹¹ (35)	2 10 ⁻⁸ (27)	3 10 ⁻¹⁴ (37)	[17]	7 10 ⁻⁷¹ (100)	3 10 ⁻¹⁵ (37)	2 10 ⁻⁷ (28)	5 10 ⁻⁵ (41)
PSD95	1 10 ⁻¹² (30)	9 10 ⁻¹⁴ (46)	2 10 ⁻¹⁰ (36)	1 10 ⁻⁶ (32)	3 10 ⁻¹⁵ (37)	4 10 ⁻⁷⁰ (100)	1 10 ⁻⁷ (27)	6 10 ⁻⁴ (33)
Cask	5 10 ⁻⁵ (25)	4 10 ⁻⁶ (35)	9 10 ⁻¹² (30)	7 10 ⁻³ (32)	2 10 ⁻⁷ (28)	1 10 ⁻⁷ (27)	7 10 ⁻⁶¹ (100)	5 10 ⁻⁴ (33)
Tiam1	9 10 ⁻⁷ (35)	[16]	5 10 ⁻⁵ (35)	[18]	5 10 ⁻⁵ (41)	6 10 ⁻⁴ (33)	5 10 ⁻⁴ (33)	1 10 ⁻⁶⁸ (100)

S'il n'y a pas de touche avec une E-value inférieure à 10, [.] donne le pourcentage d'identité du couple dans l'alignement des six séquences sauvages.

des six premières protéines de notre sélection. En effet, nous excluons Cask et Tiam1 pour le calcul des énergies de références. Pour cela, nous effectuons des recherches BLAST avec comme requête la séquence extraite du fichier PDB sur la base de données « Swiss-prot + trEmBL » d'Uniprot avec la matrice BLOSUM62, l'option « Gapped » et sans l'option « filtre ».

Nous obtenons un premier ensemble pour chaque cas en nous limitant aux homologues de bonne qualité au regard de la E-value et du pourcentage d'identité, tout en conservant en même temps une certaine diversité. Cela oblige pour certaines protéines à accepter des E-values plus hautes que 10⁻⁴⁰, notamment INAD et NHREF, respectivement 10⁻³² et 10⁻¹⁰, pour avoir un nombre d'homologues suffisant. Ensuite, les redondances les plus flagrantes sont enlevées manuellement. Finalement, les ensembles se composent de 42 à 126 homologues, avec des pourcentages d'identité supérieurs à 66% excepté pour INAD où il a fallu descendre jusqu'à 38%, voir le tableau 5.4. Les alignements des séquences homologues retenues pour un groupe constitué des six premières protéines sont représentés aux figures 5.11, 5.13, 5.15, 5.17, 5.19 et 5.21.

Table 5.4 – Sélection des homologues.

protéines	nombre	E-value	% identité
NHREF	62	≤ 10 ⁻³²	67-95
INAD	42	≤ 10 ⁻¹⁰	38-95
GRIP	48	≤ 10 ⁻⁴⁵	84-95
Syntenin	85	≤ 10 ⁻⁴³	85-95
DLG2	43	≤ 10 ⁻⁴¹	78-95
PSD95	50	≤ 10 ⁻⁴⁶	81-95
Cask	126	≤ 10 ⁻²⁷	60-85
Tiam1	50	≤ 10 ⁻²²	60-85

5.4.4 Alignements des protéines expérimentales et leurs homologues

Afin d'obtenir une caractérisation structurale de notre sélection de protéines, nous réalisons un alignement des huit séquences natives, présentées en haut de la figure 5.1. Cet alignement nous sert de base pour la définition d'un alignement structural. Nous pouvons alors définir un cœur hydrophobe de nos protéines « PDZ », il est représenté au centre de 5.1. Les 14 positions utilisées pour définir le cœur hydrophobe sont bien conservées dans l'alignement « seed » de Pfam, mais pas totalement. L'Arg, Lys et Gln apparaissent à certaines positions, puisque dans de petites protéines comme des domaines PDZ, la longue partie hydrophobe de ces chaînes latérales peut être enfouie dans le cœur tout en ayant à la pointe polaire exposée au solvant. Quelques résidus Asp et Glu apparaissent aussi, dans les endroits où l'alignement des séquences peut ne pas très bien refléter la superposition 3D les chaînes latérales.

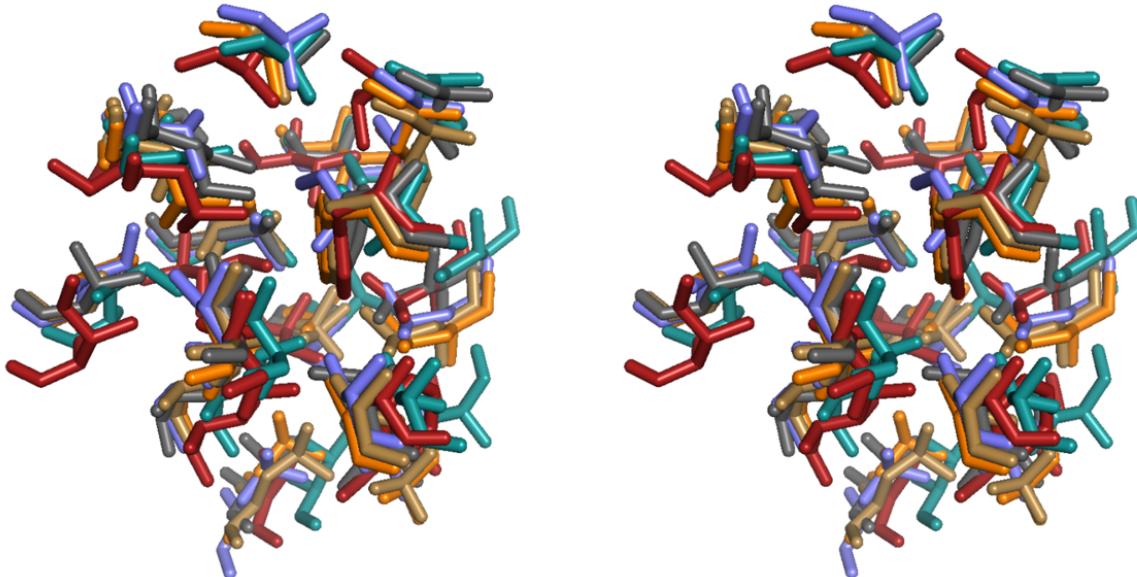
Table 5.5 – Similarité des séquences expérimentales homologues des huit protéines PDZ.

Protéine	NHREF	INAD	GRIP	Syntenin	DLG2	PSD95	Cask	Tiam1
NHREF	326	64	15	15	59	112	49	1
INAD	64	221	56	-9	88	107	25	9
GRIP	15	56	378	24	65	87	90	39
Syntenin	15	-10	24	311	-26	22	42	-18
DLG2	59	88	65	-26	325	110	24	22
PSD95	112	107	87	22	110	325	66	21
Cask	49	25	90	42	23	66	308	37
Tiam1	1	10	39	-18	22	21	37	371

5.4.5 Similarité des homologues

Comme nous avons caractérisé les homologies des séquences natives, nous calculons également la similarité des profils formés à partir des séquences expérimentales homologues. Chaque couple de profils est aligné comme l'alignement du couple de séquences natives correspondant, voir figure 5.1. Apparaissent des situations où les similarités sont négatives, c'est le cas chez les homologues de Cask et Syntenine, protéines déjà repérées comme éloignées en termes de séquences, avec une valeur de -26 entre DLG2 et Syntenine et -18 entre Tiam1 et Syntenin. Les plus hautes valeurs (excepté les valeurs d'auto similarité) sont 112 entre PSD95 et NREF et 110 entre PSD95 et DLG2. La similarité moyenne entre deux protéines différentes de notre sélection est de 50 environ. Les détails sont à la table 5.5.

NHREF RMLPRLCCLEK. GPNGYGFHLHGEKGKL GQYTRLVEPGSPAEKAG. LLAGDRLVEVNGENVEKETHQQVVSRTAALNAVRLLVVDPETDEQL
INAD GELIHMVTLDKTGKSFICIVRGEVKDSPNTKTTGFIKGIVPDSPAHLCGRLKVGDRILSNGKDVRNSTEQAVIDLKIEADFKIELEIQTFF
GRIP GAIYTVELKR. YGGPLGITISGTEEP FDPIISSLTKGGLAERTGAIHGDRILAISSSSLKGKPLSEAIHLLQMAGETVTLKIKKKQTDAQPASS
Syntenin GAMDPRTITMHKDSTGHVGIFKNN GKITSIVKDSSAARNG. LLTEHNICEINGQNVIGLKDSQIAIDLSTSGTVVTITIMPAF
DLG2 FQSMVTVEIQLFK. GPKGLGFSIAGGVGNQH. IPGDNSIYVTKIIDGGAQKDGRQLQVGDRLLMVNNYSLEEVTHEEAVAILKNTSEVVYLKGKPTTI
PSD95 EDIPREPRRIVIHR. GSTGLGFNIVGGEGE GIFTISFILAGGPADLSGELRKGDQILSVNGVDLRNASHEQAIALKNAGQTVTIIAQYKPEEYSRFEA
Cask RSRLVQFQKNTDEPMGITLKMELN HCIVARIMHGGMIHRQGTLLHVGDEIREINGISVANQTVQEQLQKMLREMRSITFKIVP
Tiam1 GAMGKVTHSIHIEKSDTAADTYGFSLSSVEED GIRRLYVNSVKETGLASKKG. LKAGDEILEINNRADALNSSLKDFLSQP . SLGLLVRTYPEL



	Y	F	L	I	A	L	L	V	V	V	I	V	L	V
NHREF	24	26	28	39	48	53	59	62	67	75	79	86	88	90
INAD	F	I	I	I	A	L	I	L	V	V	I	I	L	I
	28	30	32	50	59	65	71	74	79	87	91	98	100	102
GRIP	L	I	I	I	A	I	I	I	L	A	L	V	L	I
	682	684	686	698	707	713	719	722	727	735	739	746	748	750
Syntenin	V	F	F	I	A	L	I	I	V	I	L	V	I	I
	209	211	213	218	227	232	238	241	246	254	258	265	267	269
DLG2	L	F	I	V	A	L	L	V	L	A	L	V	L	V
	203	205	207	224	233	239	245	248	253	261	265	272	274	276
PSD95	L	F	I	I	A	L	I	V	L	A	L	V	I	A
	323	325	327	338	347	353	359	362	367	375	379	386	388	390
Cask	M	I	L	V	I	L	I	I	V	L	L	I	F	I
	501	503	505	515	524	530	536	539	544	552	556	563	565	567
Tiam1	Y	F	L	V	A	L	I	I	A	L	L	L	L	V
	858	860	862	875	884	889	895	898	903	911	915	920	922	924

Figure 5.1 – Le cœur PDZ sélectionné En haut, l’alignement des huit séquences sauvages étudiées, les positions du cœur sont en jaunes. Au centre, la structure 3D des six cœurs de \mathcal{S}_1 superposés. En bas, la séquence et les « positions PDB » de chaque cœur.

5.4.6 Les fréquences d'acides aminés

Pour chaque ensemble d'homologues, noté \mathcal{H} , nous calculons des fréquences globales de chaque type d'acide aminé sur toutes les séquences. Les fréquences sont déterminées séparément pour les positions enfouies et pour les positions exposées. Notons-les $f_t^b(\mathcal{H}), f_t^e(\mathcal{H})$, où l'indice t représente un type d'acide aminé et les exposants e et b désignent respectivement les ensembles de positions exposés et enfouis. Les ensembles de fréquences moyennes des huit protéines sont divisés selon deux groupes de protéines, d'une part le sous-ensemble $\mathcal{S}_1 = \{\text{NHREF, INAD, GRIP, Syntenin, DLG2, PSD95}\}$ et d'autre part le groupe $\mathcal{S}_2 = \{\text{Cask, Tiam1}\}$. Enfin, nous calculons la moyenne des $f_t^e(\mathcal{H})$ et des $f_t^b(\mathcal{H})$ sur \mathcal{S}_1 et sur \mathcal{S}_2 . Cela donne deux jeux de deux ensembles cibles de fréquences d'acides aminés $\{f_t^b\}$ et $\{f_t^e\}$. Nous faisons la même chose pour chaque classe de types. Cette partition en deux sous-ensembles va nous permettre d'estimer la transférabilité des énergies de références obtenues à partir d'un sous-ensemble de protéines vers l'autre.

5.5 Séquences calculées

5.5.1 Préparation

Pour réaliser les calculs Monte Carlo, deux segments manquants dans le domaine Tiam1 (résidus 851-854 et 868-869) ont été construits en utilisant le programme Modeller [136]. Le ligand peptidique a été retiré de la structure PDB avant de calculer la matrice d'énergie. Les structures ont été préparées et les matrices d'énergie calculées à l'aide d'une procédure proposée dans des travaux précédents [99, 82] et détaillée en 2.4.2.

5.5.2 Simulation Monte Carlo

La production des séquences est réalisée avec Proteus [101]. Pour optimiser les énergies de références, les positions dans lesquels la séquence native comporte une glycine ou une proline conservent toujours leur type naturel et les positions mutables ne peuvent devenir ni Gly ni Pro. Pour optimiser les énergies de référence, nous sélectionnons, alternativement le long du backbone, une position pouvant muter, puis une position ne pouvant pas muter.

Dans tous les cas, des mutations sont produites au hasard, soumises uniquement à la fonction d'énergie MMGBSA qui entraîne la simulation. Les simulations Monte Carlo utilisent des mouvements à une ou deux positions, où les rotamères, les types d'acides aminés ou les deux peuvent changer. Pour les mouvements à deux positions, la deuxième position est choisie parmi celles qui ont une énergie d'interaction significative avec la première pour au moins une conformation du couple (10 kcal/mol ou plus). De plus, l'échantillonnage est amélioré par l'échange de répliques (REMC), où 8 simulations MC de 500 millions de pas sont exécutées en parallèle à différentes températures et avec échanges périodiques suivant le protocole REMCd qui est décrit en 4.2.2. Pour le calcul de fréquences, seules les séquences produites à la température $kT=0,263$ kcal/mol sont retenues.

5.5.3 Génération de séquence Rosetta

Des simulations Monte Carlo sont également réalisées à l'aide du programme et de la fonction d'énergie Rosetta [133]. Les simulations sont faites en utilisant la version 2015.38.58158 librement disponible en ligne, en utilisant la commande :

```
fixbb -s Cask.pdb -resfile Cask.res -nstruct 10000 -ex1 -ex2 -linmem_ig 10
```

où les options ex1 et ex2 activent une recherche améliorée des rotamères pour les chaînes latérales enfouies. La dernière option correspond au calcul de l'énergie « on the fly » au cours de la recherche MC, et les paramètres par défaut sont utilisés pour les autres options. Comme pour les simulations Proteus, les résidus Gly et Pro présents dans la protéine sauvage ne sont pas autorisés à muter, et les positions qui mutent ne peuvent pas muter en Gly ou Pro. Des simulations sont exécutées pour chacun de nos domaines PDZ, jusqu'à obtenir 10 000 séquences uniques de faible énergie, ce qui correspond à des durées d'exécution d'environ 5 minutes par séquence sur un seul cœur d'un processeur Intel récent, pour un total de 10 heures par protéines en utilisant 80 cœurs. C'est comparable au coût des calculs Proteus, en comptant le temps de calcul de la matrice d'énergie plus celui des simulations Monte Carlo.

5.5.4 Caractérisation des séquences calculées

Les séquences calculées sont comparées à l'alignement Pfam pour la famille PDZ, en utilisant la matrice Blosum40 et une pénalité de gap de -6. Cette matrice est bien adaptée pour comparer des homologies éloignées (les séquences CPD et celles de Pfam). Chaque séquence Pfam est également comparée à l'alignement Pfam, ce qui permet de comparer des séquences calculées et un ensemble de domaines PDZ naturels. Pour ces comparaisons Pfam/Pfam, si un domaine test T fait partie de l'alignement, la comparaison T/T n'est pas prise en compte, pour être plus cohérent avec les comparaisons CPD/Pfam. L'alignement Pfam utilisée est le « RP55 » (voir la section 2.7.4). Les similitudes sont calculées pour les 14 résidus du cœur et pour l'ensemble des positions mutables de la protéine.

Les séquences calculées sont soumises à la bibliothèque de modèles de Markov Cachés Superfamily [103, 104] qui tente de classer les séquences selon la base de données structurelle SCOP [60], voir 2.7.1 pour les détails. Le programme hmmscan est exécuté avec une E-value seuil de 10^{-10} .

