



École Doctorale INTERFACES
Approches interdisciplinaires: fondements, applications et
innovations

Titre de la thèse

Computational protein design: a tool for protein engineering and synthetic biology

présentée et soutenue publiquement le XXX

pour l'obtention du

Doctorat de l'Université Paris-Saclay

spécialité: Les sciences du vivant

par

M. David MIGNON

Composition du jury

Rapporteurs :	Dr. Prénom1 NOM1	Rapporteur externe
	Dr. Prénom2 NOM2	Rapporteur externe
	Pr. Prénom3 NOM3	Rapporteur interne

Examinateurs :	Dr. Prénom4 NOM4	Examinateur
	Dr. Prénom5 NOM5	Directeur de thèse
	Dr. Prénom6 NOM6	Directeur de thèse

Remerciements

Je remercie les membres du jury, Julien Bigot, Alain Denis, Sophie Barbe, Jean-François Gibrat, Yves-Henri Sanojouand pour avoir accepté de lire cette Thèse et d'évaluer mon travail. Je remercie particulierement Thomas Simonson, mon directeur, pour son encadrement de qualité, son énergie à aller de l'avant, son ouverture d'esprit et pour tout ce que ce travail m'a déjà apporté. Je remercie Yves Mechulam, mon directeur du laboratoire, d'avoir accepté que je réalise cet travail, et de m'avoir donner de très bonnes conditions pour le faire tout au long de ces nombreuses années. Ma thèse s'inscrit dans un projet bien plus large dans lequel beaucoup de personnes au déjà collaborées. J'ai bénéficié de leurs travaux. Et je remercie en particulier Thomas Gaillard, pour la qualité de la modélisation sans laquelle je n'aurai pas pu avancer, Nicolas Panel pour notre collaboration sur la famille PDZ et son expertise de Cask et Tiam1, Franchescho Villa pour ses tests proteus et le GB, Karen Druart

Le travail d'une thèse est le travail de trois longues années, qui n'aurait été possible sans l'aide et le soutien de mon entourage, tant professionnel que personnel. Je vais profiter de ces quelques lignes pour remercier ces personnes qui ont contribué de près ou de loin à cette partie de ma vie. Tout d'abord, je tiens à remercier les membres du jury, Anne-Claude Camproux, Juan Cortes, Annick Dejaegere et Yann Ponty, d'avoir accepté de lire ce manuscrit et d'évaluer mes travaux de thèse. Chacun de vos domaines ont permis de mettre en relief chaque facette de ma thèse. Je remercie également Yves Mechulam, directeur du Laboratoire de Biochimie, d'avoir accepté ma présence, parfois tardive, au sein du laboratoire et d'avoir mis à ma disposition les outils nécessaires pour réaliser au mieux ces recherches. Merci au CEA et à l'IDEX d'avoir financé cette thèse. Ensuite, je tiens à remercier Thomas Simonson qui a majoritairement encadré cette thèse. Je le remercie pour sa disponibilité, son accompagnement et la confiance qu'il m'a accordé au cours de ces travaux de recherche. Il a été présent à tout moment pour discuter des différents problèmes rencontrés et a été à l'écoute de mes propositions. Il m'a permis de nombreuses fois de prendre du recul sur mes travaux et d'élargir ma vision scientifique. Ceci a permis d'ajouter de nouvelles expériences, rendant l'ensemble de mes travaux cohérent. Aussi, je le remercie

sincèrement pour sa présence lors de la rédaction de ce manuscrit, d'avoir conservé mes idées à travers les nombreuses corrections qu'il a proposé. Il a ainsi rendu ce manuscrit plus fluide et plus agréable à lire. Je remercie mon second directeur de thèse Edouard Audit qui a permis cette collaboration entre la Maison de la Simulation et le Laboratoire de Biochimie. Cette collaboration a donné une dimension pluridisciplinaire à ma thèse. Je le remercie pour sa présence aux i réunions, où il a donné son point de vue critique sur chaque point de ma thèse. Ceci a remis en question certains de mes résultats et de faire naître de nouvelles hypothèses, étoffant ainsi mes travaux. Je remercie également Julien Bigot pour sa participation active à ma thèse. Il a été présent au cours de ces trois années, où il m'a appris à affiner mes résultats, à organiser une démarche scientifique et à aller au-delà de mes limites. Je le remercie pour son aide, notamment en informatique, ce qui m'a permis de découvrir le monde de la parallélisation, des multi-threads et des speed-up. Je le remercie aussi pour sa présence extra-professionnelle, surtout notamment les petits séjours en Ardèche avec nos collègues, qui m'ont aéré l'esprit chaque été ! Enfin, je voudrais remercier mes collaborateurs de l'IDRIS, Isabelle Dupays et Laurent Leger, qui ont mis en place de la parallélisation, ainsi que Matthieu Haefele de la MDLS qui a participé à la communication XPLOR/proteus. Je remercie mes collègues de mes deux laboratoires. Tout d'abord, je tiens à remercier tous mes collègues de BIOC et l'équipe de Bioinfo pour leur soutien. Plus particulièrement, je tiens à remercier Clara pour ses longs chemins sinueux pour aller manger, Pierre-Damien pour les soirées travaux/apéros/prêt-de-maison, Nicolas pour les soirées apéros/concerts-classiques et de m'avoir tourné le dos pendant 3 ans (et demi avec ton stage), Zoltan pour la découverte de la Dobos torta et de tes talents de cuisinier, Claire pour ta joie de vivre, Savvas pour la maturité scientifique et personnelle que tu as pu m'apporter, Mélanie pour ta disponibilité dans ton bureau et les multiples bavardages girly, Thomas G. pour toutes les discussions scientifiques qu'on a pu aborder, David pour les remplacements de disques durs brûlés et les discussions Proteus, Titine pour ta bonne humeur et la découverte de jeux vidéos, Michou pour ta gentillesse, Pierre pour ta disponibilité lorsque j'avais des questions sur les analyses expérimentales, Michel pour tes blagues très drôles (la dernière en date : ribosome/ribos-“femme”), Marc D. pour nos discussions arts et techniques, ainsi que Sylvain, Francesco, Mimi, Marc G., Emma, Alexey, les Pascaux, Guillaume, Catherine, Cédric, Gaby, Clément, Aurianne, Etienne, Régis, Giuliano, ii Jérôme, Nhan, Amlam, Ditipriya, qui ont chacun apporté une touche personnelle dans la vie quotidienne du laboratoire. Je remercie également mes collègues de la MDLS, que j'ai pu côtoyer, avec qui j'ai partagé des moments importants de la thèse (sacrées Journées des Thèses), des soirées, ou bien juste un café : Julien B., Ralitsa, Maxime, Pascal, Seb, Thibault, Valérie, Frédéric, Julien D., Florence, Mohamed,

Pierre-Elliott, Philippe, Lu, Thomas, Matthieu, Adeline, Daniel, Michel, Pierre, Samuel, Martial, Mathieu, Pascal, Fabien, Rehan, Esra, Giorgio. Plus personnellement, je voudrais remercier Gautier Moroy, qui a été mon responsable de stage de M1, puis mon tuteur de thèse. Je le remercie pour son soutien, et sa disponibilité pour discuter des différents problèmes rencontrés, de m'avoir aidé à tenir bon lors de l'été 2012. Je voudrais remercier mes acolytes de l'association 2AEM-ISDD : Véronique, Mélaine, Grace et Answald. Merci pour cette expérience sans fin... Je voudrais remercier mes amis qui m'ont soutenu ces dernières années : Mélaine (encore !) pour le soutien intense que tu m'as apporté, ta joie, ta bonne humeur, pour le partage de la place "major de promo", de m'avoir initié aux joies du festival (à dormir dans une tente glacée), et pour ces quelques jours de vacances à la plage ; Inès pour ton intégrité et ton don de savoir rassurer et apaiser les gens ; Alexandre pour tes précieux conseils scientifiques et associatifs ; Julien C. pour le partage d'expérience (alors cette thèse ?) ; Laura, Lionel et Nicolas C. pour votre bonne humeur et ces voyages à travers la France ! Enfin, je voudrais remercier ma famille, qui sans eux, je ne serais pas arrivée jusque là. Tout d'abord, je voudrais remercier les membres de ma famille qui sont loin mais tout de même présents : Simone et Jean-Claude, Monique, Myriam et Laurent. Merci pour tout ! Mes frères et soeurs : Sandy, Olivia, Nicolas et Louise, qui ont suivi les péripéties à distance mais qui ne s'en sont pas moins senti impliqués. Un petit plus pour ma petite soeur qui découvre au lycée l'ADN, la transcription et la traduction des protéines. Elle a pu saisir malgré son jeune âge l'intérêt de ma thèse et nous avons pu discuter science plusieurs heures. C'est un peu grâce à elle que j'ai appris à vulgariser mes travaux. Je remercie énormément mon papa qui m'a soutenu, encouragé et conseillé toutes ces années d'études et qui m'a donné la force d'aller aussi loin. Et pour finir, je voudrais remercier mon iii compagnon David, qui m'a supporté ces derniers mois. Les gens qui me connaissent savent que ce fut une tâche ardue. Mais il y est parvenu ! Et je le remercie pour sa compréhension sur mes horaires tardives, mon implication dans la rédaction de ce mémoire et sur la préparation de ma soutenance. Et merci d'avoir subit mes répétitions de soutenance. Merci à tous... iv

à Alice et Elliott.