5.6 Résultats du modèle NEA

5.6.1 Optimisation du modèle de l'état déplié

Nous optimisons les énergies de référence E_t^r pour les six protéines de \mathcal{S}_1 , en utilisant leurs homologues naturels pour définir les fréquences d'acides aminés cibles. La constante diélectrique ϵ_p de la protéine est fixée à 8. La méthode d'optimisation est la méthode

linéaire (voir 5.2.3). Nous effectuons 20 itérations avec la contrainte des classes et 20 itérations sans cette contrainte, l'optimisation se faisant alors sur tous les types possibles. La fonction proxy calculée sur les groupes de types converge autour des valeurs 0,06 et 0,07 (respectivement pour les énergies enfouies et exposées). Pour les types les valeurs sont 0,08 et 0,14. Cela correspond à des variations pour les E_t^r inférieures à 0,05 kcal/mol pour tous les types d'acides aminés sur les 5 derniers cycles d'optimisation. Le tableau 5.6 donne les énergies de référence convergées.

Table 5.6 – Les énergies de référence (kcal/mol) optimisée sur six protéines.

Type d'acides aminés	enfouis	exposés
ALA	0,00	0,00
ARG	-28,29	-28,90
ASN	-5,94	-6,00
ASP	-9,19	-9,80
CYS	-1,04	-1,04
GLN	-4,72	-4,78
GLU	-7,90	-8,51
HID	11,96	12,39
HIE	11,43	11,85
HIP	14,53	14,96
ILE	4,72	2,11
LEU	1,17	-1,44
LYS	-4,56	-4,47
MET	-2,78	-3,54
PHE	-0,37	-2,55
SER	-3,73	-2,80
THR	-3,82	-3,82
TRP	-1,61	-3,79
TYR	-4,20	-6,10
VAL	0,83	-1,77

Le tableau 5.7 compare les fréquences d'acides aminés des homologues naturels et des séquences CPD. La population des différentes classes d'acides aminés a bien rejoint l'expérience, avec des écarts de moins de 1% dans la majorité des cas, pour les positions exposées et pour les positions enfouies. Seulement deux classes ont des écarts de plus de 2% (2,1 et 2,6 pour Lys et Arg aux positions exposées). L'accord pour les types d'acides aminés est également bon, avec seulement deux écarts de plus de 2% qui correspondent aux deux plus mauvaises classes. Pendant les 20 premiers cycles d'optimisation, la distribution des fréquences intra classe dépend par construction du δE_t^r défini pour chaque classe, qui est calculé par mécanique moléculaire (voir section 5.2.1). La seconde série de 20 itérations permet l'ajustement de ces valeurs.

Table 5.7 – Les compositions en acide aminé (%) des séquences expérimentales et Proteus après optimisation des E_t^s . Les différences entre expérimentale et théorique sont indiquées entre parenthèses. Les types d'acides aminés sont rassemblés selon les groupes d'optimisations.

Res	Experimentale				Proteus			
	Enfoui	Exposé		Enfoui	Exposé			
ALA	10,9		4,6		11,1	17,0	4,4	12,0
CYS	1,3	16,9	0,5	13,4	0,0	(0,1)	0,3	(-1,4)
THR	4,7		8,3		5,9		7,3	
ASP	4,3	6,8	6,0	17,9	4,5	6,7	5,6	16,7
GLU	2,5		11,9		2,2	(-0,1)	11,1	(-1,2)
ASN	2,6	4,7	6,7	12,2	2,5	4,7	7,5	14,0
GLN	2,1		5,5		2,2	(0,0)	6,5	(1,8)
HIP	1,2		5,0		1,0		5,2	
HIE	0,0	1,2	0,0	5,0	0,1	1,1	0,4	
HID	0,0		0,0		0,0	(-0,1)	0,0	(0,6)
ILE	16,0		4,2		16,9	52,1	4,1	14,0
VAL	16,5	50,7	5,4	14,0	16,7	(1,4)	5,6	
LEU	18,2		4,4		18,5		4,3	(0,0)
LYS	2,5	2,5	10,9	10,9	1,5	1,5 (-1,0)	13,0	13,0 (2,1)
MET	0,9	0,9	1,5	1,5	1,6	1,6 (0,7)	1,4	1,4 (-0,1)
ARG	2,8	2,8	8,7	8,7	2,5	2,5 (-0,3)	6,1	6,1 (-2,6)
SER	5,3	5,3	7,6	7,6	4,3	4,3 (-1,0)	8,7	8,7 (1,1)
PHE	4,1	4,1	2,4	2,4	4,5	4,6	2,1	2,1
TRP	0,0		0,0		0,1	(0,5)	0,0	(-0,3)
TYR	2,6	2,6	1,2	1,2	2,2	2,2 (-0,4)	0,4	0,4 (-0,8)
GLY	0,8	0,9	3,1	4,9	0,0	0,0	0,0	0,0
PRO	0,1		1,8		0,0	(-0,9)	0,0	(-4,9)

5.6.2 Tests de reconnaissance de famille

À partir de nos énergies optimisées, nous générerons des séquences pour chaque protéine. Les simulations Proteus utilisent l'algorithme REMC avec huit répliques (ou marcheurs), 750 millions de pas par réplique et des énergies thermiques kT qui varient de 0,125 à 3 kcal/mol. Il s'agit du protocole REMCd détaillé en 4.2.2. Toutes les positions sont autorisées à muter librement vers tous les types d'acides aminés exceptés Gly et Pro. Les simulations ont été faites avec la fonction d'énergie MMGBSA, sans aucun biais vers les séquences naturelles ni aucune limite sur le nombre de mutations. Les 10 000 séquences avec les énergies les plus faibles parmi celles échantillonées par au moins un des marcheurs MC sont retenues pour l'analyse. De la même façon, 10 000 séquences produites par Rosetta sont également retenues. Ces séquences sont analysées par les outils de reconnaissance de pli « Superfamily » (voir 2.7.1). Avec une constante diélectrique de 8,

nous avons obtenu un pourcentage élevé de séquences correctement associées à la famille et superfamille PDZ : 100% pour NHREF, INAD, GRIP et DLG2, 99% pour Syntenin, seule PSD95 donne un score relativement mauvais de 47% pour la famille et 50% pour la superfamille. Les E-values sont inférieures à 10^{-3} pour les affectations à la famille. Ces valeurs sont semblables à celles obtenues par Rosetta pour les cinq premiers cas, par contre l'affectation à la superfamille est meilleure pour Rosetta avec des E-values compris entre $3,7 \cdot 10^{-23}$ et $1,3 \cdot 10^{-9}$, alors que Proteus obtient des E-values compris entre $4 \cdot 10^{-4}$ et $2 \cdot 10^{-12}$ si l'on exclut PSD95. Les détails sont présentés aux tables 5.8 et 5.9.

Table 5.8 – Résultats Superfamily pour les séquences Proteus avec le modèle NEA.

Protéine	Match/seq size	Superfamily Evalue	Superfamily success	Family Evalue	Family success
NHREF	81/91	$2,0 \cdot 10^{-12}$	10000	$1,0 \cdot 10^{-2}$	10000
INAD	84/94	$4,8 \cdot 10^{-11}$	10000	$2,8 \cdot 10^{-3}$	10000
GRIP	82/95	$4,7 \cdot 10^{-8}$	10000	$5,6 \cdot 10^{-3}$	10000
Syntenin	63/91	$4,0 \cdot 10^{-4}$	9999	$1,0 \cdot 10^{-2}$	9999
DLG2	84/97	$3,8 \cdot 10^{-10}$	10000	$3,7 \cdot 10^{-3}$	10000
PSD95	46/97	$7,6 \cdot 10^{-1}$	5029	$4,1 \cdot 10^{-2}$	4719

Table 5.9 – Résultats Superfamily pour les séquences Rosetta

Protein	Match/seq size	Superfamily Evalue	Superfamily success	Family Evalue	Family success
NHREF	79/91	$1,3 \cdot 10^{-13}$	10000	$2,2 \cdot 10^{-3}$	10000
INAD	85/94	$7,4 \cdot 10^{-14}$	10000	$3,7 \cdot 10^{-3}$	10000
GRIP	84/95	$2,2 \cdot 10^{-10}$	10000	$1,2 \cdot 10^{-3}$	10000
Syntenin	76/82	$7,3 \cdot 10^{-13}$	10 000	$1,8 \cdot 10^{-3}$	10 000
DLG2	86/97	$1,3 \cdot 10^{-9}$	10 000	$9,6 \cdot 10^{-4}$	10 000
PSD95	90/97	$3,7 \cdot 10^{-23}$	10 000	$5,2 \cdot 10^{-4}$	10 000

5.6.3 Séquences et diversité de séquences

Une sélection des meilleures séquences calculées par Proteus, au sens de l'énergie, pour le sous-ensemble de six protéines est montrée aux figures 5.12, 5.14, 5.16, 5.18, 5.20 et 5.22. Les homologues naturels pour les huit protéines sont présentés aux figures 5.11, 5.13, 5.15, 5.17, 5.19, 5.21, 5.23 et 5.24. Comme dans de nombreuses études de CPD antérieures [137, 71], l'accord avec l'expérience pour les positions du cœur est très bon, alors que l'accord pour les résidus de surface est nettement plus faible.

La diversité des séquences naturelles et des séquences calculées est caractérisée par l'entropie résiduelle (voir 2.7.7). Comme référence nous utilisons l'ensemble Pfam Seed qui est constitué d'un sous-ensemble représentatif des domaines PDZ naturels (2.7.4). L'entropie S_i est calculée aux positions i de chaque protéine. Nous caractérisons chaque

protéine par la moyenne $\langle e^{S_i} \rangle$ sur les positions i . La diversité des séquences Rosetta est légèrement meilleure avec des valeurs comprises entre 1,40 et 1,68 alors que celles de Proteus sont comprises 1,24 et 1,55. La diversité de Pfam Seed correspond à une entropie (exponentielle) moyenne de 3,11. Le regroupement des séquences calculées de NHREF, INAD, GRIP, Syntenin, DLG2 et PSD95 donne une entropie de 2,88 avec Rosetta et 2,42 avec Proteus. Ainsi six géométries de backbone ne peuvent pas produire les mêmes niveaux de diversité que l'ensemble Seed, construit pour caractériser la diversité des domaines PDZ et notamment la diversité de leur squelette.

Table 5.10 – Moyenne de l'exponentielle de l'entropie sur les ensembles des positions des protéines

Protéine	Proteus	Rosetta	Pfam seed
NHREF	1,38	1,45	3,15
INAD	1,37	1,55	3,06
GRIP	1,33	1,44	3,09
Syntenin	1,39	1,43	3,03
DLG2	1,24	1,57	3,11
PSD95	1,27	1,40	3,15
6 protéines	2,42	2,88	
CASK	1,55	1,68	3,15
TIAM1	1,22	1,57	3,15

5.6.4 Scores de similarité Blosum

Les scores de similarité Blosum40 entre les séquences calculées et les séquences naturelles sur l'ensemble des positions sont globalement faibles (voir la figure 5.2). Les similitudes Proteus et Rosetta chevauchent le bas du pic des scores naturels pour NHREF, INAD, GRIP et DLG2 avec des valeurs Proteus en dessous de celles de Rosetta de quelques dizaines de points : environ 20 pour NHREF, mais près de 50 pour Syntenin. Les résultats Proteus pour PSD95 sont beaucoup moins bons que ceux de Rosetta avec un écart moyen de plus de 70. En ce qui concerne les résidus du cœur, montrés à la figure 5.3 la similitude des séquences calculées avec les séquences naturelles est beaucoup plus forte. Beaucoup de séquences Proteus ont des scores de plus de 30 pour NHREF, INAD et DLG2. Proteus fait jeu égal avec Rosetta sur NHREF et Syntenin, et est globalement meilleur sur INAD et DLG2, et moins bon sur GRIP et PSD95.

5.6.5 Tests de validation croisée

Cette partie a été effectuée en commun avec Nicolas Panel, doctorant de notre laboratoire. Les détails sont publiés dans [50]. Comme premier test de validation croisée, nous appliquons nos énergies de référence aux deux domaines de notre sous-ensemble \mathcal{S}_2 , qui ne fut pas utilisé dans la partie optimisation, voir 5.4.6. Nous

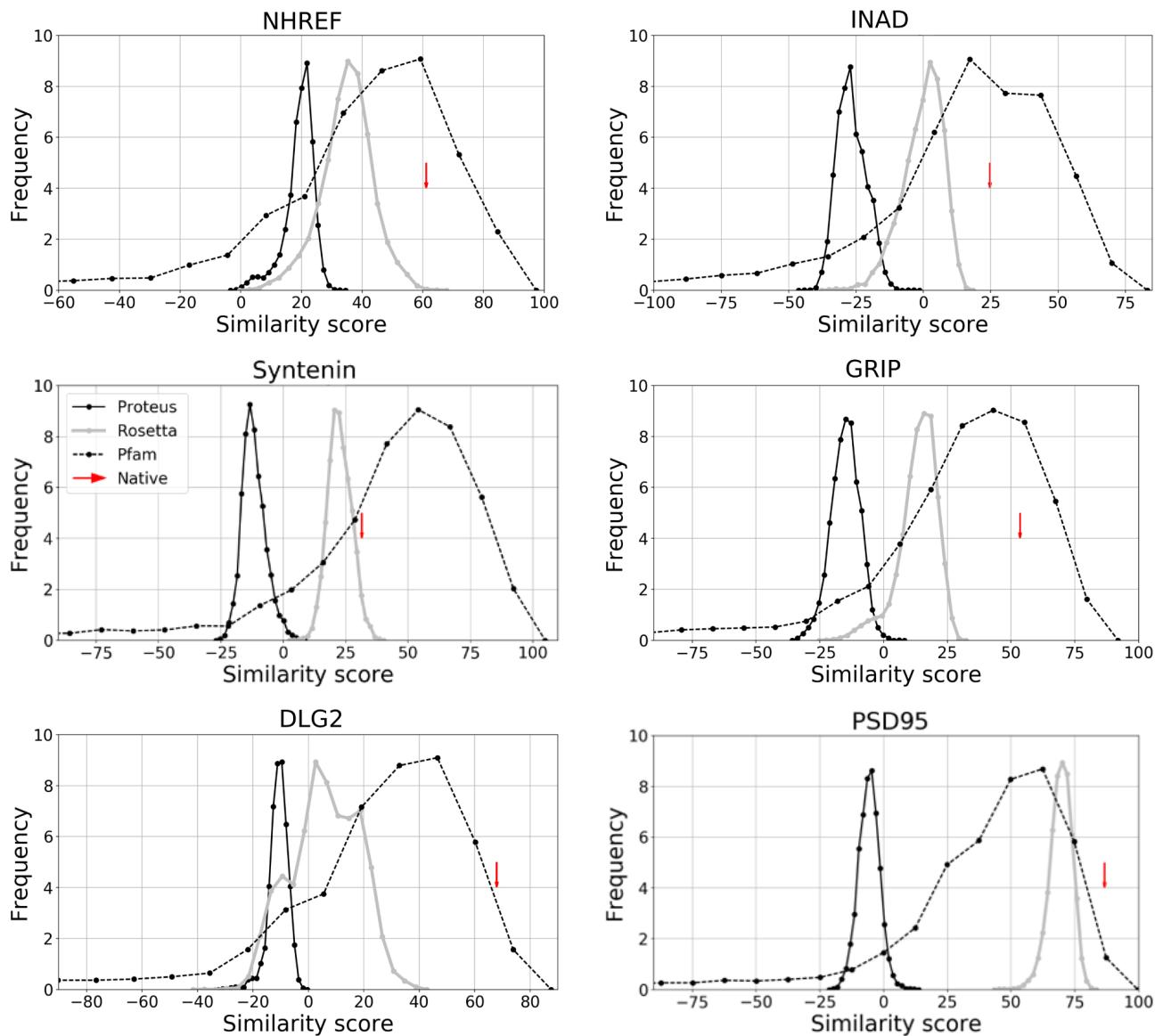


Figure 5.2 – Similarité des séquences des 6 protéines produites par Proteus et Rosetta à l’alignement Pfam RP55, sur l’ensemble des positions.

générions des séquences Tiam1 et Cask qui sont alors soumises aux tests Superfamily et aux calculs de similarité. La performance de Tiam1 sur la super-famille est 84,6%, un peu en dessous des protéines de \mathcal{S}_1 . Le score de Tiam1 pour la reconnaissance de la famille est de 76,6%. Les scores de Cask sont du même ordre que ceux de nos six premières protéines avec 100% de reconnaissance pour la famille et la super-famille, avec des E-values comparables.

Comme validation croisée supplémentaire, les énergies de références ont été optimisées par Nicolas Panel, en utilisant notre sous-ensemble \mathcal{S}_2 comme ensemble alternatif de domaines PDZ, et la troisième méthode d’optimisation, voir 5.12. Nous générions alors, avec ce modèle, des séquences pour Syntenin et DLG2 qui font partie de \mathcal{S}_1 . Encore une fois, les scores sont proches de ceux obtenus avec le modèle optimisé sur \mathcal{S}_1 . Les reconnaissances sont à 100% et les E-values montent de $4,0 \cdot 10^{-4}$ à $1,3 \cdot 10^{-2}$ avec Syntenin pour la super-famille et de $3,8 \cdot 10^{-10}$ à $8,0 \cdot 10^{-9}$ pour DLG2.

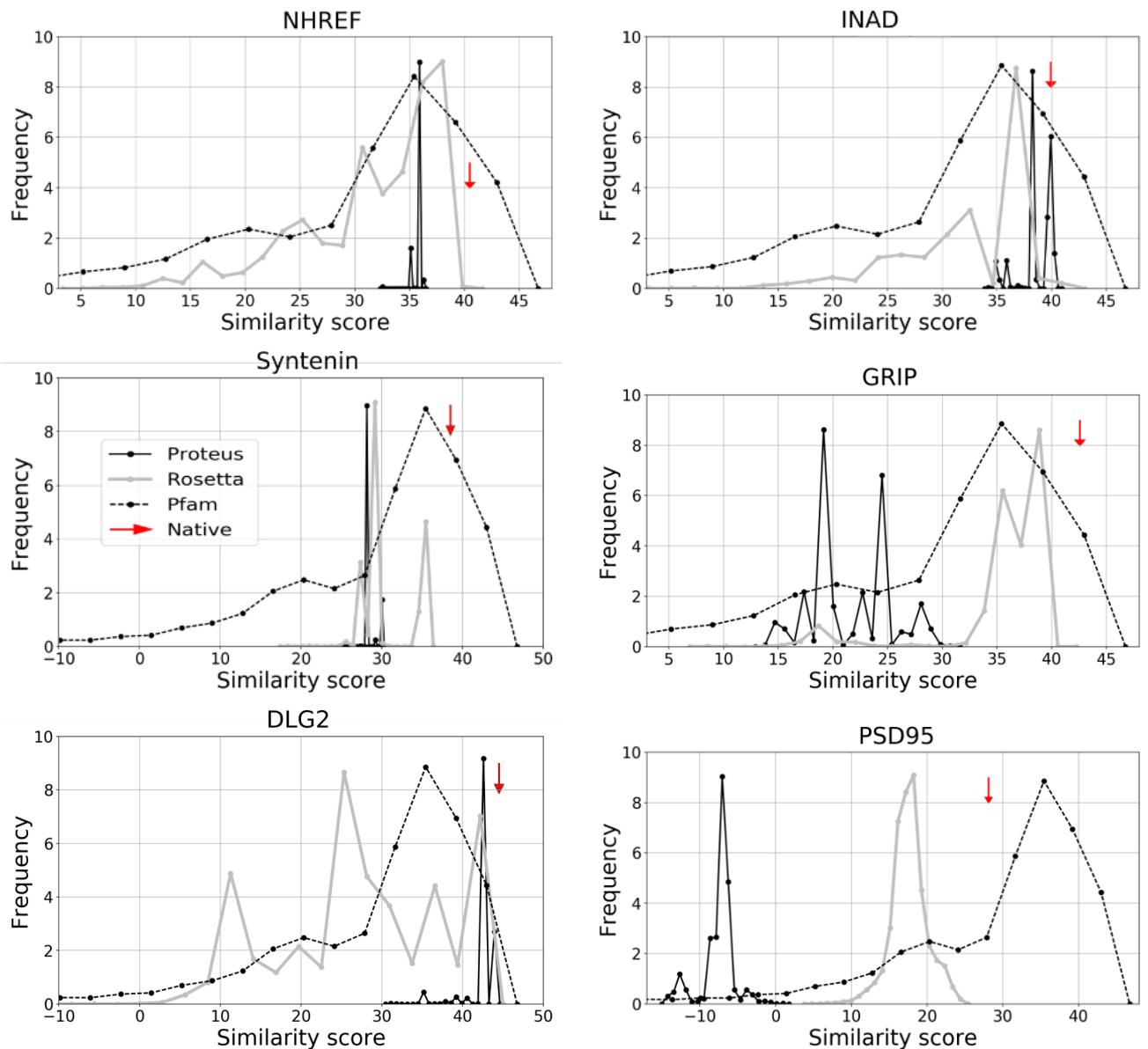


Figure 5.3 – Similarité des séquences des 6 protéines produites par Proteus modèle NEA et Rosetta à l’alignement Pfam RP55, sur les positions du cœur hydrophobe.

Les histogrammes des scores de similarité Blosum montrent que les scores globaux pour Tiam1 et Cask sont très semblables pour les deux modèles. Pour DLG2 et Synténine, nous calculons également les scores de similarité en utilisant les deux modèles. Les scores de similarité avec le modèle S_2 sont légèrement plus faibles qu’avec le modèle S_1 . Le score global a diminué d’environ 20 points pour la Synténine et environ 10 points pour DLG2. Dans l’ensemble, les modèles de validation croisée ont légèrement dégradé les performances. Ainsi, pour tout domaine d’intérêt PDZ, il semble préférable d’optimiser les énergies de référence spécifiquement pour ce domaine plutôt que de transférer des valeurs paramétrées en utilisant d’autres domaines PDZ.

5.7 Résultats du modèle FDB

5.7.1 Optimisation du modèle de l'état déplié

Nous optimisons maintenant les énergies de référence E_t^r en utilisant la variante FDB plus rigoureuse du modèle de solvant GB, voir le paragraphe 2.3.3. Avec cette variante, les fluctuations de la frontière protéine-solvant sont prises en compte (« Fluctuating Dielectric Boundary ») au prix d'une réorganisation du calcul. Nous nous limitons à trois protéines : NHREF, Syntenin et DLG2. La constante diélectrique de la protéine est fixée à 4, et nous utilisons un jeu de paramètres surfaciques optimisés sur ces trois protéines (voir 5.3.1). La méthode d'optimisation est la méthode parabolique (voir 5.2.3) avec 20 itérations avec la contrainte des classes d'acides aminés et 20 itérations sans cette contrainte. Chaque itération se fait avec 100 millions de pas. La fonction proxy calculée sur les classes converge aux valeurs 0,06 et 0,03 pour les énergies enfouies et exposées et aux valeurs 0,03 (enfouis) et 0,02 (exposés) pour la fonction proxy calculées sur les types. Cela donne pour les E_t^r des fluctuations inférieures à 0,05 kcal/mol sur les 5 derniers cycles pour tous les types sauf Arg et Hip dans le cas enfoui, où les variations montent à 0,1 kcal/mol. La population des types d'acides aminés est proche de la population sauvage. Les écarts sur les classes d'acides aminés dans le cas des positions enfouies sont inférieurs à 1% sauf pour le groupe {Asp, Glu} (1,2%) et {Hip, Hie, Hid} (1,4%). Dans le cas exposé, les écarts sont un peu moins bons avec cinq groupes entre 2 et 3%. Les détails sont donnés dans le tableau 5.11.

Pour évaluer l'apport de l'optimisation FDB dans notre modèle, nous optimisons également les E_t^r pour nos trois protéines avec le modèle NEA et la constante diélectrique de la protéine à 4. La même méthode d'optimisation est utilisée. Dans ces conditions, les E_t^r se stabilisent à 0,05 kcal/mol près pour tous les types enfouis ou exposés. Ces quatre jeux d'énergies sont donnés dans le tableau 5.12. Ainsi nous avons deux modèles qui ne diffèrent plus que par la variante GB utilisée.

5.7.2 Tests de reconnaissance de famille

À présent, nous générerons des séquences pour chaque protéine et pour chacun des jeux des E_t^r , FDB et NEA. Le protocole Proteus est identique à celui utilisé plus haut (section 5.6), la constante diélectrique de la protéine étant maintenant de 4. Ici encore, les 10 000 séquences de meilleures énergies parmi celles échantillonées par les répliques REMC sont retenues pour l'analyse. Les résultats Superfamily sont présentés au tableau 5.13. Pour le FDB, la reconnaissance des familles et des superfamilles est de 100% pour les trois protéines tout comme Rosetta. En termes de E-value, Proteus FDB fait jeu égal avec Rosetta, avec des valeurs pour la super-famille allant de $2,85 \cdot 10^{-6}$ à $8,54 \cdot 10^{-14}$ pour Proteus et de $1,3 \cdot 10^{-9}$ à $1,3 \cdot 10^{-13}$ pour Rosetta et des valeurs très proches pour la famille. Pour le NEA, les résultats sont corrects pour la reconnaissance avec 98% pour les trois protéines, mais moins bons pour les E-values de la super-famille.

Table 5.11 – La composition en acide aminé (%) des séquences expérimentales et Proteus après optimisation FDB des énergies de référence. La différence entre expérimentales et Proteus sur les classes est donnée entre crochets.

Res	3 protéines expérimentales				FDB			
	Enfoui		Exposé		Enfoui		Exposé	
	type	classe	type	classe	type	classe	type	classe
ALA	8,7		5,5		9,6		1,8	
CYS	1,9	16,8	0,4	13,6	2,8	17,1 [-0,3]	0,6	11,3 [2,3]
THR	6,2		7,7		4,7		8,9	
SER	4,4	4,4	6,7	6,7	5,2 [-0,8]		7,9	7,9 [-1,2]
ASP	4,8	7,4	6,1	17,1	5,8	8,6	8,1	20,6
GLU	2,6		11,0		2,8	[-1,2]	12,5	[-3,5]
ASN	3,4	5,1	6,9		4,2	6,0	8,8	16,0
GLN	1,7		5,8	12,7	1,8	[-0,9]	7,2	[-3,3]
HIP	2,0		5,9		0,0	0,6	0,4	
HIE	0,0	2,0	0,0	5,9	0,5	[1,4]	2,7	
HID	0,0		0,0		0,1		1,9	[0,9]
ILE	12,4		4,1		11,2	49,4	0,6	
VAL	21,1	50,3	5,1	14,0	21,2 [0,9]		5,2	12,5 [1,5]
LEU	16,8		4,8		17,0		6,7	
MET	1,3	1,3	1,8	1,8	1,0 [0,3]	1,0	2,2	2,2 [-0,4]
LYS	3,4	3,4	10,8	10,8	4,2 [-0,8]	4,2	12,4	12,4 [-1,6]
ARG	1,1	1,1	9,8	9,8	1,8 [-0,7]	1,8	11,8	11,8 [-2,0]
PHE	4,6	4,6	2,4		3,9	3,9	0,0	0,0
TRP	0,0		0,1	2,5	0,0 [0,7]	0,0		[2,5]
TYR	2,4	2,4	1,2	1,2	2,0 [0,4]	2,0	0,0	0,0 [1,2]
GLY	1,0		1,9		0,0	0,0	0,0	
PRO	0,1	1,1	1,8	3,7	0,0 [1,1]	0,0	0,0	[3,7]

5.7.3 Scores de similarité Blosum

Les scores de similarité Blosum40 sont calculés entre les séquences CPD et les séquences Pfam. Nous calculons ces similarités pour les séquences Proteus FDB, Proteus NEA et Rosetta, sur l'ensemble des positions sauf une petite partie au début et à la fin de la séquence, parce qu'elles ne sont pas conservées dans l'alignement Pfam. Cela représente moins de 10% de la longueur de la séquence. Pour NHREF, l'approximation FDB améliore très nettement les scores de Proteus NEA avec un gain d'environ 50 points. Cela place Proteus à peu près au niveau de Rosetta. Pour Syntenin, les écarts sont plus serrés, avec le FDB légèrement moins bon que le NEA, mais proche de Rosetta. Dans le cas de DLG2 le FDB domine les séquences NEA et près de la moitié de celles produites par Rosetta. Sur les positions du cœur hydrophobe, les résultats Proteus sont excellents, avec le plus souvent des similarités avec Pfam compris entre 30 et 40. Le NEA et le FDB font jeu égal

Table 5.12 – Les énergies de référence obtenues avec l'optimisation sur 3 protéines. La constante diélectrique est fixée à 4.

acides aminés	NEA		FDB	
	Pos. Enf.	Pos Exp.	Pos. Enf.	Pos Exp.
ALA	0,00	0,00	0,00	0,00
CYS	-0,89	-2,57	-1,06	-1,64
THR	-5,31	-8,075	-4,84	-6,68
SER	-5,55	-6,55	-4,45	-5,24
ASP	-17,26	-22,06	-14,56	-18,82
GLU	-16,12	-20,68	-14,52	-18,21
ASN	-16,38	-20,41	-14,02	-17,80
GLN	-14,00	-18,41	-13,14	-16,61
HID	11,21	6,95	10,85	8,13
HIE	10,63	6,15	10,41	7,37
HIP	15,17	10,72	12,86	10,98
ARG	-53,40	-57,36	-51,37	-54,76
LYS	-8,20	-12,34	-8,24	-11,35
ILE	6,76	3,44	5,50	3,06
VAL	0,43	-2,19	-0,05	-1,66
LEU	0,52	-3,72	0,00	-2,94
MET	-1,61	-3,21	-2,85	-3,09
PHE	1,86	-2,68	0,17	-3,18
TRP	-0,23	-7,67	-1,94	-5,53
TYR	-5,10	-10,90	-5,91	-10,14

sur DLG2, le NEA étant au-dessus pour les deux autres. Il n'y a donc pas d'amélioration sur le cœur ce qui s'explique par le fait que ces résidus sont trop éloignés du solvant pour bénéficier de FDB. Les scores Rosetta pour Syntenin sont proches, mais pour les deux autres protéines, les scores sont nettement plus variables et globalement moins bons. Tout cela est représenté à la figure 5.4.

5.7.4 Taux d'identité à la séquence native

Pour chaque protéine, nous calculons le taux d'identité des 10 000 séquences de meilleures énergies par rapport à la séquence native, ainsi que pour les 10 000 séquences Rosetta, voir 2.7.3. On considère uniquement les positions mutables sans celles aux extrémités des séquences comme expliqué au paragraphe précédent. Ce taux varie pour Proteus FDB entre 24% et 33% et entre 20% et 33% pour la version NEA. Pour Rosetta les taux se situent entre 35 et 40% avec un écart de 11% pour NHREF sur le FDB et de 7% pour les deux autres protéines (voir la table 5.14). Les séquences Rosetta sont nettement plus proches des séquences natives que celles de Proteus, avec environ six mutations de moins en moyenne.

Model	Protein size	Match/seq E-value	Superfamily success	Superfamily E-value	Family success	Family
Proteus FDB epsilon=4	NHREF	80/91	8,54 10 ⁻¹⁴	10000	8,94 10 ⁻³	10000
	Syntenin	70/82	2,85 10 ⁻⁶	10000	2,69 10 ⁻³	10000
	DLG2	88/97	3,26 10 ⁻¹²	10000	1,96 10 ⁻³	10000
Rosetta	NHREF	79/91	1,3 10 ⁻¹³	10000	2,2 10 ⁻³	10000
	Syntenin	76/82	7,3 10 ⁻¹³	10000	1,8 10 ⁻³	10000
	DLG2	86/97	1,3 10 ⁻⁹	10000	9,6 10 ⁻⁴	10000
Proteus NEA epsilon=4	NHREF	62/91	3,22 10 ⁻³	9857	1,00 10 ⁻²	9857
	Syntenin	70/82	2,83 10 ⁻³	9879	3,62 10 ⁻³	9879
	DLG2	83/97	1,66 10 ⁻³	9876	3,18 10 ⁻³	9876

Table 5.13 – Résultats Superfamily pour les séquences Proteus avec le modèle FDB (énergies de références optimisées sur 20 cycles selon les classes + 20 cycles selon les types), le modèle NEA et Rosetta.

Séquences	Proteus FDB	Proteus NEA	Rosetta
NHREF	24	20	35
Syntenin	31	33	38
DLG2	33	31	40

Table 5.14 – Pourcentage d’identité moyen à la séquence native

5.7.5 Logos des séquences obtenues

Finalement, nous montrons les séquences obtenues. Elles sont représentées sous forme de logos à la figure 5.5 pour les positions du cœur hydrophobe, et la figure 5.6 pour les positions exposées. L’accord avec les séquences naturelles sur les positions du cœur est très bon et l’accord sur les positions exposées est nettement moins bon. Mais au regard de la diversité des types aux positions exposées dans les séquences naturelles de Pfam, le consensus entre les séquences naturelles est lui-même quasi inexistant.

5.8 Application : Croissance du noyau hydrophobe

Comme application de nos modèles optimisés, nous examinons une possibilité de « design » du cœur hydrophobe des domaines PDZ. Deux de nos domaines PDZ, Tiam1 et Cask, sont soumis à une simulation REMC avec une succession de fonctions d’énergie biaisées qui favorisent progressivement les résidus hydrophobes. La première simulation comprend un terme d’énergie avec un biais $\delta = 0,4$ kcal/mol par position, qui pénalise les types d’acides aminés hydrophobes (I, L, M, V, A, W, F et Y). Le biais augmente alors graduellement et passe par les valeurs intermédiaires $\delta = 0,2$, $\delta = 0$ et $\delta = -0,2$ kcal/mol. La dernière simulation comprend un terme de biais $\delta = -0,4$ kcal/mol (par position) qui favorise les types hydrophobes. En diminuant progressivement la valeur du biais d’énergie δ , nous « titrons » ainsi les résidus hydrophobes.