Table des matières

Liste des figures	xii
Liste des tables	xvii
Abreviations	xix
1 Le « CPD » : Conception de protéine par ordinateur	1
1.1 L'espace des séquences-conformations	3
1.1.1 L'état replié	3
Les chaînes latérales	4
Le squelette	4
1.1.2 L'état déplié	5
1.2 L'énergie d'une protéine	5
1.2.1 La mécanique moléculaire	5
Les interactions liées	6
Les interactions non liées	7
1.2.2 D'autres approches	8
1.3 La modélisation du solvant	8
1.3.1 Le modèle de solvant explicite	9
1.3.2 Le modèle implicite	10
1.3.3 Le modèle CASA	11
1.3.4 Le modèle Poisson-Boltzmann	12
1.3.5 Le modèle de Born Généralisé	13
« Native Environment Approximation » (NEA)	15
1.4 L'algorithme d'exploration	15
1.4.1 L'algorithme du champ moyen	16
1.4.2 Le Dead-End Elimination	17
1.4.3 Le CFN	19

Table des matières

L'algorithme « Depth-First Branch and Bound » (DFBB)	19
Equivalence Preserving Transformation	21
1.4.4 L'heuristique Multistart Steepest Descent (MSD)	21
1.4.5 L'algorithme génétique	23
1.4.6 Le Monte Carlo	23
Le principe de la balance détaillée	25
L'algorithme de Metropolis	26
1.4.7 Le Monte Carlo avec échanges de répliques (REMC)	28
2 Proteus et nos outils d'analyse	31
2.1 Un modèle fondé sur la Physique	31
2.2 Organisation générale	33
2.3 Décomposition par paires de la fonction d'énergie	33
2.3.1 Décomposition du terme de surface	34
2.3.2 « Native Environnement Approximation » (NEA)	35
2.3.3 « Fluctuating Dielectric Boundary » (FDB)	35
2.4 La matrice d'énergie	36
2.4.1 Les énergies de l'état déplié	36
2.4.2 Déroulement de la construction de la matrice	37
2.4.3 Les fichiers d'énergies	39
2.5 Configuration de l'exploration	39
2.5.1 Les déplacements Monte Carlo	41
2.5.2 Restriction de l'espace séquence-conformation	43
2.5.3 Définition de la fonction de score	43
Définition de groupes	44
Déclaration de la fonction de score	45
2.6 Les fichiers de sortie	46
2.7 Outils d'analyse de séquences	47
2.7.1 Superfamily/SCOP	47
2.7.2 Taux d'identité de séquences	48
2.7.3 Taux d'identité par position	48
2.7.4 Alignements Pfam	48
2.7.5 Score BLOSUM	49
2.7.6 Similarité d'un ensemble à un alignement Pfam	49
2.7.7 Entropie par position	49

3 Développements	55
3.1 Les Modèles	55
3.2 OpenMP pour le REMC	57
3.2.1 Présentation d'OpenMP	57
3.2.2 REMC dans proteus	58
3.3 Amélioration de la fonction d'acceptation MC	59
3.4 Seuil d'impression	59
3.5 Nouveau système de déplacements	61
3.6 Label	62
3.7 Allocation variable de la matrice	62
4 Comparing three stochastic search algorithms for CPD	63
4.1 Introduction	64
4.2 Methods	66
4.2.1 Monte Carlo : general framework	66
4.2.2 MC and REMC : implementation details	69
4.2.3 Heuristic sequence optimization	70
4.2.4 Cost function network method	70
4.2.5 Energy function	71
4.2.6 Test systems and preparation	71
4.2.7 Sequence characterization	72
4.3 Results	73
4.3.1 Quality of the designed sequences	74
4.3.2 Finding the GMEC	77
CPU and memory limits for each method	77
Optimal sequences/structures with up to 10 designed positions . . .	79
Optimal sequences with 20 or 30 designed positions	82
4.3.3 Density of states above the GMEC	83
4.4 Concluding Discussion	87
5 Application aux domaines PDZ	99
5.1 Introduction	99
5.2 Le modèle d'état déplié	100
5.2.1 Les énergies de référence	100
5.2.2 La vraisemblance des énergies de référence	101
5.2.3 Recherche du maximum de vraisemblance	102
5.3 Méthodes de calcul	103

Table des matières

5.3.1	Énergie de l'état replié	103
5.3.2	Les énergies de référence de l'état déplié	104
5.4	Séquences expérimentales et modèles structuraux	106
5.4.1	L'ensemble des protéines PDZ	106
5.4.2	Alignements Blast croisés	106
5.4.3	Sélection des homologues	106
5.4.4	Alignements des protéines expérimentales et leurs homologues	108
5.4.5	Similarité des homologues	108
5.4.6	Les fréquences d'acides aminés	110
5.5	Séquences calculées	125
5.5.1	Préparation	125
5.5.2	Simulation Monte Carlo	125
5.5.3	Génération de séquence Rosetta	125
5.5.4	Caractérisation des séquences calculées	126
5.6	Résultats du modèle NEA	126
5.6.1	Optimisation du modèle de l'état déplié	126
5.6.2	Tests de reconnaissance de famille	129
5.6.3	Séquences et diversité de séquences	129
5.6.4	Scores de similarité Blosum	130
5.6.5	Tests de validation croisée	131
5.7	Résultats du modèle FDB	133
5.7.1	Optimisation du modèle de l'état déplié	133
5.7.2	Tests de reconnaissance de famille	134
5.7.3	Scores de similarité Blosum	136
5.7.4	Taux d'identité à la séquence native	137
5.7.5	Logos des séquences obtenues	138
5.8	Application : Croissance du noyau hydrophobe	140
5.9	Conclusion	163
5.9.1	Modèle mis en œuvre	163
5.9.2	Tests et application	165

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 à un ligand.	2
1.2	Représentation des énergies liées De la gauche vers la droite et du haut vers le bas, sont représentés les énergies de liaison, d'angle, du dièdre et du dièdre impropre.	7
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.	11
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.	12
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.	14

Liste des figures

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.	18
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	20
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.	22
1.9	L'algorithme génétique	24
2.1	Cycle thermodynamique qui définit la stabilité relative de deux séquences-conformations S_A et S_B	32
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 choix la zone noire.	34
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.	37

2.4	les fichiers d'énergies en mode CASA	40
2.5	les fichiers d'énergies en mode GB/NEA et FDB	41
2.6	Cycle thermodynamique qui définit l'affinité.	43
2.7	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.	45
2.8	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).	46
2.9	Les fichiers en sortie de proteus en haut pour le mode MONTECARLO, en bas pour le mode POSTPROCESS	47
2.10	Les principales structures « physiques » dans proteus	52
2.11	Les principales structures « logiques » dans proteus	53
2.12	Les principales structures « dynamiques » dans proteus	54
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.	56
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.	58
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	60
4.1	width=1cm	75
4.2	width=1cm	76
4.3	width=1cm	77

Liste des figures

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.	79
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.	80
4.6	width=1cm	86
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.	109
5.2	L’alignement de notre sélection de séquences homologues à la protéine NHREF (code PDB : 1G9O)	111
5.3	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	112
5.4	L’alignement de notre sélection de séquences homologues à la protéine INAD (code PDB : 1IHJ)	113
5.5	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	114
5.6	L’alignement de notre sélection de séquences homologues à la protéine Syntenin (code PDB : 1R6J)	115
5.7	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	116

5.8 L'alignement de notre sélection de séquences homologues à la protéine GRIP (code PDB : 1N7E)	117
5.9 Une sélection de séquences proteus, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine GRIP (code PDB : 1N7E), modèle NEA	118
5.10 L'alignement de notre sélection de séquences homologues à la protéine DLG2 (code PDB : 2BYG)	119
5.11 Une sélection de séquences proteus, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine DLG2 (code PDB : 2BYG), modèle NEA	120
5.12 L'alignement de notre sélection de séquences homologues à la protéine PSD95 (code PDB : 3K82)	121
5.13 Une sélection de séquences proteus, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine PSD95 (code PDB : 3K82), modèle NEA	122
5.14 L'alignement de notre sélection de séquences homologues à la protéine Tiam1 (code PDB)	123
5.15 L'alignement de notre sélection de séquences homologues à la protéine Cask (code PDB)	124
5.16 Similarité des séquences des 6 protéines produites par Proteus et Rosetta à l'alignement Pfam RP55, sur l'ensemble des positions.	132
5.17 Similarité des séquences des 6 protéines produites par proteus modèle NEA et Rosetta à l'alignement Pfam RP55, sur les positions du cœur hydrophobe.	133
5.18 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.	138
5.19 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	139
5.20 Les séquences conçues par Proteus, les homologues à la native et les séquences naturelles représentées sous forme de logo pour les positions exposées	140
5.21 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.	141

Liste des figures

5.22 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.	142
5.23 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.	163
5.24 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.	163

Liste des tables

2.1	une partie des balises possibles du fichier de configuration de proteus	42
4.1	Selected MC and REMC protocols	70
4.2	Test proteins	72
4.3	Designed sequence quality measures	78
4.4	Tests with 10 designed positions	82
4.5	Tests with 20 and 30 designed positions	84
4.6	Designed and Pfam sequence entropies	85
4.7	Test proteins	88
4.8	Designed sequence quality measures	89
4.9	Tests with 10 designed positions	90
4.10	Tests with 20 and 30 designed positions	91
4.11	Designed and Pfam sequence entropies	92
4.12	La sélection des positions pour tests 5-actives	95
4.13	La sélection des positions pour tests 10-actives	96
4.14	La sélection des positions pour tests 20-actives	97
4.15	La sélection des positions pour tests 30-actives	98
5.1	Les groupes d'acides aminés utilisés pour l'optimisation des énergies de référence.	105
5.2	La sélection de domaines protéiques PDZ	106
5.3	E-value et pourcentage d'identité des alignements Blast native versus native pour nos séquences PDZ	107
5.4	Sélection des homologues.	107
5.5	Similarité des séquences expérimentales homologues des huit protéines PDZ.	108
5.6	Les énergies de référence (kcal/mol) optimisée sur six protéines.	127

Liste des tables

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.	128
5.8	Résultats Superfamily pour les séquences Proteus avec le modèle NEA. . .	129
5.9	Résultats Superfamily pour les séquences Rosetta	130
5.10	Moyenne de l'exponentielle de l'entropie sur les ensembles des positions des protéines	131
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.	135
5.12	Les énergies de référence obtenues avec l'optimisation sur 3 protéines. La constante diélectrique est fixée à 4.	136
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).	137
5.14	Pourcentage d'identité moyen à la séquence native	137

Abbreviations

3D tri-dimensionnel(le)	NEA Naive Environnement Aproximation
AMBER Assited Model Building and Energy Refinement	PB Poisson-Boltzmann
BLOSUM BLocks of amino acid SUbstitution Matrix	REMC Replica Exchange Monte-Carlo
CASA Coulomb Accessible Surface Area	GMEC "Global minimal energie cost"
CHARMM Chemistry at HARvard Macromolecular Mechanics	Pfam "Protein family databank"
DEE Dead-End Elimination	backbone (ou squelette) chaîne principale de la protéine
CPD Computational Protein Design	$\frac{1}{RT}$ avec R la constante des gaz parfait et T la température
GB Born Généralisé	DFBB Depth-First Branch and Bound
FDB Fluctuating DIelectric Boundary	EPT Equivalence Preserving Transformation
H algorithme heuristique	MSD Multistart Steepest Descent heuristic
MC algorithme Monte-Carlo	

Chapitre 1

Le « CPD » : Conception de protéine par ordinateur

L'ingénierie des protéines est l'ensemble des techniques pour modifier la fonction, ou la structure 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.