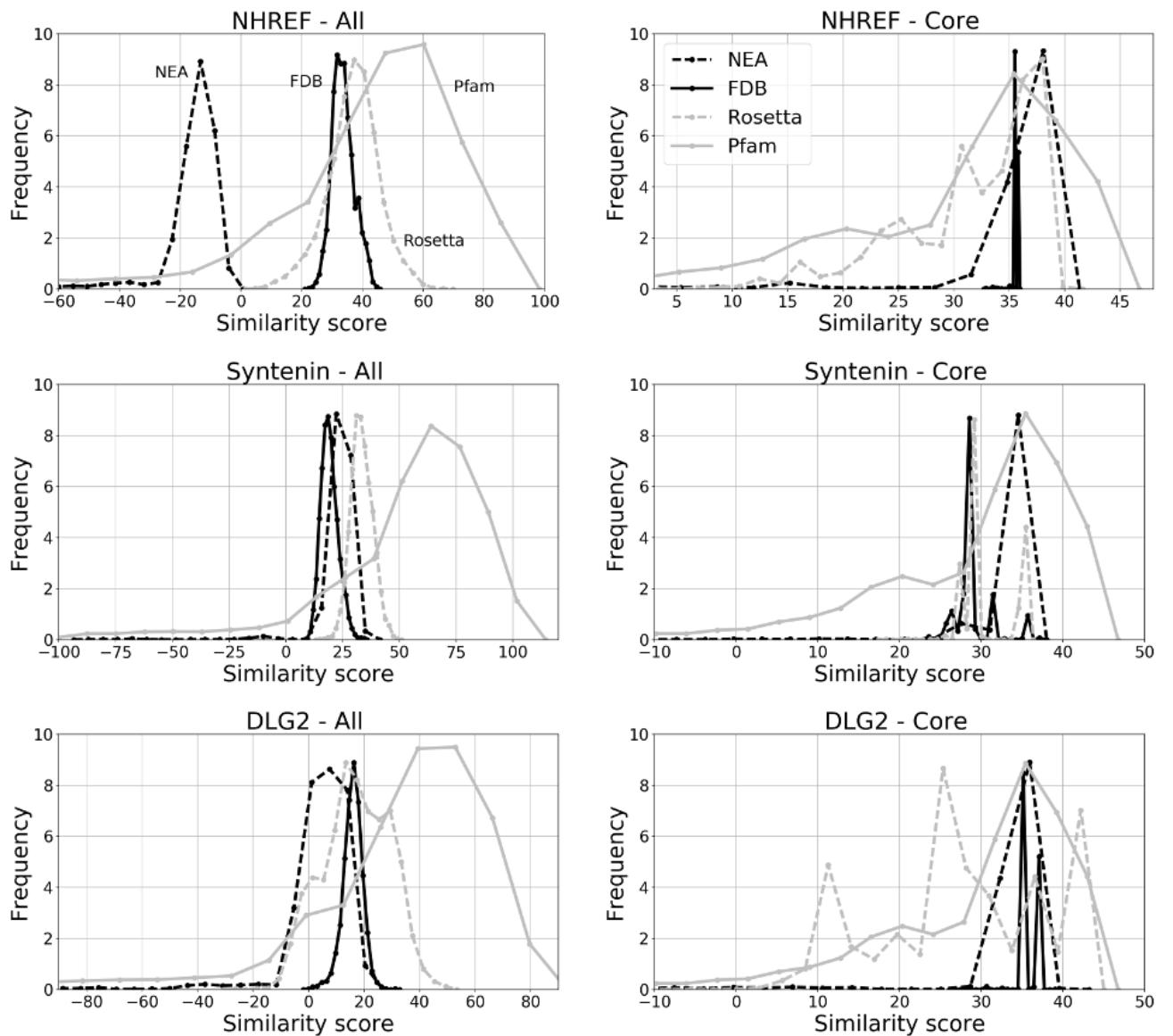


Figure 5.4 – Similarité des séquences Proteus (NEA et FDB), Rosetta et des séquences de l’alignement Pfam RP55, sur toutes les positions à gauche et sur les positions du cœur à droite.

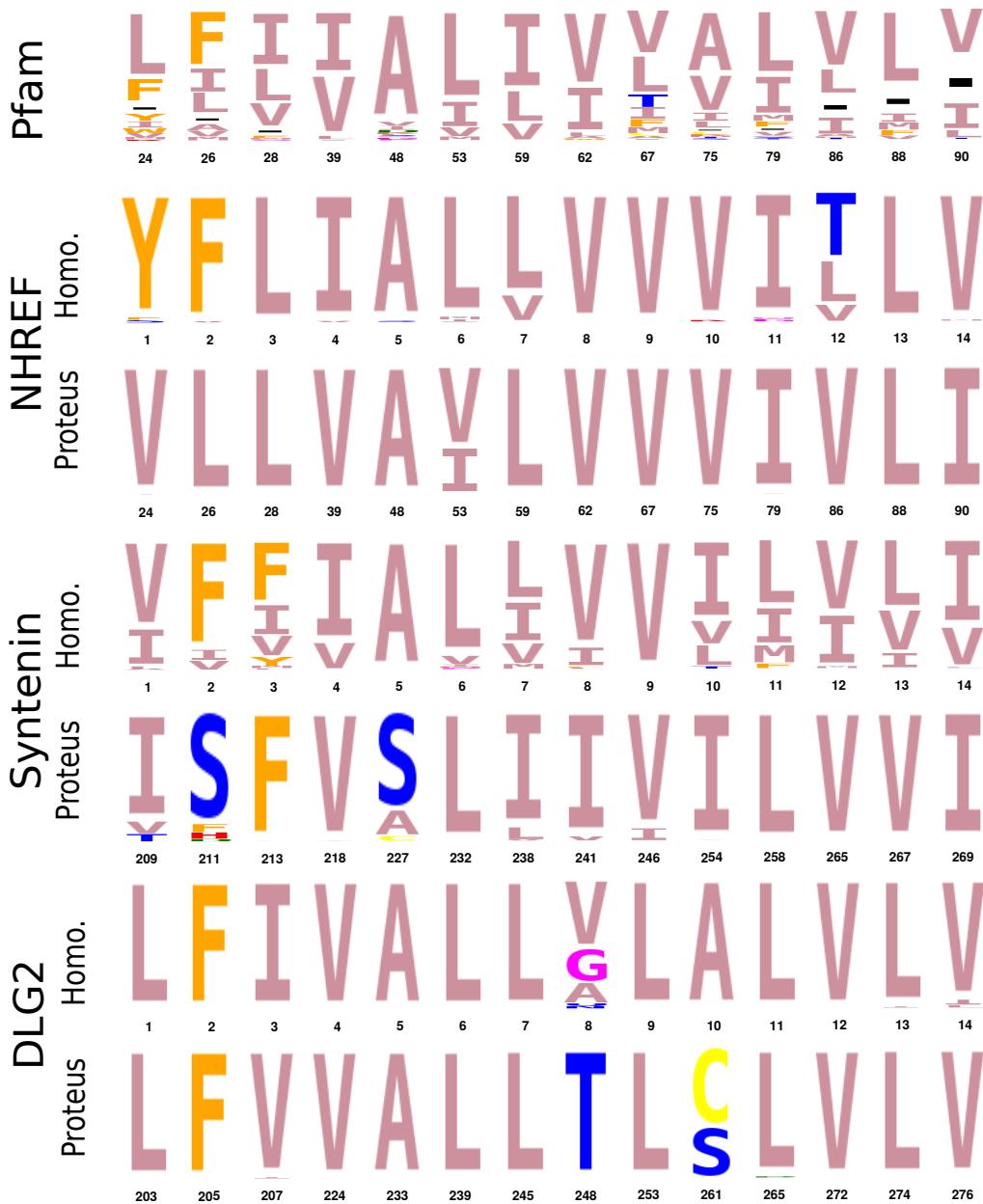
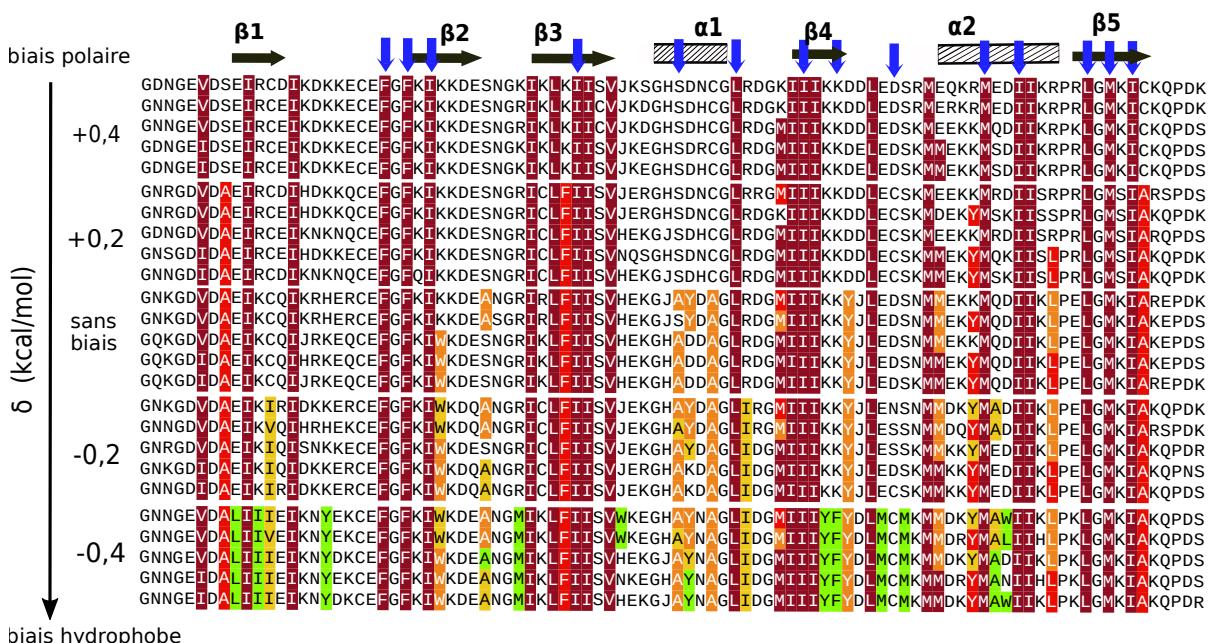


Figure 5.5 – Les séquences conçues par Proteus, les homologues à la native et les séquences naturelles représentées sous forme de logo pour les positions du cœur hydrophobe



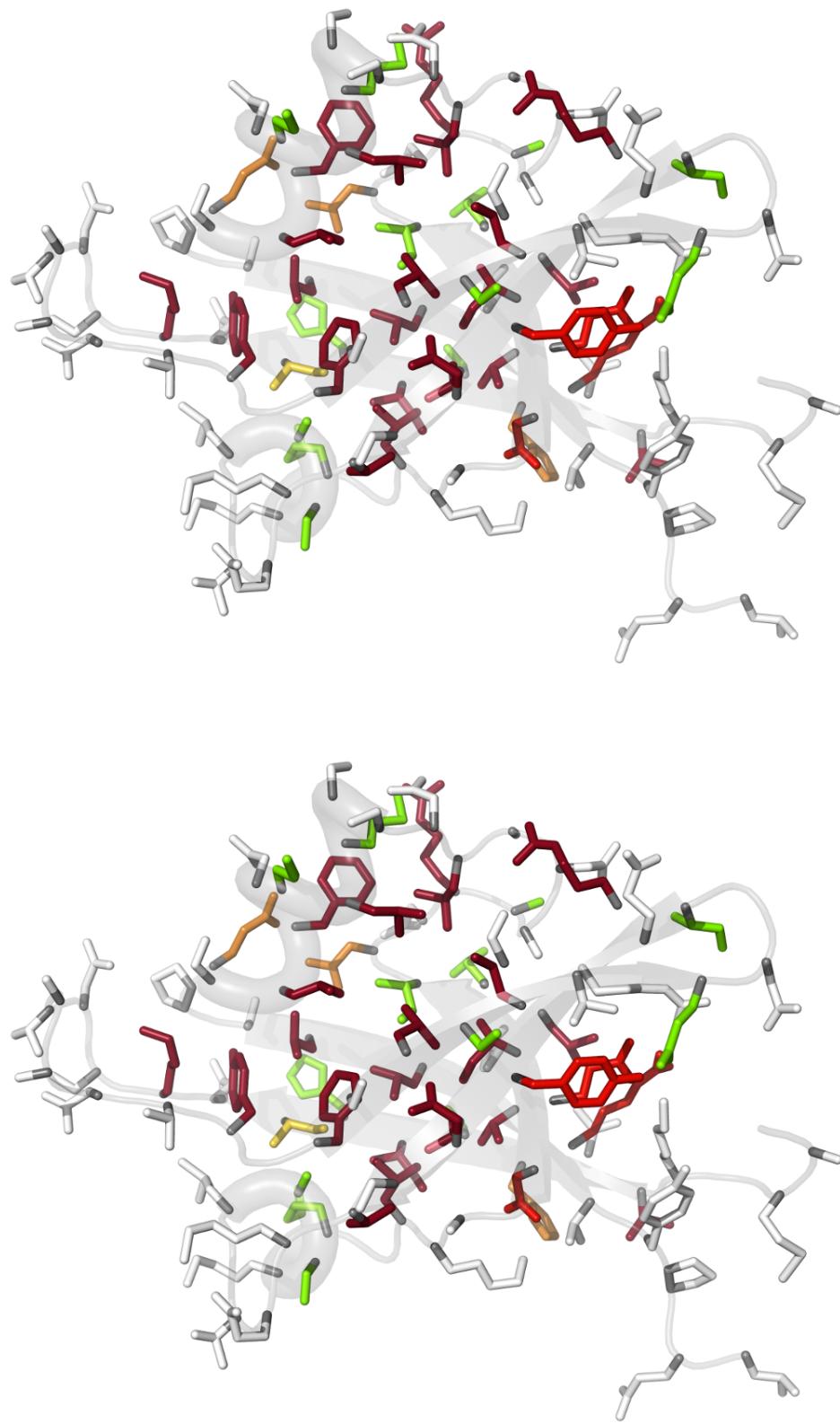
Figure 5.6 – Les séquences conçues par Proteus, les homologues à la native et les séquences naturelles représentées sous forme de logo pour les positions exposées

Figure 5.7 – Séquences Tiam1 obtenues avec un delta des énergies de références à -0,4, -0,2, 0, 0,2 et 0,4 et la structure native. Les hydrophobes pour des deltas de -0,4, -0,2, 0, 0,2 et 0,4 sont représentés par un dégradé allant du rouge foncé au vert clair en passant par le jaune.



Les flèches bleues indiquent les positions du cœur hydrophobe PDZ défini à partir de notre sélection de six domaines. Chaque groupe de séquences est une sélection à δ fixé parmi les séquences de plus faible énergie.

Figure 5.8 – Structure native Tiam1 avec les hydrophobes pour des δ de -0,4, -0,2, 0, 0,2 et 0,4 sont représentés par un jeu de couleurs allant du rouge foncé au vert clair en passant par le jaune.



Les résultats pour Tiam1 sont présentés à la figure 5.7 et à la figure 5.8. À la plus grande valeur de δ le cœur hydrophobe de Tiam1 est réduit, avec environ 10 positions d'acides aminés sur 94 qui changent en un type polaire, comparés aux séquences générées sans biais. Les positions modifiées se situent principalement sur le bord extérieur du cœur. À la valeur intermédiaire de 0,2 kcal/mol, le cœur hydrophobe ne compte plus que 4 ou 5 changements en type polaire. À la valeur de δ la plus négative, le cœur hydrophobe devient plus grand, s'étendant vers les régions de surface, avec globalement 14 positions polaires changées en types hydrophobes. Ainsi le nombre de positions modifiées est approximativement symétrique (environ +/- 12 changements), reflétant le biais. Environ 2/3 des changements se produisent dans des éléments de structure secondaire. Dans l'ensemble, les propensions observées de chaque position à devenir polaire ou hydrophobe en présence d'un biais de pénalité petit ou grand d'énergie δ peuvent être considérées comme un indice de design hydrophobe. Ici, 11 des 14 positions du cœur PDZ (toute sauf les positions 884, 898 et 903) sont restées hydrophobes au plus haut niveau de biais polaire, avec à peu près 13 autres positions, indiquant que ces positions ont la plus grande propension à être hydrophobes. De plus, près de 14 positions ont basculé de polaire à hydrophobe avec le biais le plus élevé, indiquant que ces positions aussi ont une certaine propension à être hydrophobes. Les résultats pour Cask sont similaires, avec 11 positions changées en polaire au plus haut biais polaire et 9 changées en hydrophobe au plus haut biais hydrophobe, voir 5.10.

Nous introduisons alors un indice pour décrire le nombre de changements relatifs de type d'acide aminé par unité d'énergie du biais. Cet indice ψ_h est défini comme le nombre δN de positions qui ont changées de non polaire à polaire, divisé par le produit de la variation δE dans l'énergie de polarisation et le nombre moyen N de positions non polaires à biais nul. Nous appelons ψ_h la sensibilité hydrophobe. Pour le domaine PDZ Tiam1, ce calcul donne : $\psi_h = \frac{1}{N} \frac{\delta N}{\delta E} = 0,9$ changements par position et par kcal/mol. Pour Cask, la sensibilité hydrophobe est $\psi_h = 0,7$ changements par position et par kcal/mol.

5.9 Conclusion

5.9.1 Modèle mis en œuvre

Nous avons paramétré notre modèle CPD pour la conception informatique de domaines PDZ, mis en œuvre dans le logiciel Proteus. Pour la modélisation de l'état replié, nous avons utilisé un champ de force protéique de qualité. Nous avons effectué des premiers paramétrages sur un ensemble de huit domaines PDZ, dont deux utilisés pour évaluer la transférabilité des résultats. Nous avons utilisé un premier traitement du solvant « GB », le modèle NEA et une constante diélectrique ϵ_P égale à 8. Puis nous avons effectué de nouveaux paramétrages sur un ensemble de trois domaines PDZ, avec une constante diélectrique ϵ_P égale à 4 et avec deux traitements du solvant : le NEA et la nouvelle méthode « GB », le modèle FDB et un jeu de paramètres surfaciques optimisé.

Figure 5.9 – Structure native Cask avec les hydrophobes pour des δ de -0,4,-0,2,0,0,2 et 0,4 sont représentés par un dégradé allant du rouge foncé au vert clair, en passant par le jaune.

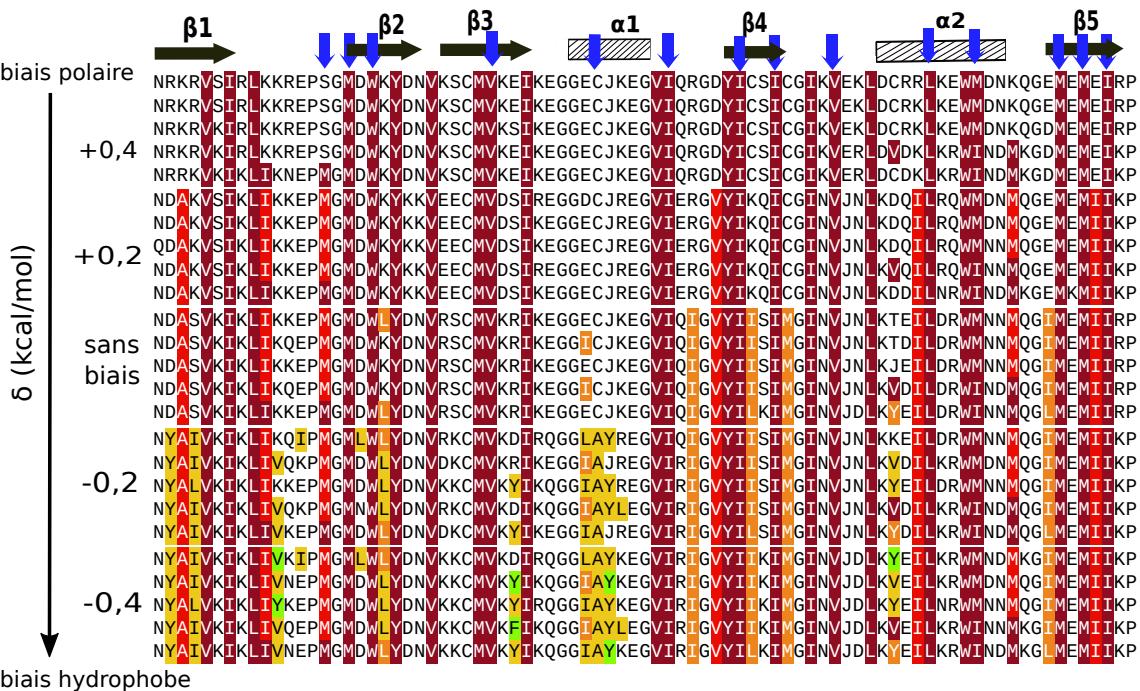


Figure 5.10 – Séquences Cask obtenues avec un delta des énergies de références à -0,4,-0,2,0,0,2 et 0,4 et la structure native. Les hydrophobes pour des deltas de -0,4,-0,2,0,0,2 et 0,4 sont représentés par un jeu de couleurs allant du rouge foncé au vert clair, en passant par le jaune.

Pour les chaînes latérales, nous utilisons une bibliothèque de rotamères simple et discrète et une courte minimisation de chaque paire pendant le calcul de la matrice d'énergie, pour atténuer l'approximation de la discréttisation des rotamères. La fonction d'énergie et la description des rotamères ont été testées de manière approfondie et ont démontré de très bonnes performances pour les tests de reconstruction de chaînes latérales [58] (comparable au programme très populaire Scwrl4 [57]).

La représentation de l'état déplié utilise un modèle simple caractérisé par un ensemble de potentiels chimiques d'acide aminé empiriques ou énergies de référence. Ces énergies sont déterminées par une procédure de maximisation de vraisemblance, décrite ici, afin de reproduire la composition d'acides aminés d'homologues naturels soigneusement sélectionnés. L'état déplié utilisé ici bénéficie d'un raffinement supplémentaire, puisque des valeurs d'énergies de référence distinctes sont utilisées pour les positions d'acides aminés selon qu'elles soient enfouies ou exposées à l'état replié.

Cette méthode suppose qu'il existe une structure résiduelle à l'état déplié, où certaines positions sont plus enfouies que d'autres. En outre, cela devrait rendre le paramétrage plus robuste et moins sensible à la taille et à la structure des homologues naturels utilisés pour définir les compositions d'acide aminé cibles, car les fréquences d'acide aminé des positions exposées et des régions enfouies sont calculées séparément. En principe, cela double le nombre d'énergies de référence à ajuster. Cependant, nous avons réduit ce nombre en introduisant des classes de similarités d'acide aminé, avec une seule énergie de

référence ajustable par classe. Cette contrainte est levée dans la seconde moitié des cycles d'optimisation. Lors de l'optimisation des énergies de référence, nous effectuons, des calculs de séquences pour chaque protéine de notre jeu de test où une position sur deux peut muter (à l'exception de Gly et Pro), avec une simulation distincte pour chaque moitié. Ainsi, lors de l'optimisation des paramètres, une position mutable est toujours entourée d'un environnement identique au type sauvage au moins sur les deux positions immédiatement voisines sur le squelette. Les calculs s'appuient sur une méthode d'exploration Monte Carlo avec échange de réplique, qui utilise un demi-milliard de pas par simulation et produit des milliers de séquences par simulation.

Le modèle présente plusieurs limitations, dont la plupart sont très répandues en CPD. La première est l'utilisation de la stabilité des protéines comme seul critère de conception, sans prendre en compte explicitement la spécificité du pli [100, 30], la protection contre l'agrégation ou des considérations fonctionnelles comme la liaison des ligands. Toutefois, nous notons que les tests superfamily n'ont pas entraînés de mauvaises affectations (séquences perçues comme préférant un autre pli SCOP), donc en pratique, la spécificité du pli est vérifiée.

Une limitation supplémentaire est introduite par l'utilisation d'un squelette protéique fixe lors du calcul de la matrice d'énergie. En fait, le squelette n'est pas vraiment fixe. Certains mouvements sont autorisés, à travers l'utilisation d'une constante diélectrique protéique supérieure à 1 ($\epsilon_p = 4$ ou 8) [30]. Cette valeur diélectrique signifie que la structure protéique (y compris son squelette) est autorisée à se réorganiser en réponse à des mutations ou changements de rotamères. Cependant, la réorganisation est modélisée non pas explicitement, mais implicitement, et elle n'implique pas de mouvement des centres atomiques ou de leur sphère de Van der Waals associée. Ainsi, le squelette ne peut pas se réorganiser en réponse à une répulsion stérique produite par des mutations ou des changements de rotamères. L'utilisation d'un squelette fixe peut être en partie compensée en concevant plusieurs structures PDZ. Par exemple, la mise en commun des séquences calculées sur six protéines a donné une entropie moyenne de séquence nettement plus proche de celle de l'ensemble expérimental Pfam. Une nouvelle méthode pour la conception de protéine multi-backbone a récemment été développée dans Proteus, sur la base d'une méthode Monte Carlo hybride qui préserve l'échantillonnage de la distribution de Boltzmann [138]. Cette méthode pourra être appliquée dans les prochaines études.

Une autre limitation de notre modèle est la nécessité, pour des résultats optimaux, de paramétrier les énergies de référence spécifiquement pour un ensemble donné de protéines. Cette étape est bien automatisée et de façon très parallèle. Cependant, cela implique plusieurs choix qui sont partiellement arbitraires. Ceux-ci comprennent le choix d'un ensemble de domaines protéiques pour représenter la protéine ou la famille d'intérêt. Nous devons également choisir un seuil de similarité pour définir les homologues cibles à partir desquels sont calculées les compositions expérimentales d'acides aminés. Ici, nous avons choisi d'utiliser les homologues de chaque membre de la famille, de calculer leurs compositions, puis de moyenner sur les familles. Cette méthode a bien fonctionné, mais

d'autres choix sont possibles et des travaux complémentaires sont nécessaires pour pouvoir tirer des conclusions définitives sur ces choix.

5.9.2 Tests et application

Les séquences conçues par Proteus sont comparées aux séquences naturelles, à travers des tests de reconnaissance du pli, des calculs de similarité, des calculs d'entropie et des taux d'identité de séquences. Dans les simulations, nous concevons la totalité de la séquence de la protéine, de sorte que toutes les positions (à l'exception de Gly et Pro) peuvent muter librement, soumis à la seule contrainte de générer une composition moyenne en acides aminés similaire à la composition moyenne expérimentale (à travers les énergies de références). Malgré la quasi-absence de contraintes expérimentales, les séquences obtenues ont une forte similitude globale avec les séquences naturelles de Pfam, mesurée par les scores de similarité Blosum40. Les scores obtenus sont, pour l'essentiel, comparables aux scores de similarité entre les paires de séquences Pfam. La similitude est très forte pour les résidus au cœur de la protéine, comme cela a été observé dans des études CPD précédentes [99, 30]. En revanche, pour les résidus pris sur l'ensemble de la protéine, les scores de similarité sont plus faibles, mais l'approximation FDB couplée à une constante diélectrique ϵ_p égale à 4 donne toujours des séquences similaires à des homologues naturels modérément éloignés.

Notez que de nombreux résidus de surface sont impliqués dans des interactions fonctionnelles, comme les onze résidus de liaison aux peptides dans les domaines PDZ. Les résidus de surface sont également sélectionnés selon l'évolution pour éviter l'agrégation ou des adhésions indésirables. Ces contraintes fonctionnelles ne sont pas explicitement prises en compte dans notre protocole de design. Malgré ces difficultés sur les résidus de surface, la reconnaissance de pli avec l'outil Superfamily appliquée aux meilleurs modèles conçus est presque parfaite. Les tests de reconnaissance de pli antérieurs qui utilisaient une fonction d'énergie plus simple donnaient un taux de reconnaissance de pli inférieur, environ 85% (pour un ensemble de tests plus large et plus diversifié) et des similitudes inférieures [139, 82]. De toute évidence, l'utilisation combinée d'un champ de force protéique amélioré, du solvant GB FDB et des énergies de référence spécifiques à la famille conduisent à des séquences calculées proches des séquences natives.

Les séquences Proteus ont également été comparées aux séquences obtenues avec le logiciel Rosetta, qui a lui-même été testé de manière approfondie. Sur la base des scores de similarité de Blosum (par rapport aux séquences naturelles dans Pfam) et des tests de reconnaissance du pli, les séquences Proteus et Rosetta sont globalement de même qualité. Cependant, Rosetta fait moins de mutations que Proteus ; de sorte que les scores d'identité, par rapport à la protéine de type sauvage correspondante, sont entre 7% et 11% plus haut chez Rosetta que pour la version FDB de notre modèle. Proteus modifie environ six positions en plus, en moyenne, par domaine PDZ. Pour finir, nous signalons que certaines séquences Proteus de Nicolas Panel se replient expérimentalement, voir [140].

Annexe du chapitre 5

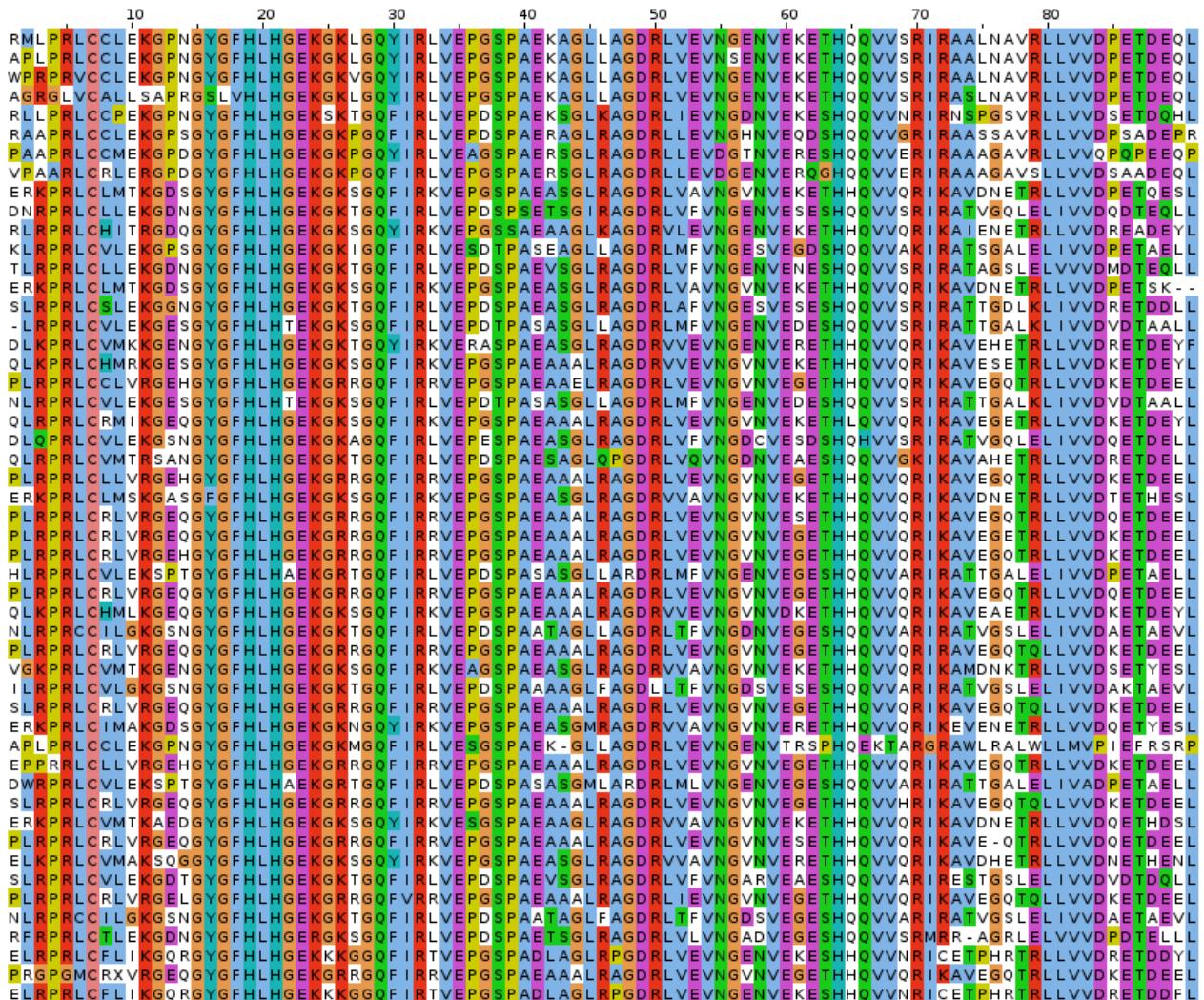


Figure 5.11 – L'alignement de notre sélection de séquences homologues à la protéine NHREF (code PDB : 1G9O)