Dans ce domaine existe la mutagenèse dirigée dans laquelle la première étape est l'identification des mutations intéressantes pour l'objectif fixé, 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 une séquence de protéine d'intérêt et toutes les séquences ainsi produites sont testées afin de trouver la caractéristique 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 qui exploite la capacité de calculs des ordinateurs 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 être capable de représenter une ou un petit nombre de conformations de la chaîne principale du polypeptide et un ensemble de positionnements de la chaîne latérale de chacun des résidus possibles.

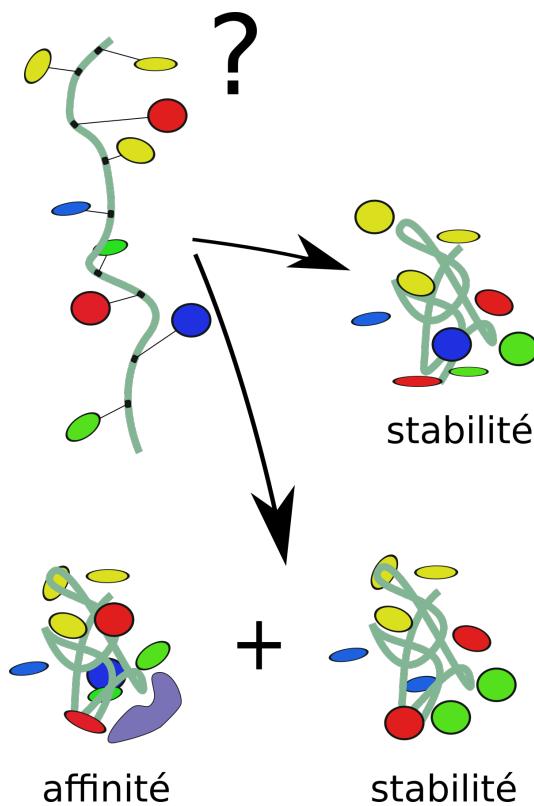


Figure 1.1 – Le CPD pour « Computational Protein Design » recherche les séquences d’acides aminés compatibles avec une protéine dont la structure 3D est connue, c’est-à-dire compatibles avec un repliement et éventuellement une affinité à un ligand.

2. Une fonction d’énergie

Elle permet d’évaluer la pertinence des conformations.

3. Un algorithme d’exploration de l’espace de conformation

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 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 des d'états d'une chaîne polypeptidique, dans le cadre de la mécanique moléculaire est alors constitué d'un vaste espace continu de conformation possibles.

D'autre part, si un domaine polypeptidique a un nombre N de résidus compris entre 50 et 100, et que chacune des N positions de la chaîne peut adopter l'un des 20 types d'acides aminés, le nombre de polypeptides à considérer est égale à N^{20} . 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 [1] 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.

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 [2]. Présentons maintenant, la modélisation des chaînes latérales et celle du backbone, c'est-à-dire le squelette polypeptidique.

Les chaînes latérales

Finkelstein et Ptitsyn [3], Janin et al. [4] ainsi que Ponder et Rischards [1] 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 [5, 6].

Le squelette

Partant du fait que les positionnements des chaînes latérales n'influencent que faiblement la structure adoptée par le backbone, la chaîne principale de la protéine est fixée 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 avec le backbone, ces deux types sont souvent traités à part. Cette approche a obtenu de nombreux succès, voir par exemple [7].

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 [8]. En réponse à ce problème, Druart et al. [9] 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 [8]. Dantas et al. [10] font des simulations avec minimisation après chaque mouvement de chaîne latérale. Kuhlman et al. [11] 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 appelé « backrub ». Ce sont des mouvements naturels du backbone mis en évidence par David et al. [12], 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 [13, 14] 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 de X dans l'état déplié [15]. 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 ou fonction de score 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 [16]
- CHARMM : Chemistry at HARvard Molecular Mechanics [17]

- OPLS : Optimized Potential for Liquid Simulations [18]
- GROMOS : GRONingen MOlecular Simulation [19]

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 de déformations 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.

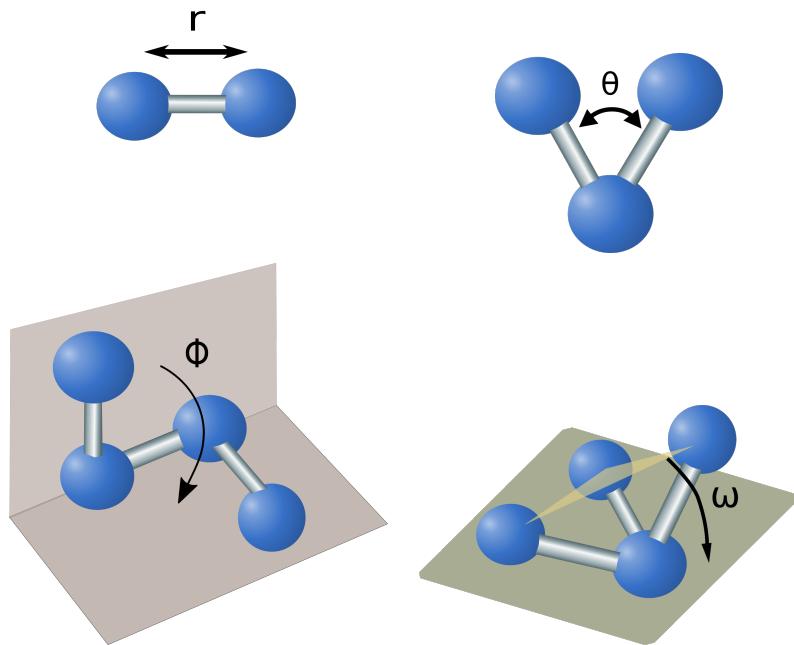


Figure 1.2 – **Représentation des énergies liées** De la gauche vers la droite et du haut vers le bas, sont représentés les énergies de liaison, d'angle, du dièdre et du dièdre impropre.

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, Delye, 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 hydrogènes avec une fonction d'énergie de Coulomb qui contient un ϵ qui varie en fonction de la distance interatomique [20]. Il existe des fonctions d'énergie comportant des éléments relevant de statistiques sur les protéines [21]. 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 [22].

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 hydrogènes entre molécules d'eau et avec tous autres groupes polaires. Ainsi, l'eau à l'état liquide possède beaucoup de liaisons hydrogènes.

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 ou pour les 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, c'est-à-dire, l'espace à $6N$ dimensions pour une simulation du solvant avec N molécules d'eau telle que chaque élément de cet espace décrive une configuration des positions et des vitesses dans l'espace 3D de la collection de molécules. 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).

Mais, 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 tenter 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 pour le solvant, 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. La surface accessible au solvant 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} .

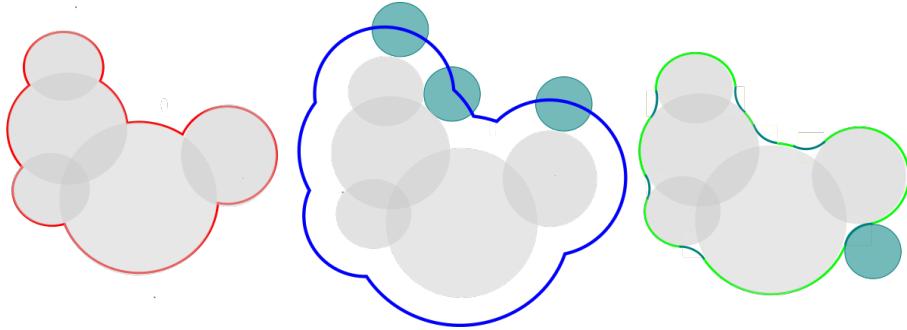


Figure 1.3 – Les trois types classiques de surfaces pour une molécule À gauche et en rouge la surface de Van der Waals (SVdW), au centre et en bleu, la surface accessible au solvant (SA), à droite, et en vert la surface de Connolly ou surface moléculaire (SM). La surface SA est l'ensemble des positions 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.

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

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

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 Eisinger [23] fait de lui un modèle fréquemment utilisé. Cependant, il est en difficulté pour le traitement de la surface des protéines.

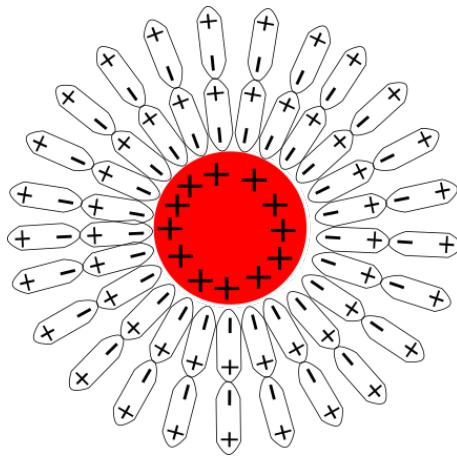


Figure 1.4 – Représentation schématique de l’organisation des dipôles des molécules d’eau autour d’un soluté sphérique chargé positivement à sa surface. Les dipôles des premières couches de solvatation s’orientent suivant les lignes du champ électrique produit par le soluté.

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. Par contre, les charges de la protéine sont traitées de façon explicite. Ainsi ce modèle prend en compte l’écrantage du 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 pour l’eau.