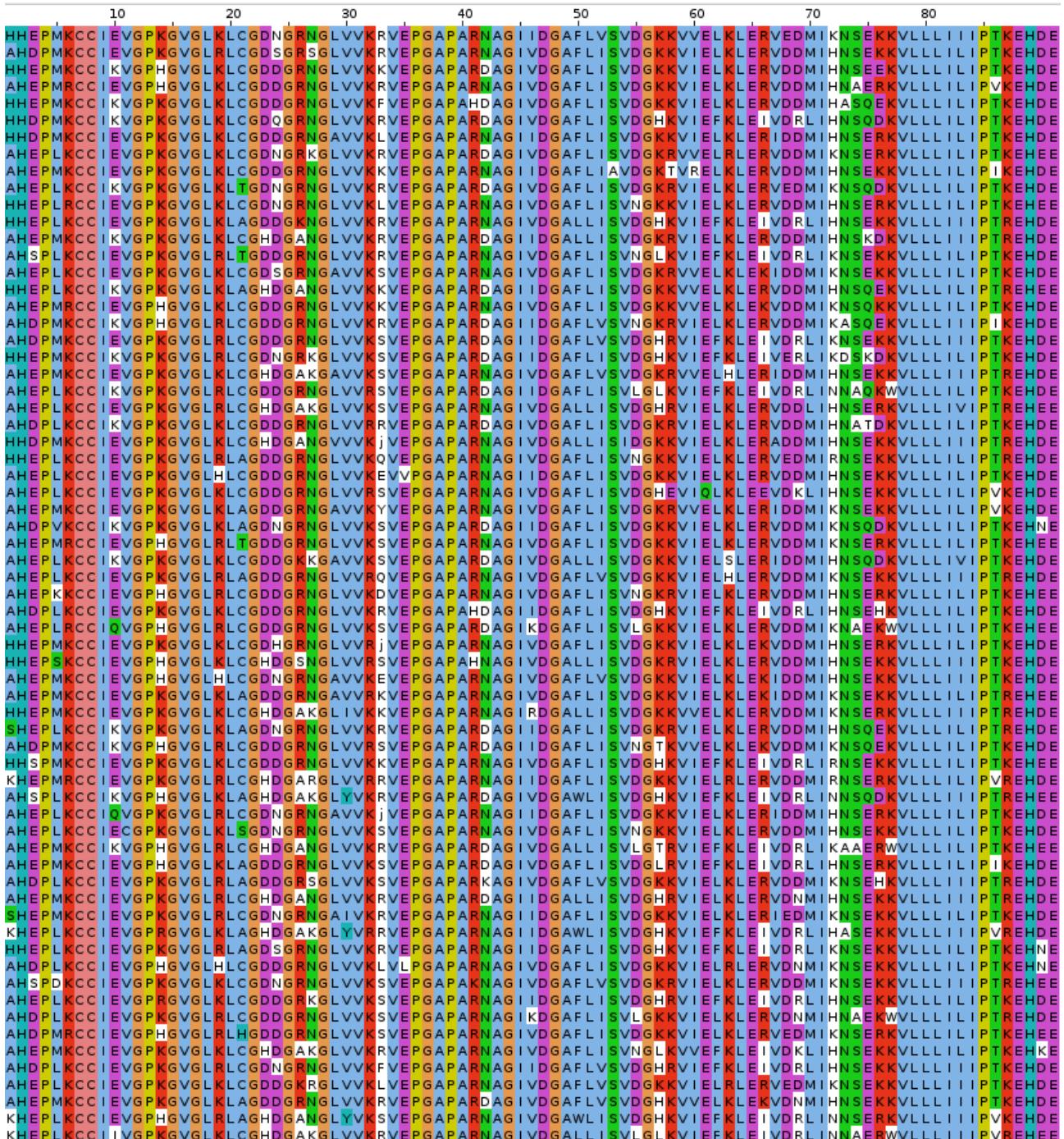


Figure 5.12 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine NHREF (code PDB : 1G9O), modèle NEA

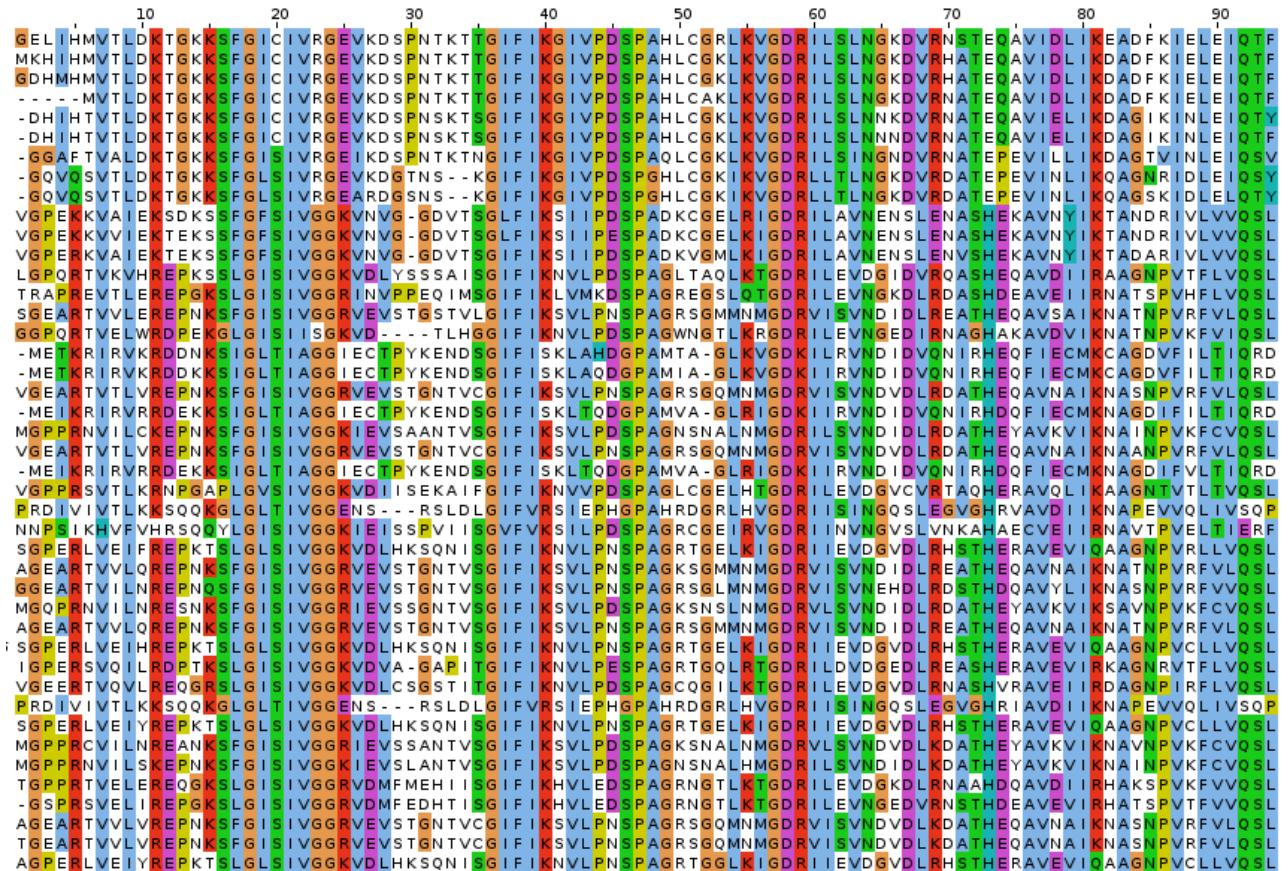


Figure 5.13 – L’alignement de notre sélection de séquences homologues à la protéine INAD (code PDB : 1IHJ)

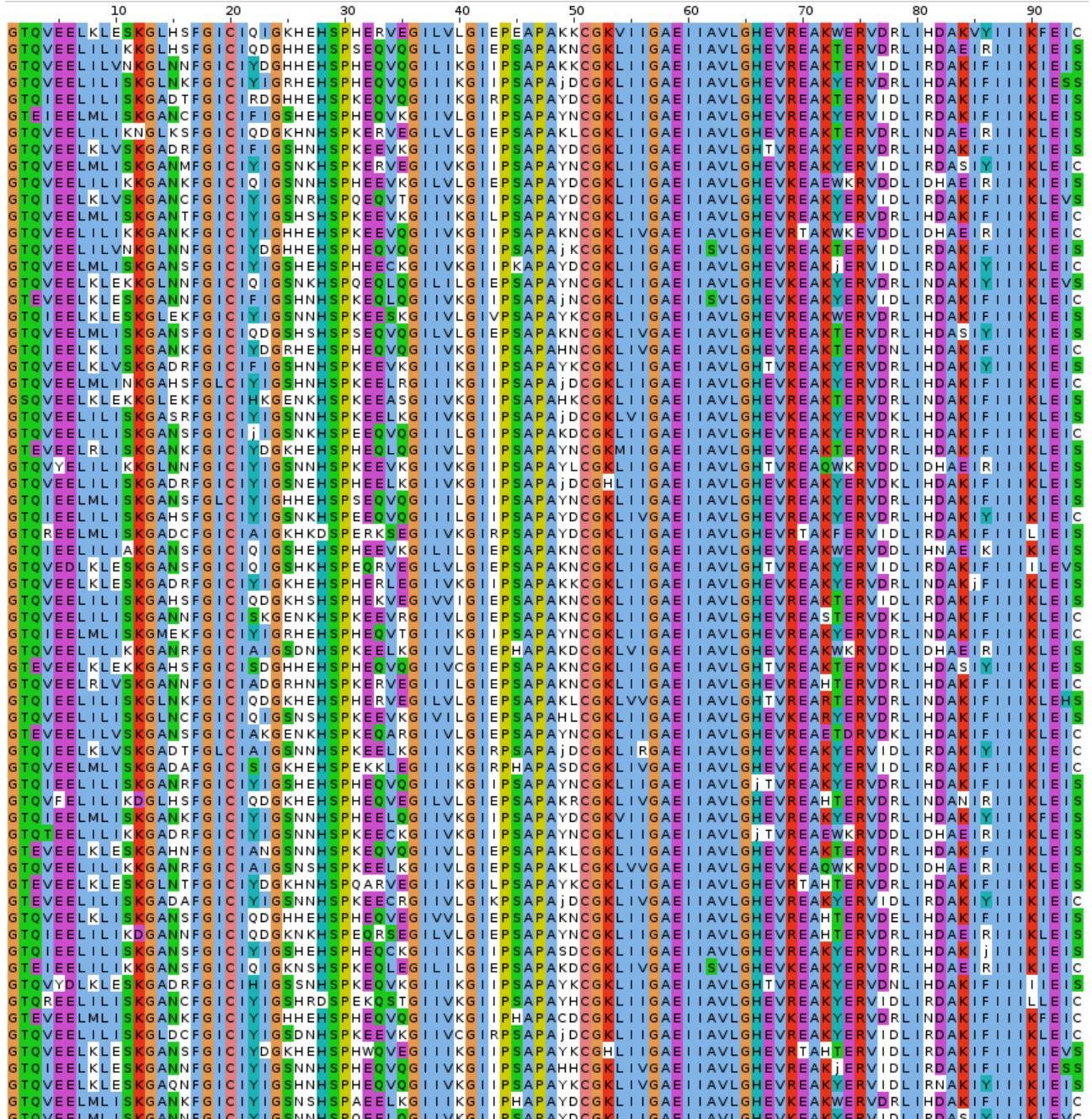


Figure 5.14 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine INAD (code PDB : 1IHJ), modèle NEA

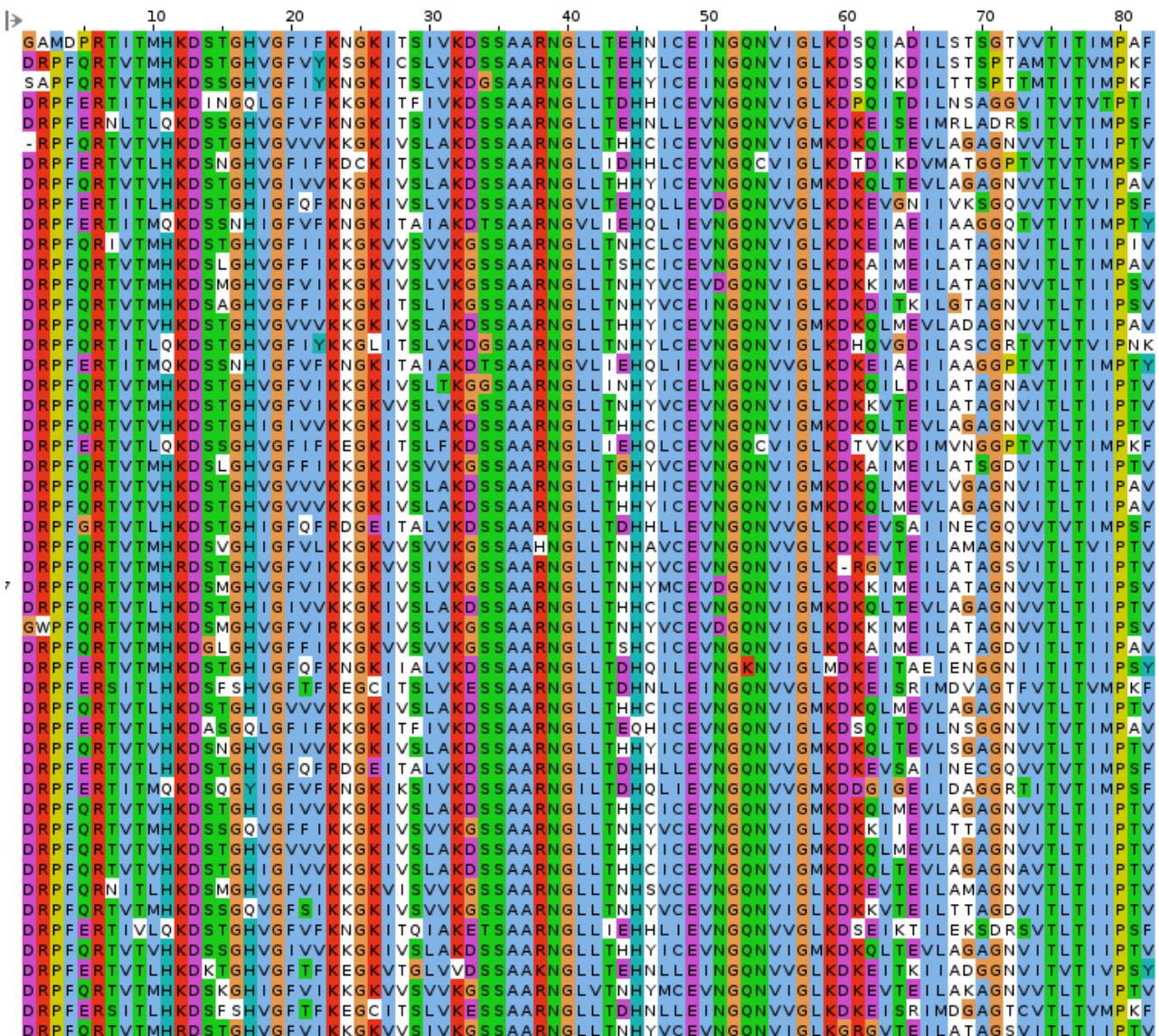


Figure 5.15 – L’alignement de notre sélection de séquences homologues à la protéine Syntenin (code PDB : 1R6J)

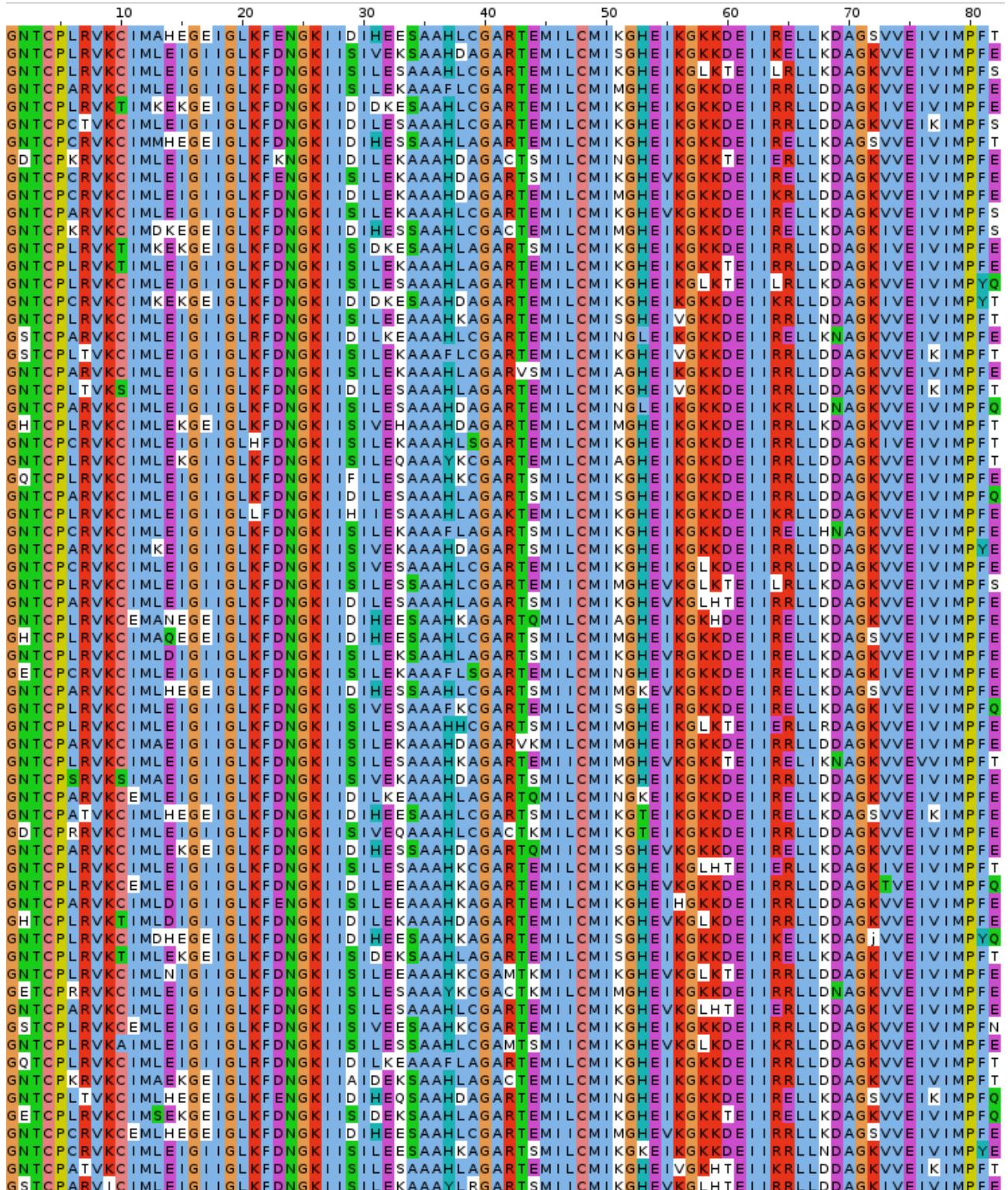


Figure 5.16 – Une sélection de séquences protéiques, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine Syntenin (code PDB : 1R6J), modèle NEA

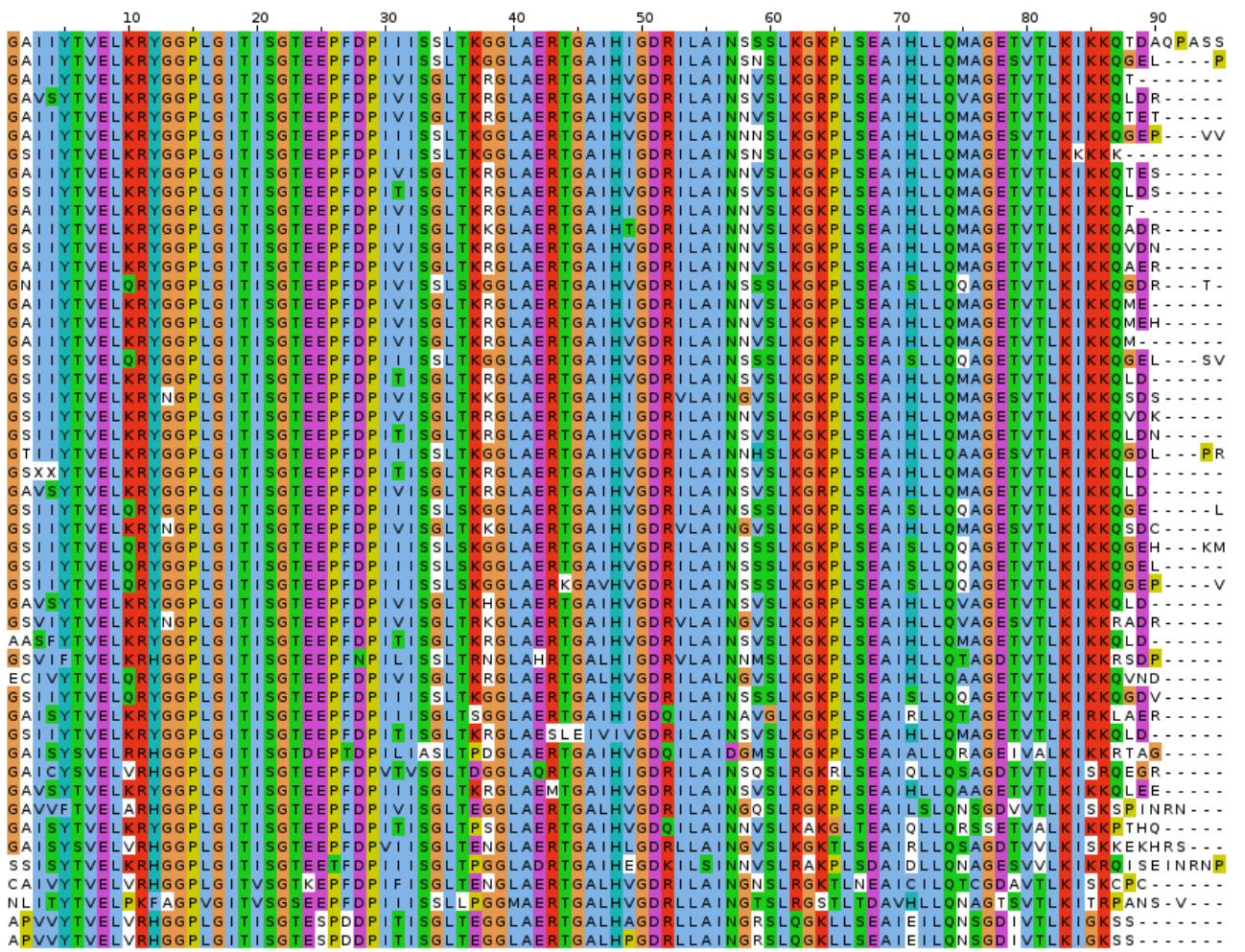


Figure 5.17 – L’alignement de notre sélection de séquences homologues à la protéine GRIP (code PDB : 1N7E)

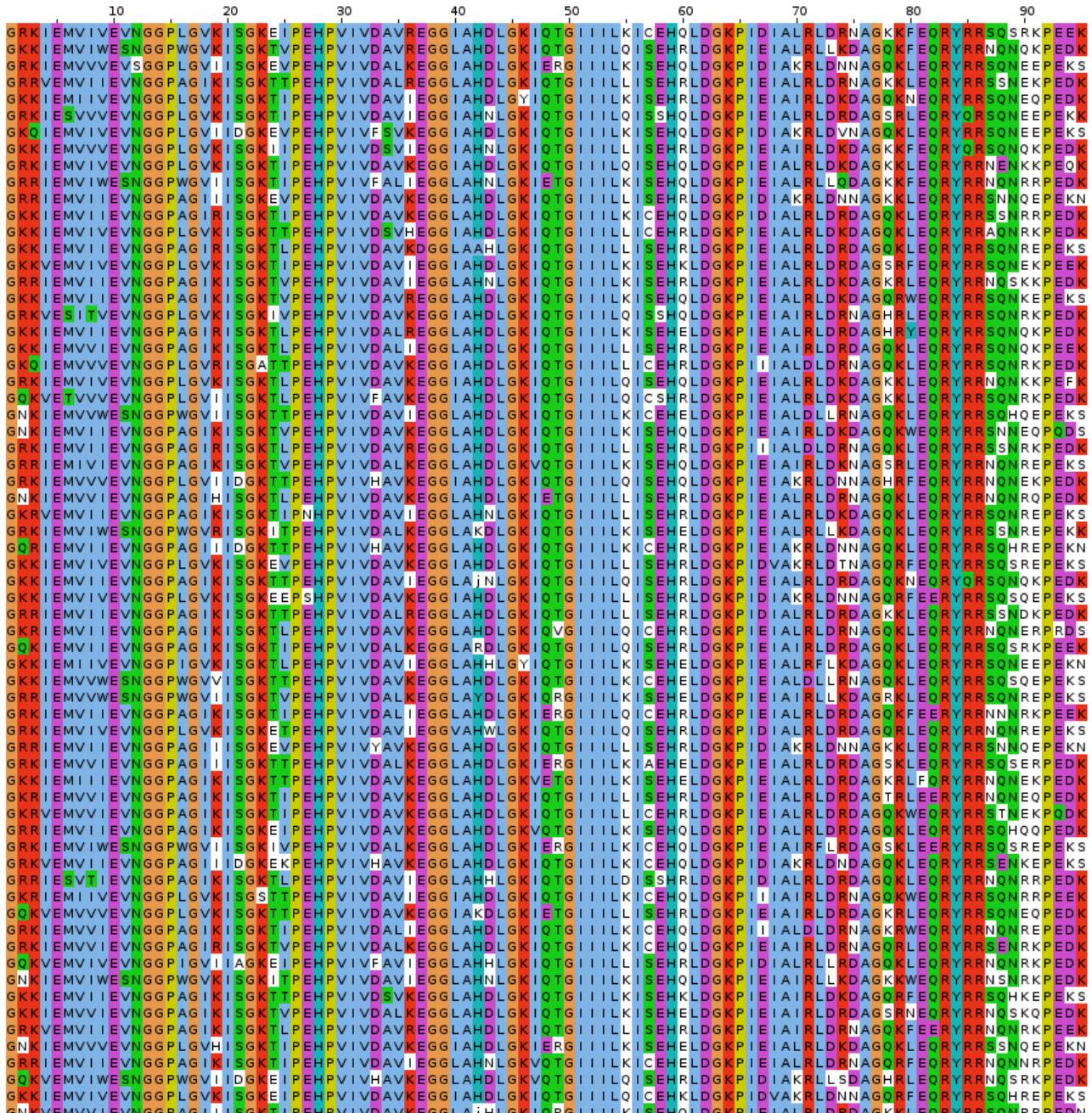


Figure 5.18 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine GRIP (code PDB : 1N7E), modèle NEA

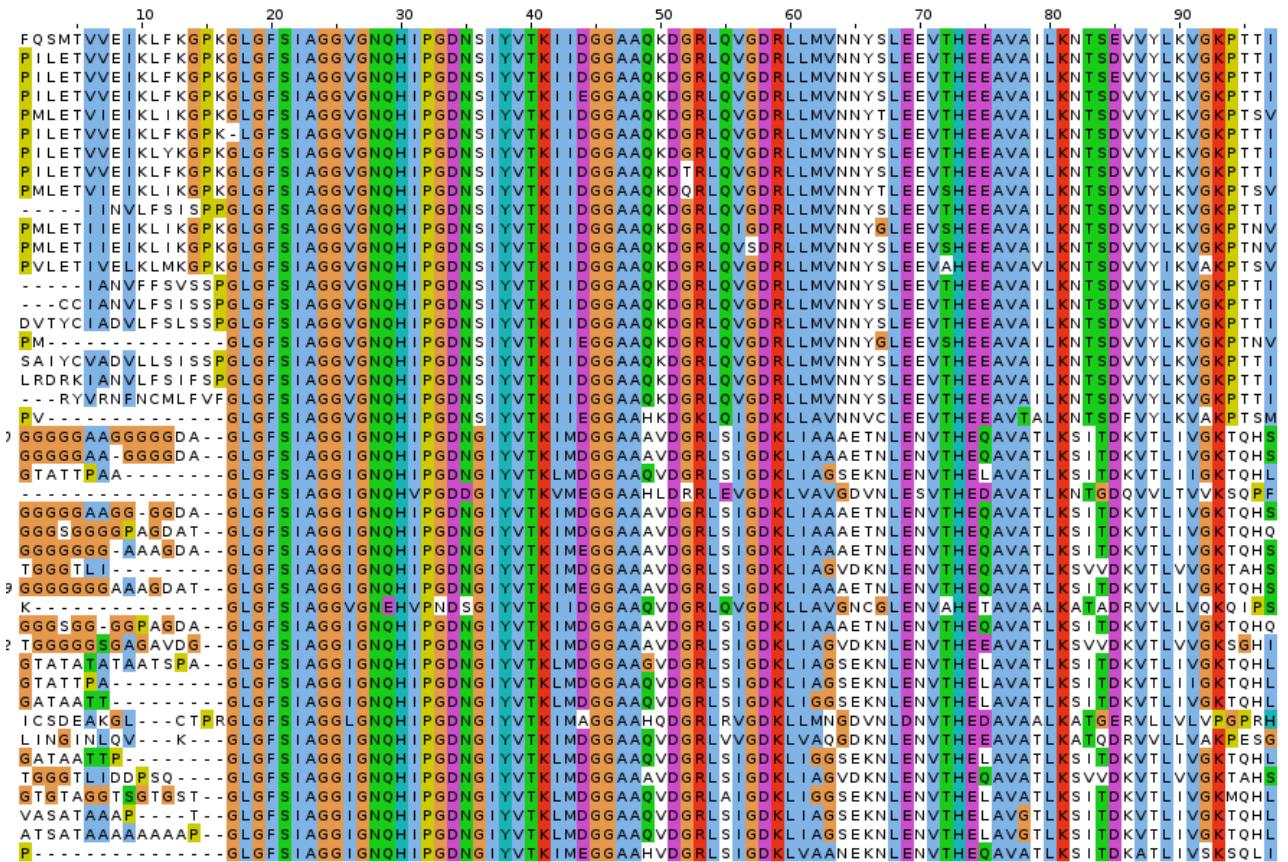


Figure 5.19 – L’alignement de notre sélection de séquences homologues à la protéine DLG2 (code PDB : 2BYG)

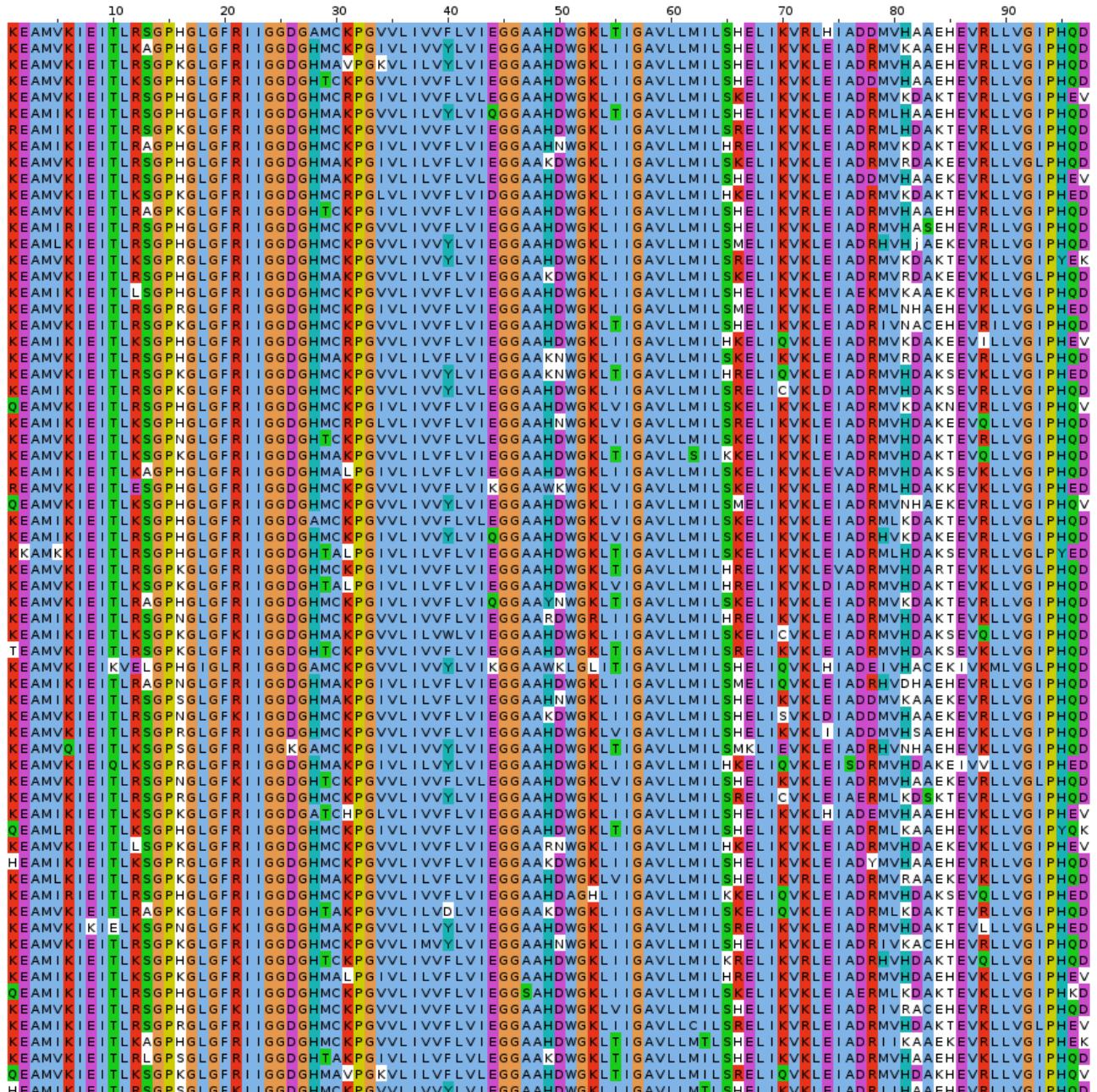


Figure 5.20 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine DLG2 (code PDB : 2BYG), modèle NEA