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 :

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

1.3. La modélisation du solvant

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 [24] et APBS [25] 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 [26] 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. [27] 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 :

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

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 :

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

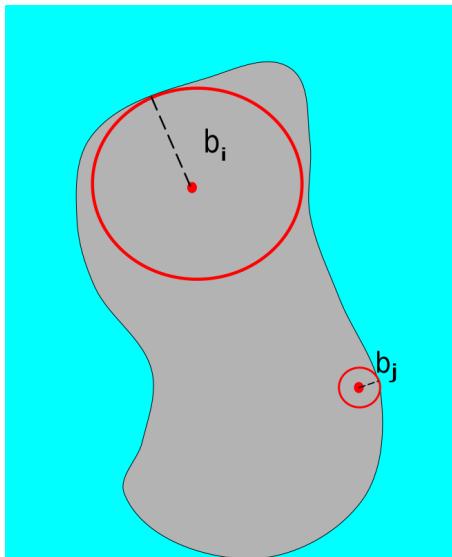


Figure 1.5 – Des rayons de Born de deux atomes dans une protéine Le rayon de Born pour l'atome enfui est environ le rayon de la protéine, alors que le rayon de Born d'un atome à la surface est environ son rayon de Van der Waals.

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 postions 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 [28–30]. 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 dépendant 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'approche 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 discréétisés, par exemple [31] 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. A chaque position, les rotamères sont équiprobales.
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 [32] :

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 $\min_{r_i \in C} E(C) > \max_{r'_i \in C} E(C)$ 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 $E_i(r_i) + \sum_{j \neq i} \min_{r_j} E_{ij}(r_i, r_j) > E_i(r'_i) + \sum_{j \neq i}^N \max_{r_j} E_{ij}(r'_i, r_j)$ 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.

S'il existe (r_i, r_j) tel que

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

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 [33] : 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 critère peut s'exprimer de la façon suivante :

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

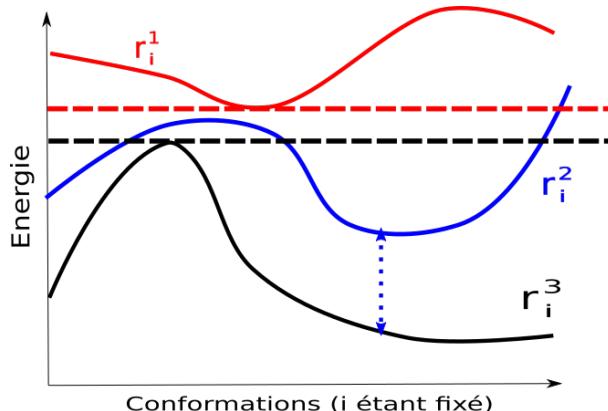


Figure 1.6 – **DEE, graphes de la fonction d'énergie pour des rotamères fixés à la position i** le critère simple permet l'élimination du rotamère r_i^1 parce que l'énergie des conformations qui le contiennent, l'énergie rouge, est toujours moins bonne que la noire. Mais seul le critère de Goldstein 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 est plus de rotamère 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 [34]. Beaucoup de variantes du DEE [35, 36] ont été proposées notamment en travaillant sur des combinaisons de plus de deux rotamères.

1.4.3 Le CFN

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 position 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 certain critère, 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. 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.

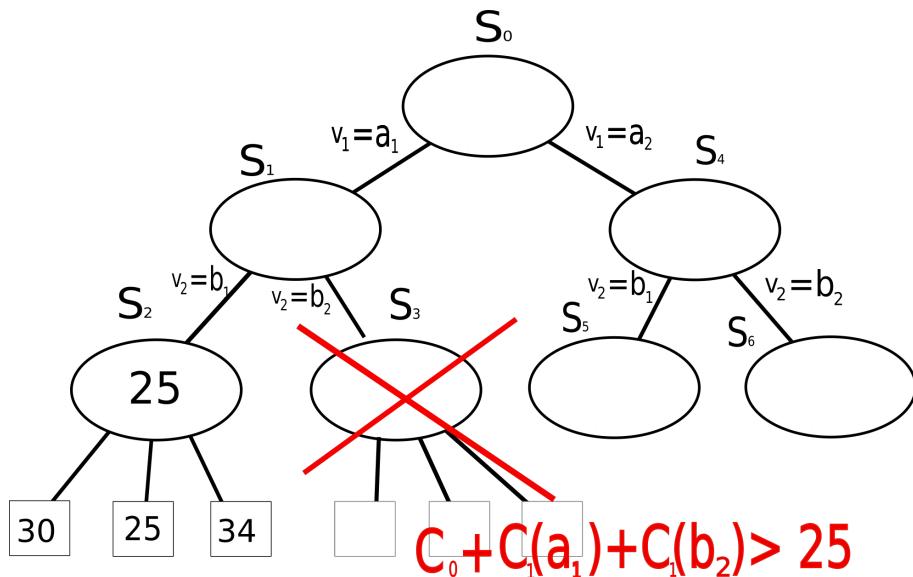


Figure 1.7 – **Principe de l'algorithme du Depth-First Branch and Bound** Les solutions d'un CFN sont représentées sous forme d'un arbre dans lequel le sommet S_0 représente l'ensemble des solutions et les fils S_1 et S_4 une partition de S_0 , avec pour chacun une première variable v_1 fixée. Le DFBB, 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 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 de les attribuer aux fonctions qui portent sur une seule variable au plus (les coûts unitaires et les coûts constants). L'objectif est de faire apparaître de bons minorants [37] pour le DFBB. 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 [38, 39], 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 issu du CPD proposés aux différents outils.

1.4.4 L'heuristique Multistart Steepest Descent (MSD)

En 2000, Wernisch, Hery et Wodak [40] 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.

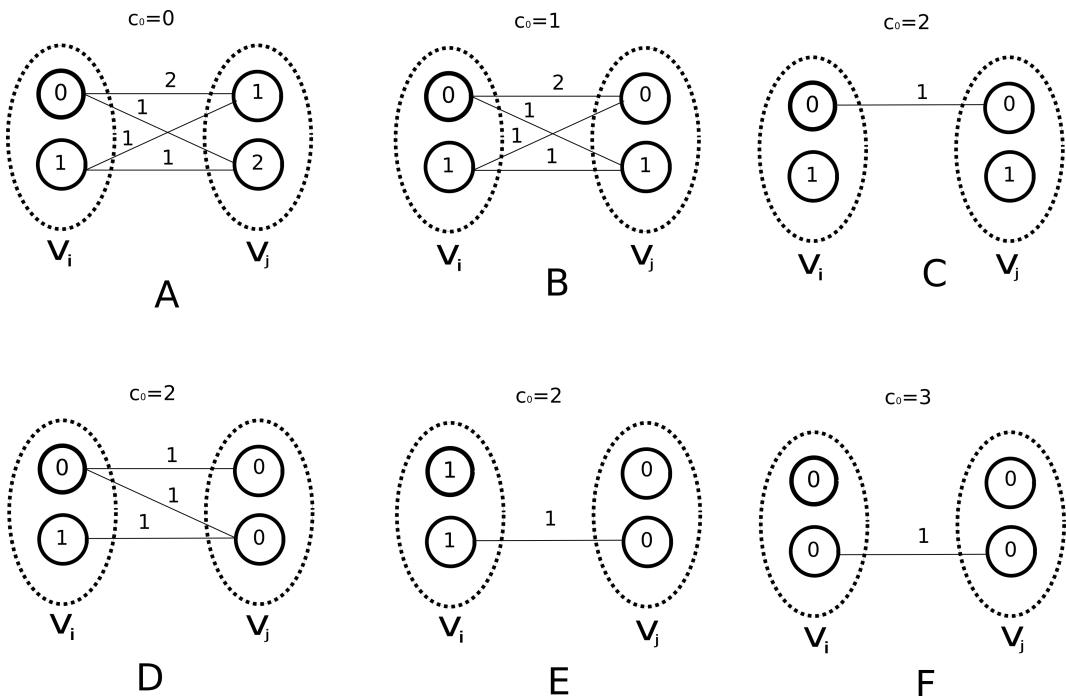


Figure 1.8 – **Exemple de transformations EPT sur un CFN composé de 2 variables v_i et v_j ayant deux valeurs possibles.** La transformation de A vers B est une projection de v_j vers le coût constant C_0 . Puis une projection des fonctions binaires mène en C . La seconde valeur de v_j est distribuée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

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 . 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  $C$  ;

```

Algorithm 1 : L'algorithme Multistart Steepest Descent

1.4.5 L'algorithme génétique

Holland et ces collaborateurs [41] 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, ce qui l'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 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, meilleur 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

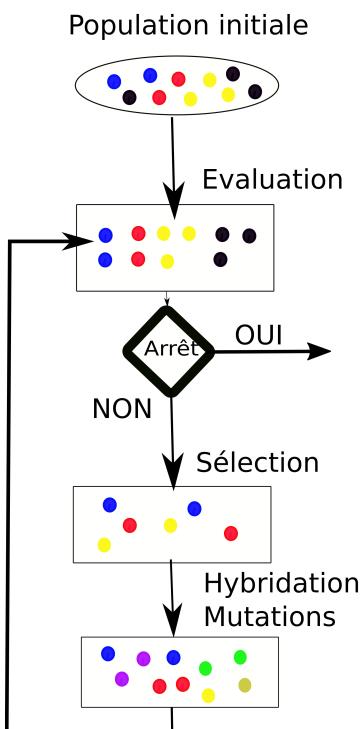


Figure 1.9 – L'algorithme génétique

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 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 générale 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 :

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

Dans la suite, pour simplifier les notations, on utilise l'écriture matricielle avec

$$\mu_0 = (P(X_0 = a))_{a \in E}$$

on a alors

$$\mu_1 = \mu_0 \cdot P$$

avec . le produit matriciel.

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 [42]. 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 :

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

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 .

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 x 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 l'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

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

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 avec $acc()$ 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  $\text{sel}(.,S)$ ;
4   calculer une probabilité d'acceptation  $\text{acc}$ ; si  $\text{acc} \geq 1$  alors
5      $S_{i+1} = S'_i$  ;
6     sauvegarder  $S_{i+1}$  ;
7   sinon
8     alors  $S_{i+1} = S'_i$  avec une probabilité  $\text{acc}$ ;
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 [43]. 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 $m = 1,..,N$ 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, celui 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.

- 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 acc_{swap} est calculée **si** $acc_{swap} \geq 1$ **alors**
- 5 Les marcheurs échangent leur température ;
- 6 **sinon**
- 7 | Les marcheurs échangent leur température , avec la probabilité acc_{swap}

Algorithme 3 : L'algorithme REMC

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 à entraîné sa grande popularité, en particulier dans le domaine de la modélisation moléculaire, mais aussi en physique, chimie, intelligence artificielle.

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) [15], OSPREY (Open Source Protein REdesign for You) [44], Rosetta (le CPD n'est qu'une partie des fonctionnalités proposées par cette suite logicielle) [11] et Proteus [29, 45]. 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 26.

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. De plus, on prend la différence entre états replié et déplié. Il s'agit donc d'une énergie libre de Gibbs G .

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

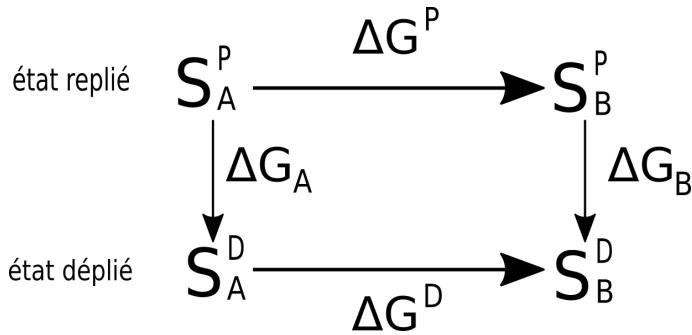


Figure 2.1 – Cycle thermodynamique qui définit la stabilité relative de deux séquences-conformations S_A et S_B .

Nous faisons des simulations avec deux mouvements élémentaires possibles : une changement de rotamère dans la sequence 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 respecte 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.
- un ensemble de scripts Xplor, qui pilote le calcul de la matrice d'énergie. Xplor est un programme de simulation moléculaire [46], disponible en téléchargement sur le site de l'université de Yale. Les modifications CPD sont disponibles sur notre site Web.
- 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 rotamère. 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 pouvant 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 de redesign de la tyrosyl-ARNt synthétase [47], sur les calculs de pK_a [48] ou la création de nouveaux domaines PDZ [49].

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 11). 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 10, 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 chaîne latérale, voir la figure 2.2.

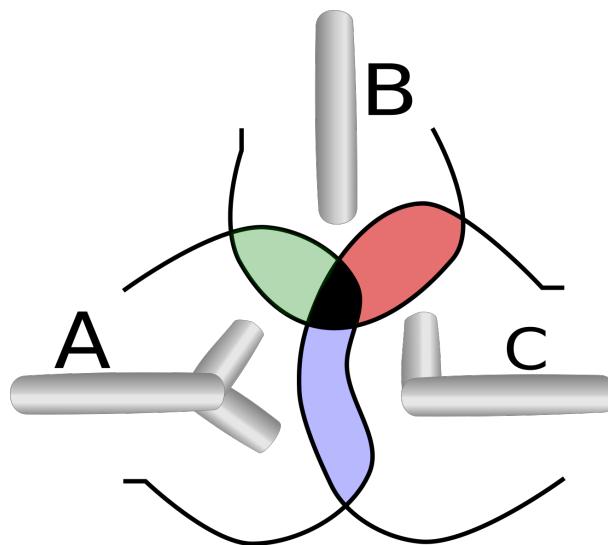


Figure 2.2 – Une représentation de la surface accessible de trois résidus. Les résidus A et B réduisent mutuellement leur surface exposée au solvant : c'est la zone verte. De même pour B, C : la zone rouge et A, C, la zone bleue. Un calcul par paires de résidus naïf surestime la surface accessible en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

Pour rendre ce terme décomposable, nous utilisons la méthode de Street et al. [50] 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 [51, 30].