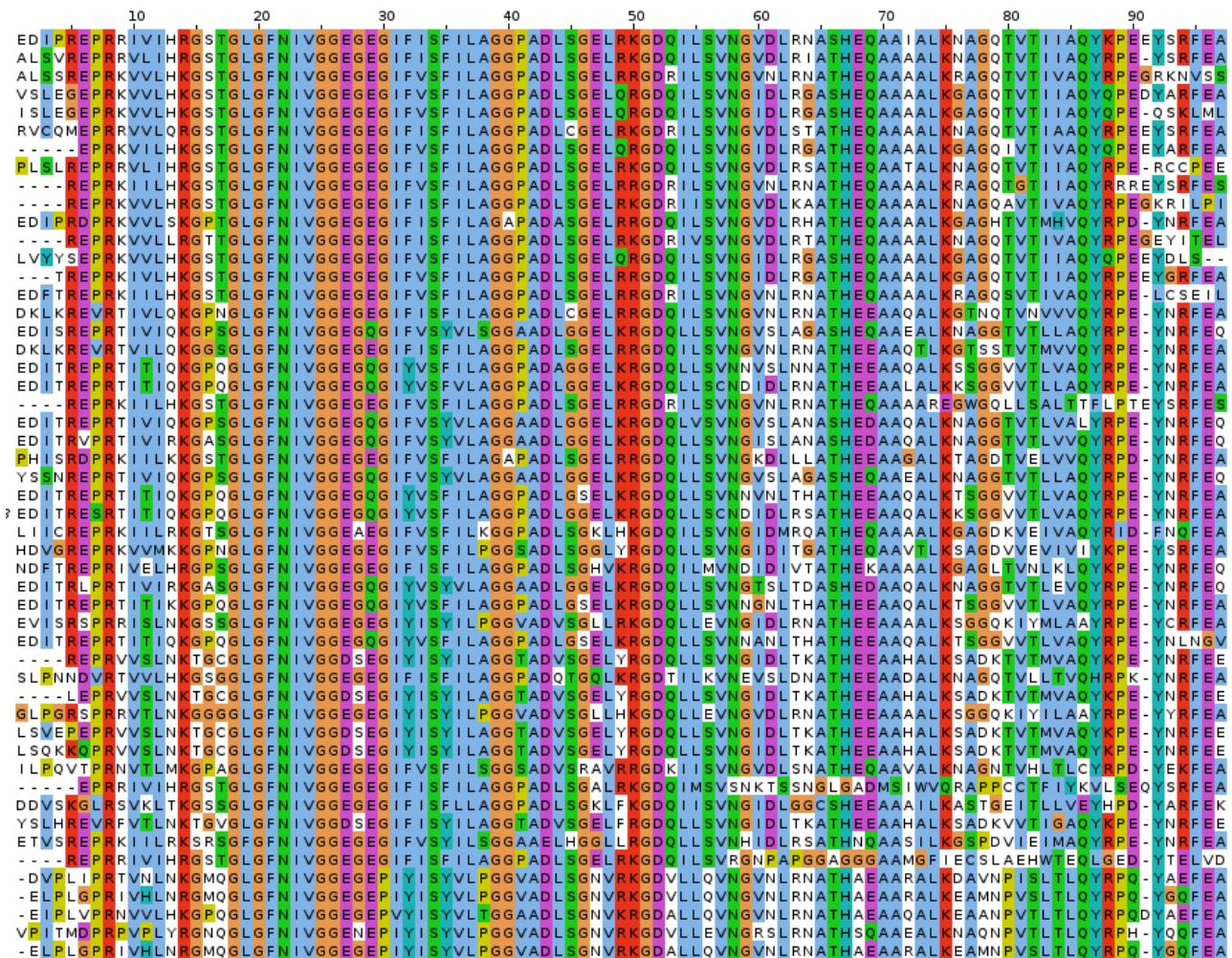


Figure 5.21 – L’alignement de notre sélection de séquences homologues à la protéine PSD95 (code PDB : 3K82)

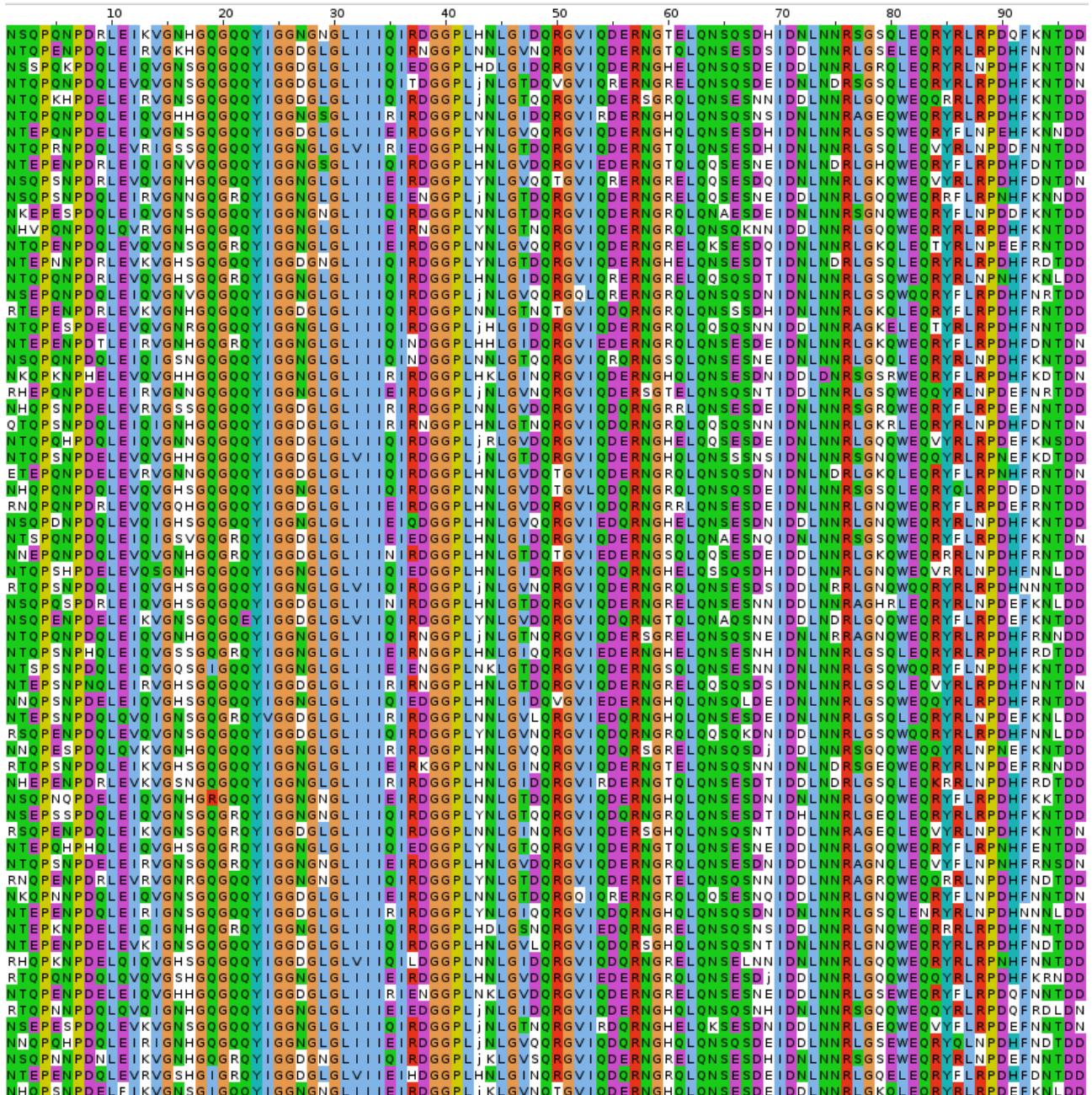


Figure 5.22 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine PSD95 (code PDB : 3K82), modèle NEA

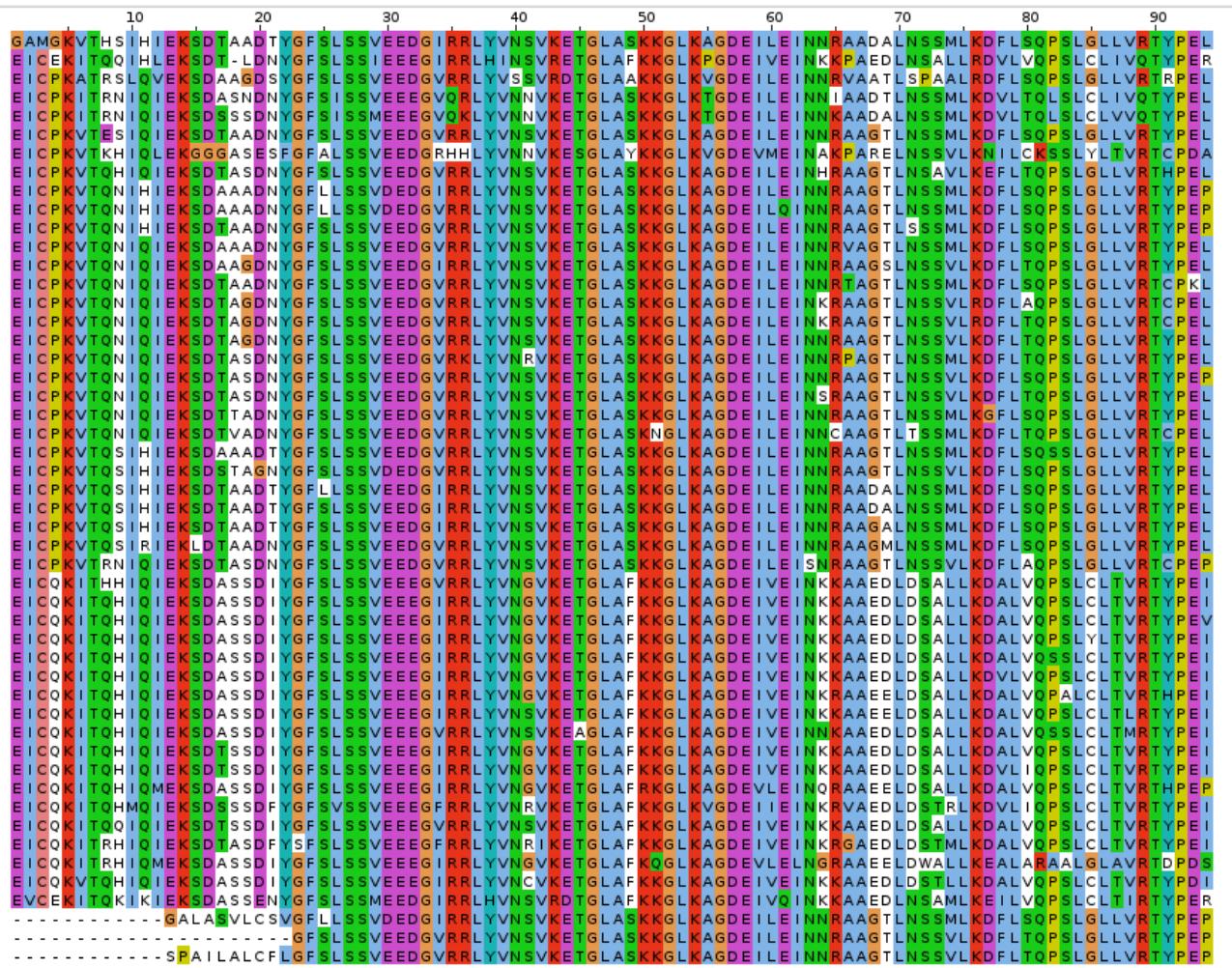


Figure 5.23 – L’alignement de notre sélection de séquences homologues à la protéine Tiam1 (code PDB)

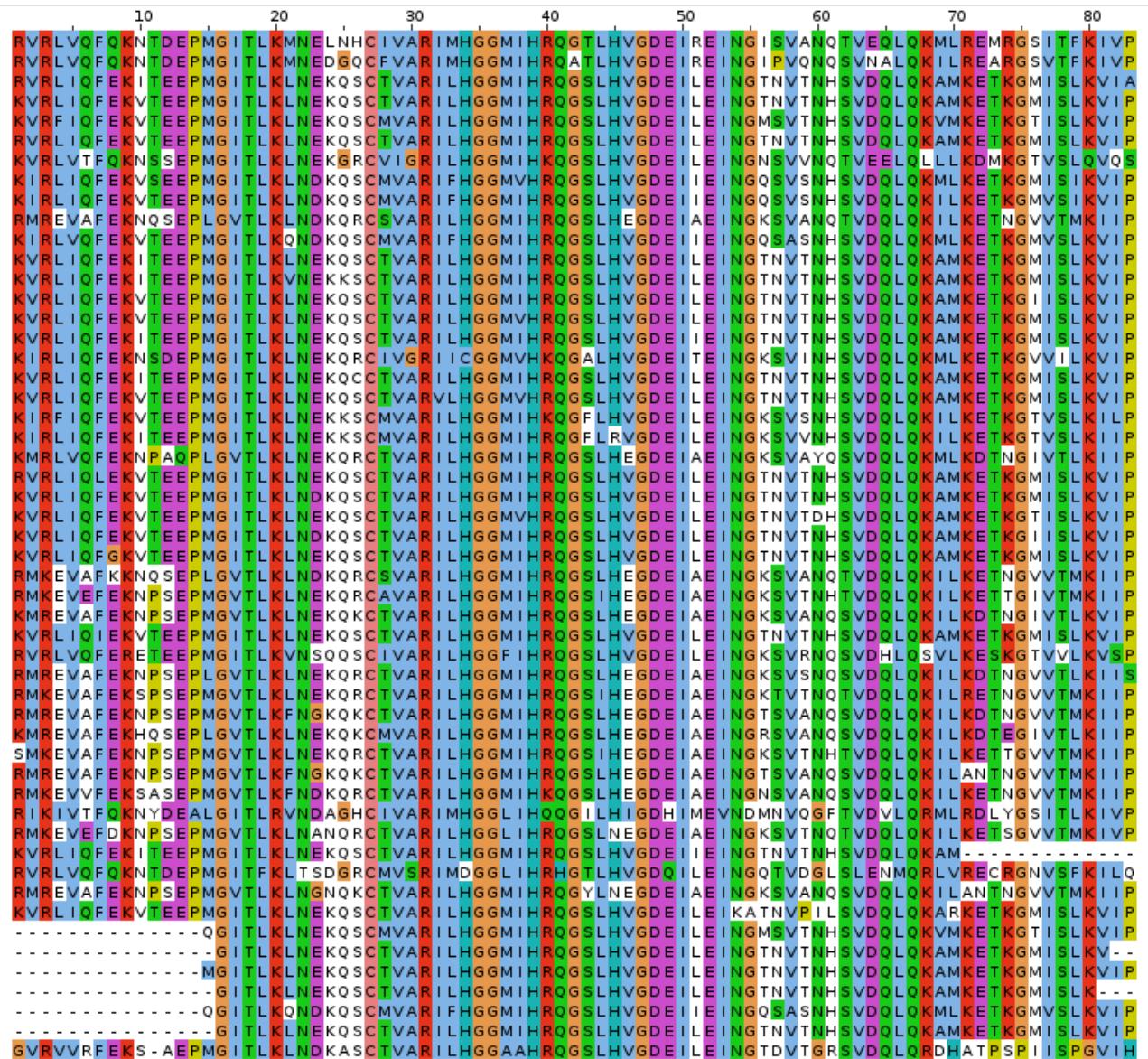


Figure 5.24 – L'alignement de notre sélection de séquences homologues à la protéine Cask (code PDB)

Publications

Simonson T., Gaillard T., Mignon D., Schmidt am Busch M., Lopes A., Amara N. and et al (2013). Computational protein design : the proteus software and selected applications. *J Comput Chem*, volume 34, page 2472–2484, doi : 10.1002/jcc.23418.

Polydorides S, Michael E, Mignon D, Druart K, Archontis G. and Simonson T. (2016). Proteus and the design of ligand binding sites. *Methods in Molecular Biology : Computational Design of Ligand Binding Proteins*, volume 1414, page 77–97, doi : 10.1007/978-1-4939-3569-7_6.

Mignon D. and Simonson T. (2016). Comparing three stochastic search algorithms for computational protein design : Monte carlo, replica exchange monte carlo, and a multistart, steepestdescent heuristic. *J Comput Chem*, volume 37, page 1781–1793, doi : 10.1002/jcc.24393

Mignon D., Panel N., Chen X., Fuentes E. and Simonson T. (2017). Computational design of the tiam1 pdz domain and its ligand binding. *J Chem Theory Comput*, volume 13, pages 2271–89, doi : 10.1021/acs.jctc.6b01255.

Villa V., Mignon D., Polydorides S. and Thomas Simonson T. (2017). Comparing pairwise-additive and many-body generalized born models for acid/base calculations and protein design. *J. Comp. Chem.*, volume 38(28), page 2396–2410, doi : 10.1002/jcc.24898.