2.3. Décomposition par paires de la fonction d'énergie

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 15. 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 [48], 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'atome. Ces rayons sont eux-mêmes sommes de paires sur les atomes de la protéine [52], [53]. 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} ,

voir 1.19 page 13, 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 exclue 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 [54], 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. Comme les rayons de sont eux-mêmes des sommes sur les paires, 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. Celui-ci permet d'obtenir 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.

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, voir la section 1.1.2 page 5, telle que :

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

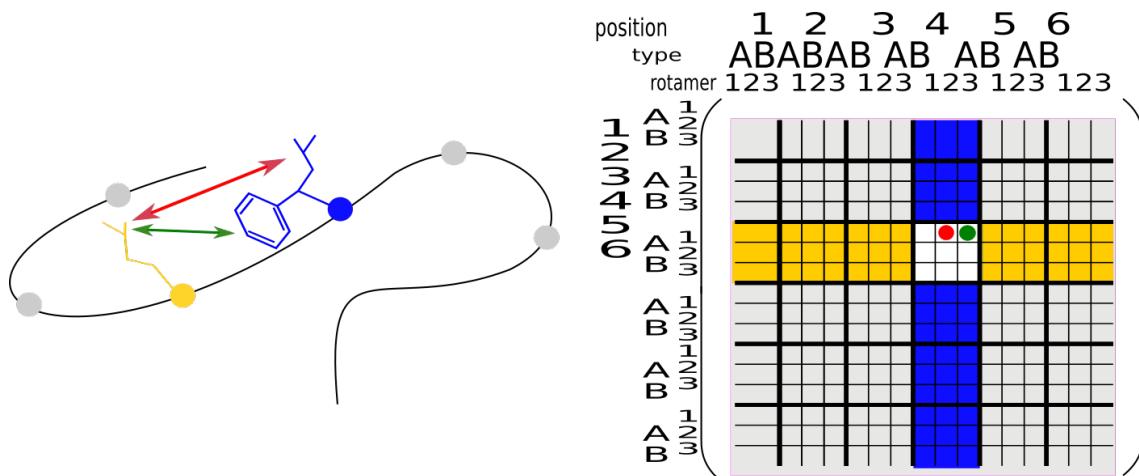


Figure 2.3 – **La matrice d'énergie.** Cet exemple montre un polypeptide de 6 résidus, chaque position possède 2 types d'acides aminés possibles et 3 rotamères possibles (2 pour le type A et 1 pour le type B). La matrice organise toutes les interactions de paires de chaînes latérales possibles. Les interactions 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.

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 [46]. 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 [9].
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 petite 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 petite 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. [55] 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 [30]. 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 [56, 57]. 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 42. 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 21. Pour le MC/REMC, le fichier de configuration détermine le nombre de marcheurs, la taille de chaque trajectoire, 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 1.40 page 28. 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

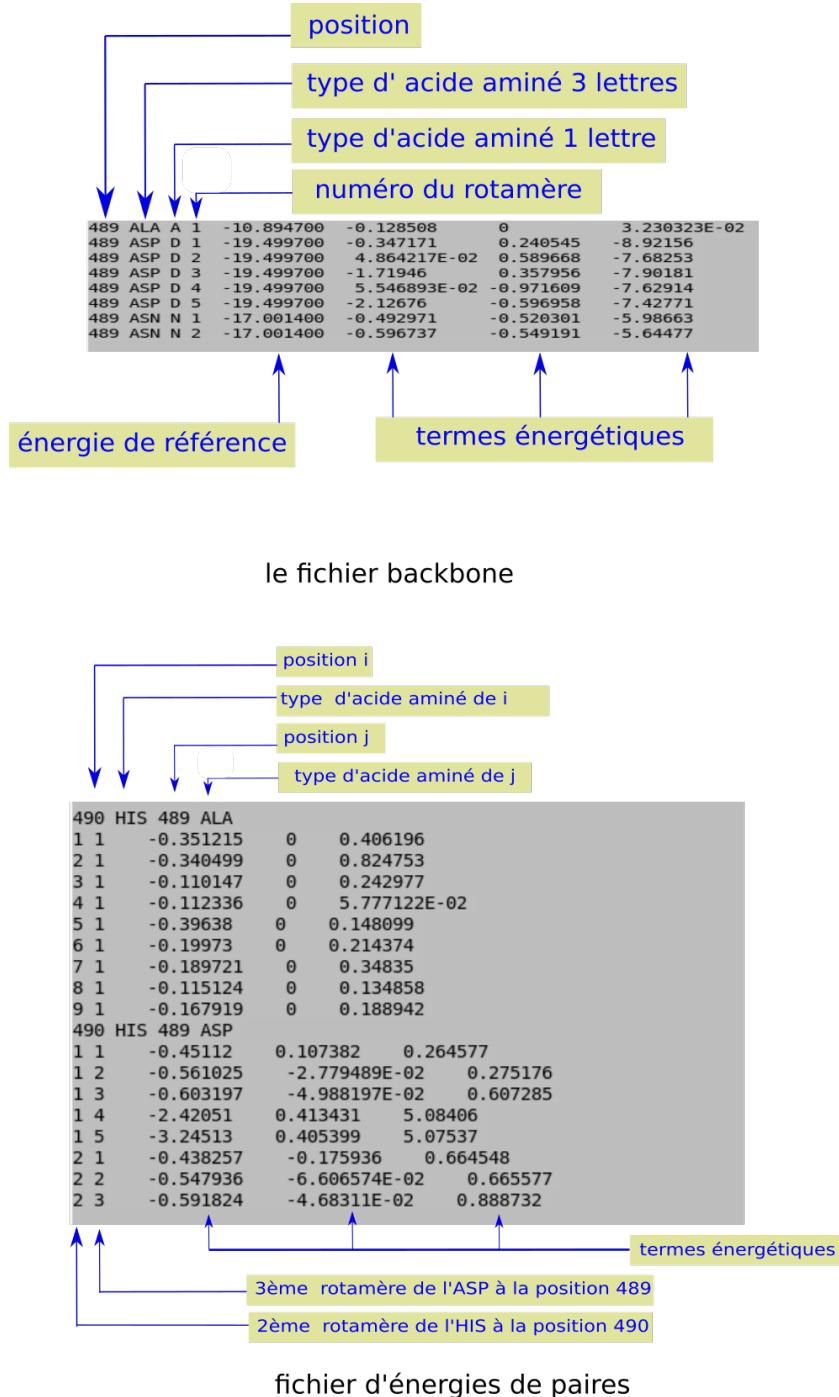


Figure 2.4 – les fichiers d'énergies en mode CASA

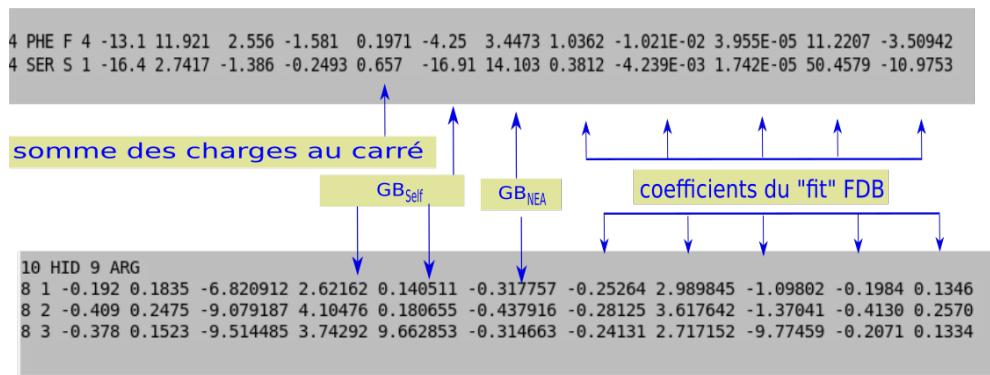


Figure 2.5 – les fichiers d'énergies en mode GB/NEA et FDB

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 2.5.3 page 44.

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, c'est-à-dire au cours d'un même pas Monte-Carlo. Le choix d'une première position à modifier se fait par tirage équiprobable 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 :

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

Chapitre 2. Proteus et nos outils d'analyse

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	restreint les états pouvant être visités
paramètre	Temperature	attribue les températures aux marcheurs REMC
Configuration Monte Carlo	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
	Step_Definition_Proba	probabilité de changer un rotamère ou un type d'acide aminé
	Neighbor_Threshold	Définit la taille des voisinages.
Input/Output	Fasta_File	le nom du fichier produit par POSTPROCESS
	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

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 59.

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.

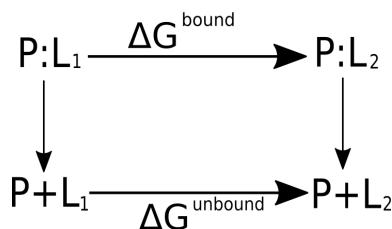


Figure 2.6 – Cycle thermodynamique qui définit l'affinité.

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

Chapitre 2. Proteus et nos outils d'analyse

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. Nous détaillons maintenant la façon dont ce dispositif fonctionne.

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 :

$$E(grp_i) = \sum_{a \in grp_i} E_a + \sum_{a \in grp_i} \sum_{b \in grp_i, a < b} E_{a,b} \quad (2.13)$$

et

$$E(grp_i, grp_j) = \sum_{a \in grp_i} \sum_{b \in grp_j} E_{a,b} \quad (2.14)$$

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 2.7. 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.

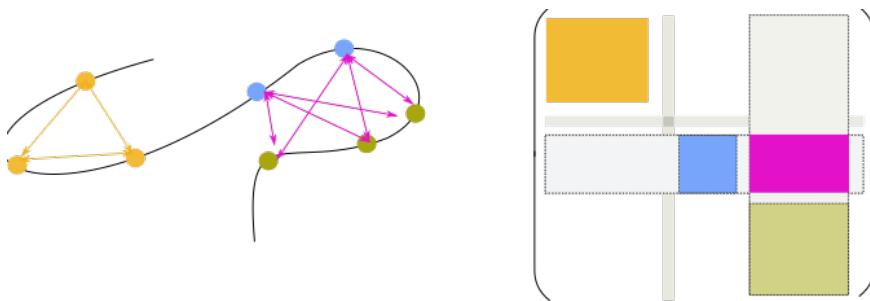


Figure 2.7 – Les contributions aux énergies de groupes et d'interaction entre groupes dans la matrice d'énergie. L'énergie des groupes orange, bleu et vert est une somme de termes situés des carrés de même couleur dans la matrice d'énergie. L'interaction entre le groupe bleu et le groupe vert est la somme de tous les termes du rectangle rouge.

Déclaration de la fonction de score

On peut d'introduire une nouvelle matrice qui rassemble les énergies des groupes et les énergies d'interaction entre les groupes, voir la figure 2.7. 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+0.3grp2~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 $t(\dots)$ 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 2.8 – **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).

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 21. 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.

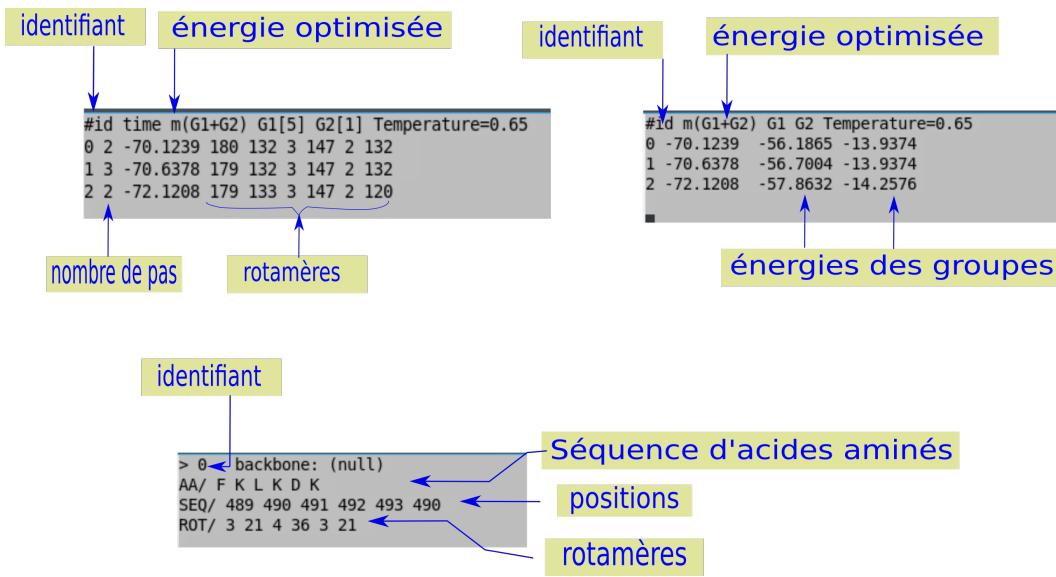


Figure 2.9 – Les fichiers en sortie de proteus en haut pour le mode MONTECARLO, en bas pour le mode POSTPROCESS

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 2.9.

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 [58] comprend :

- une base de données de modèles de Markov cachés, où chaque modèle représente une structure 3D d'un domaine de la classification SCOP [59].
- 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) [60] et HMMER (version 3.0) [61], deux outils de manipulation de modèles de Markov cachés pour la biologie recommandées 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 N est égal au pourcentage de positions où l’acide aminé est identique dans S et N :

$$Id(S,N) = \frac{1}{l} \sum_{1 \leq i \leq l} \mathbb{1}(s_i, n_i) \times 100 \quad (2.15)$$

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 :

$$Id(A_S, i) = \frac{1}{m} \sum_{1 \leq j \leq m} \mathbb{1}(s_i^j, n_i) \times 100 \quad (2.16)$$

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) [62, 63] 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. Par exemple, 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) [64] 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 [65, 66]. 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 [67], à partir de la formule :

$$S_i = - \sum_{j=1}^6 f_j(i) \ln f_j(i) \quad (2.17)$$

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 [68]. 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 2.10. 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 2.5.3. 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 2.11.

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 2.8. La structure `conformation` contient la séquence-conformation courante. Une liste de structures `ins_modif` représente les modifications à effectuer sur la conformation courante, figure 2.12. Ceci permet la mise à jour des différentes énergies au cours de la trajectoire.

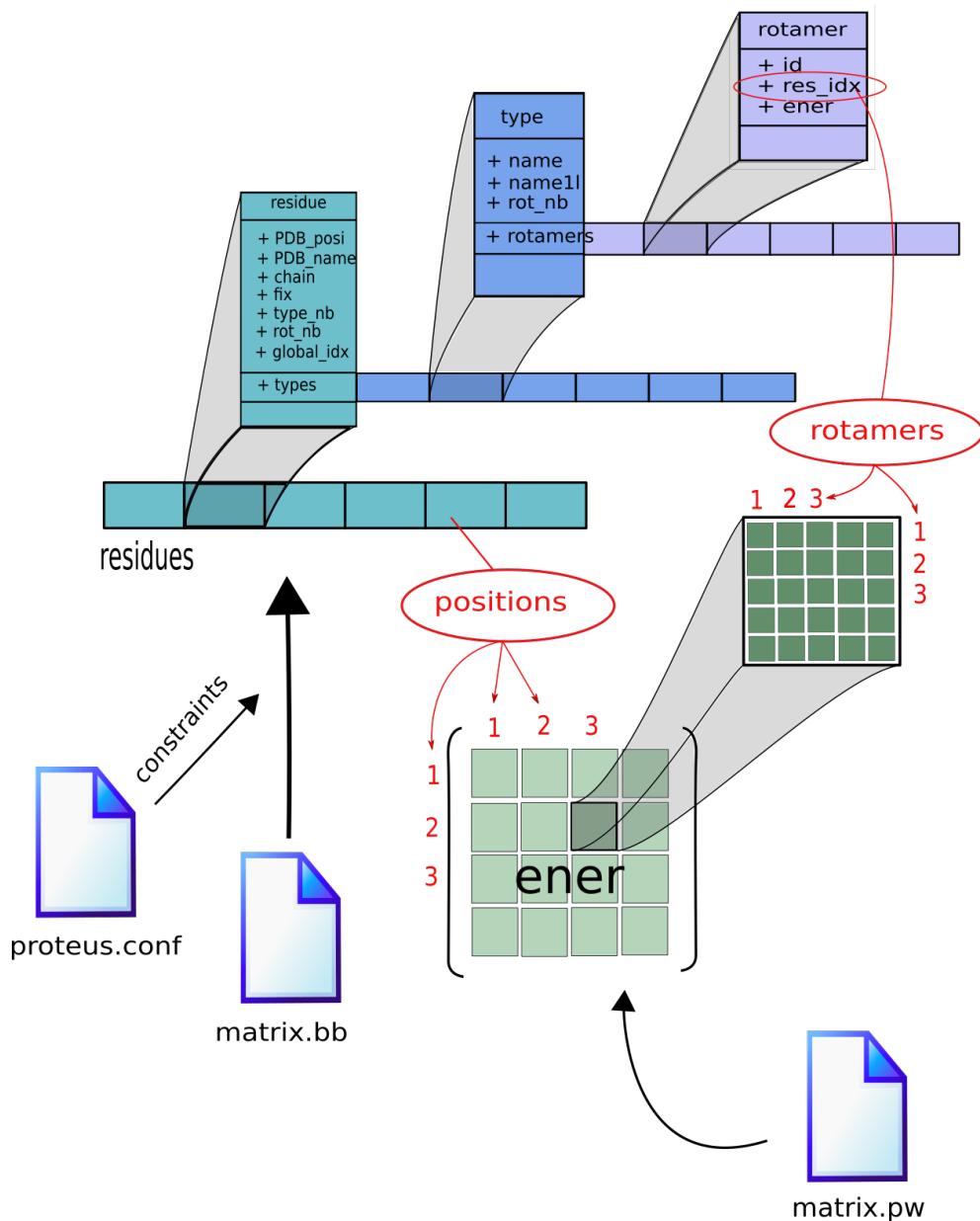


Figure 2.10 – Les principales structures « physiques » dans proteus

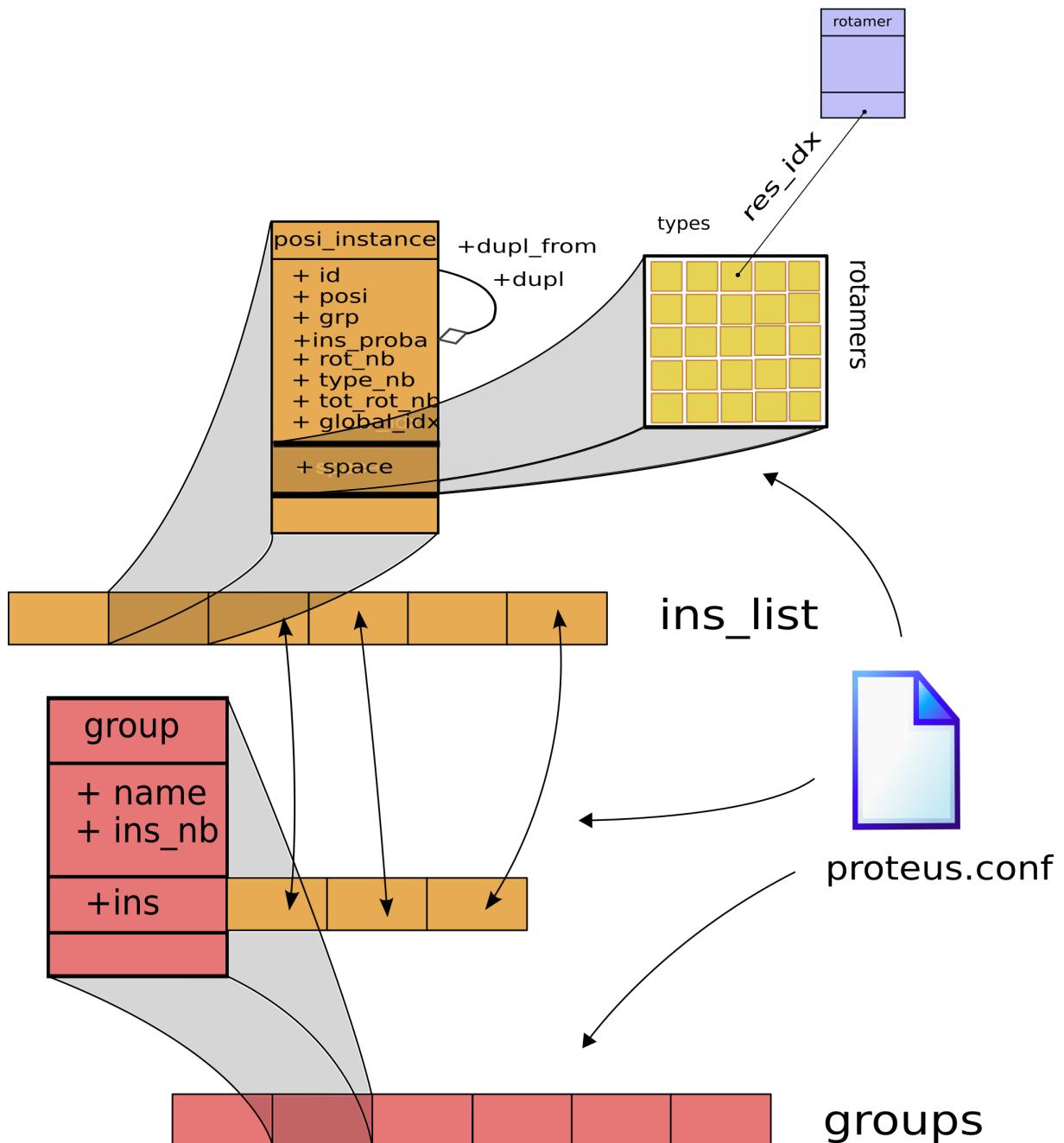


Figure 2.11 – Les principales structures « logiques » dans proteus

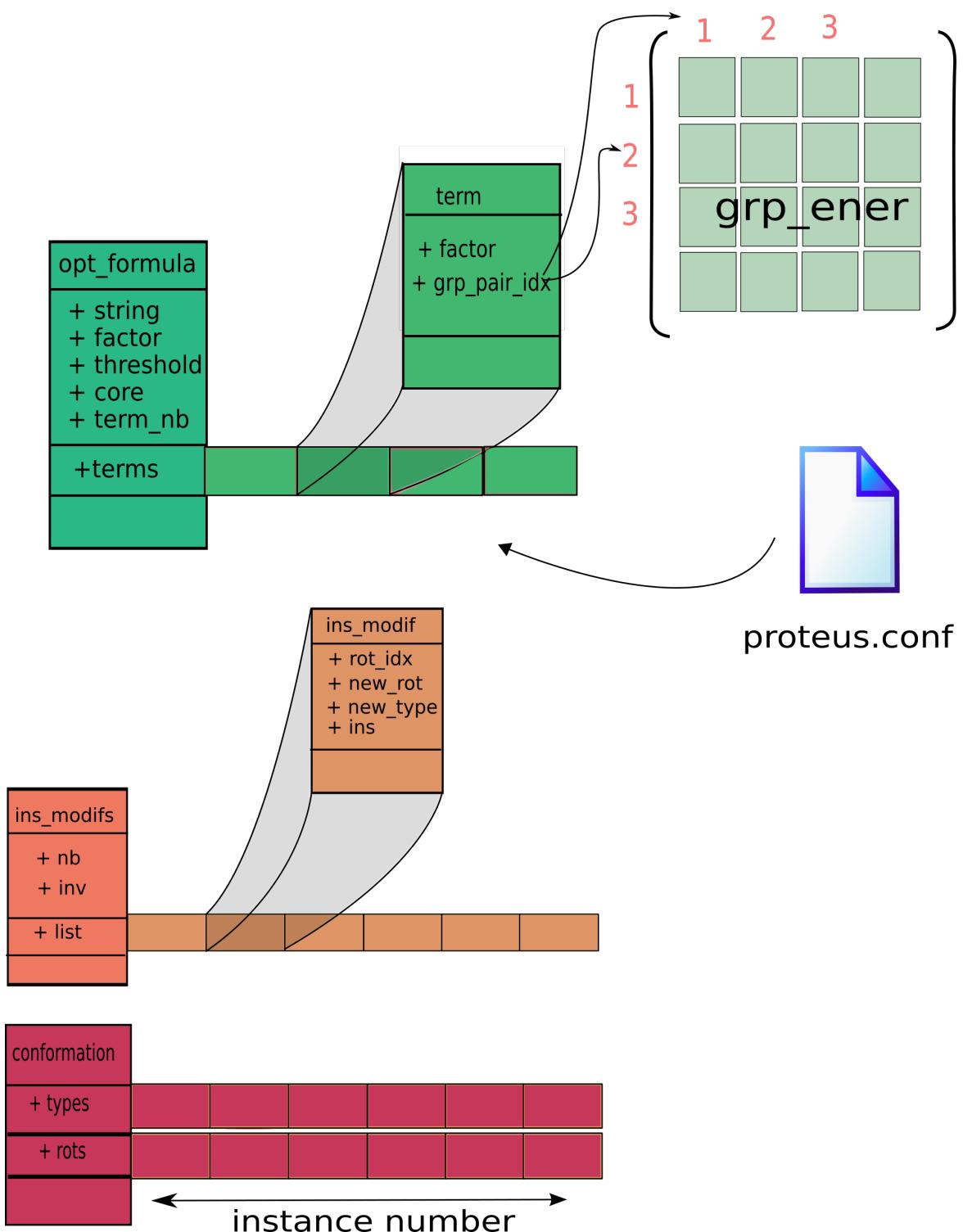


Figure 2.12 – Les principales structures « dynamiques » dans proteus

Chapitre 3

Développements

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 [46]. 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

Chapitre 3. Développements

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.

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

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 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.

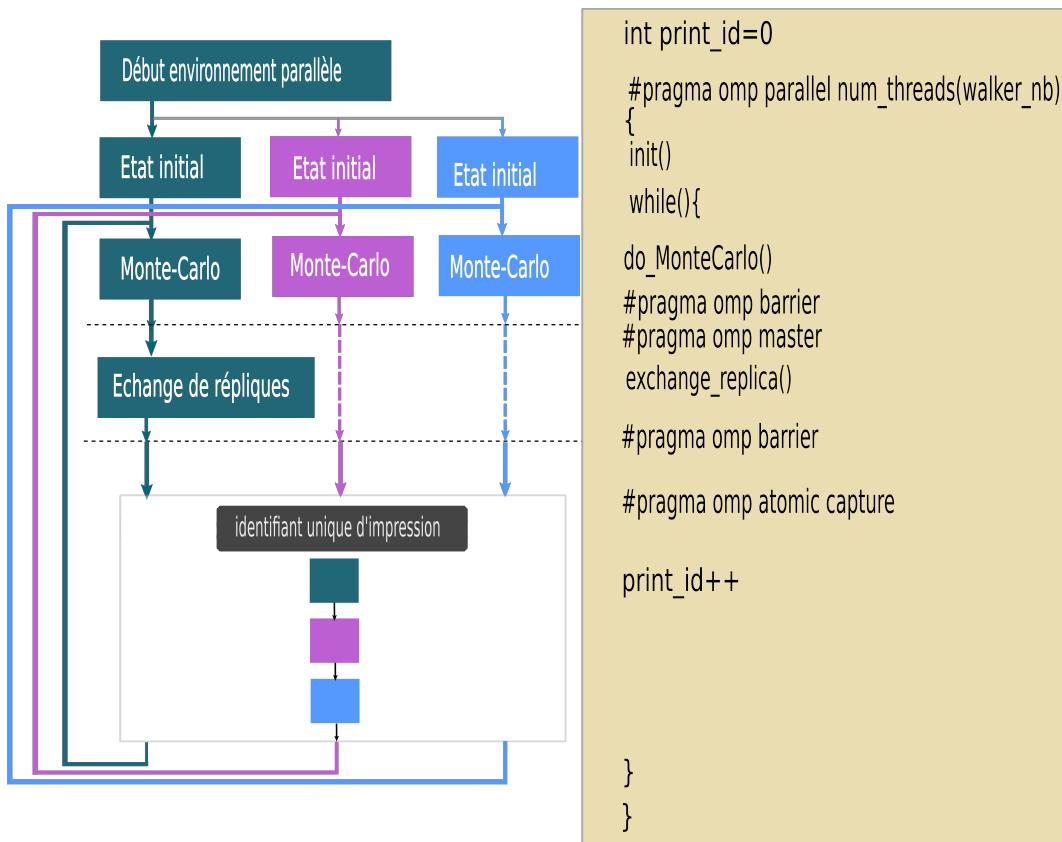


Figure 3.2 – La correspondance entre les directives OpenMP à droite et le comportement des fils d'exécution à gauche sur un REMC simplifié. Sont schématisés, la création d'une région parallèle, deux synchronisations (les lignes pointillées à gauche), une région exécutée uniquement par un fil maître,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

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

3.3. Amélioration de la fonction d'acceptation MC

seul marcheur maître. Pour empêcher les autres de marcher avant un échange éventuel, une seconde directive « barrier » est placée après les instructions d'échanges.

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.

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 ???. Cette nouvelle probabilité d'acceptation est décrite en détail en 4.6 page 68.

3.4 Seuil d'impression

La quantité de séquences-conformations écrite dans les fichiers de sortie peut devenir importante surtout en REMC. Pour limiter la taille des fichiers et faciliter le post-traitement, nous introduisons la balise `<Print_Threshold>`. Elle prend en paramètre un seuil s

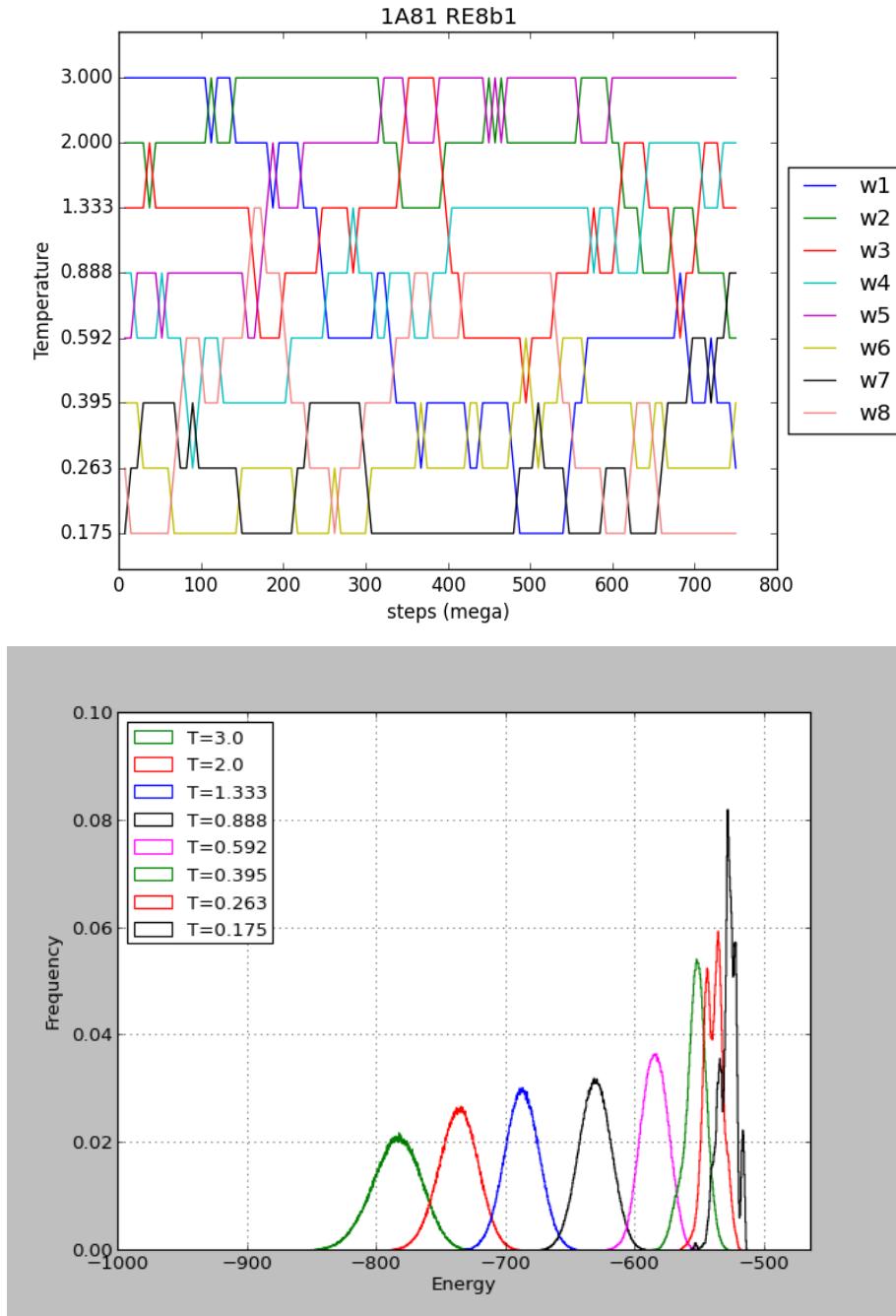


Figure 3.3 – Comportement de proteus pour un REMC à huit marcheurs. En haut, la trajectoire des marcheurs dans l'ensemble des huit températures. En bas, la distribution des énergies par marcheurs. On peut imaginer la distribution cumulée des marcheurs et remarquer que son graphe serait proche de celle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

qui conditionne l'impression d'une séquence-conformation de la façon suivante. Chaque marcheur conserve au cours d'une simulation la meilleure énergie obtenue E_{best} . À tout moment une séquence-conformation est imprimée seulement si l'écart entre son énergie et E_{best} est strictement plus petit que s . Ainsi, un seuil de 0 limite les impressions aux seules séquences-conformations qui améliorent l'énergie durant la simulation.

3.5 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.

3.6 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 déclaration, 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 « exposed » 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.

3.7 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 » [110].

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 [69? –73]. 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 [74–76]. 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”) [1, 77? –79]. 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 [?]. 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 [? 80, 81].

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 [82, 13], branch-and-bound methods [? ?], and cost function network methods [39, 38]. 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 [83, 75]. 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 [? ?].

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 [40, 79?]. 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 [30], verified here), then a very long trajectory is guaranteed to converge to a stationary state and obey the so-called detailed balance condition [28] (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 [84? ?]. The third method is an enhanced, multiwalker MC, which performs “replica exchange” [85–87]. 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 [39, 38]. 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 [?]. 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 [88–90]. With these simplifications, all possible residue pair interactions can be computed ahead of time and stored in a lookup table [91]; 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 [89, 90] (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 exception (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 [92, 93], 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 form 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 [84? ?] 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 [84? ?]. 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 [84? ?] :

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

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 [84].

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 :

$$\begin{aligned} acc(t \rightarrow t') &= e^{-\beta(\Delta E_f - \Delta E_u)} \frac{n_f(t)}{n_f(t')} = e^{-\beta(\Delta E_f - \Delta e_u)} \frac{n_f(t)n_u(t')}{n_u(t)n_f(t')} \quad \text{if } \Delta E_M > 0 \quad (4.8) \\ &= 1 \quad \text{otherwise} \end{aligned} \quad (4.9)$$

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 [86, 87].

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 [86]. 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 ;	$0.75 \cdot 10^9$	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 [40, 89]. 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 [39, 38]. 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 [?], 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 [?]. The last term represents the contribution of solvent. We used a “Coulomb + Accessible Surface Area”, or CASA implicit solvent model [51, 88] :

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

Here, ϵ is a dielectric constant that scales the Coulomb energy, set to 23, following earlier tests [88]. 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 [94], 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 [50, 51]. Energy calculations were done with a modified version of the Xplor program [46, 90].

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 [89]; they are reported in Table ??.

4.2.6 Test systems and preparation

We considered nine protein domains, from the SH3, SH2, and PDZ families, listed in Table 4.2. 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 [79?]. 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 [55], 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 [49]. 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) [56, 57].

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 4.2 – 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	1M61	109	Zap70
SH3	1ABO	58		Abl	SH2	1O4C	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 [92, 93], which attempts to classify sequences according to the SCOP classification [59]. 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 [67], 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 [?], and also by analyzing residue-residue contact energies in proteins [68]. 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 4.3 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 4.3. Thus, our designed sequences are mostly similar to experimental ones. Sequence identities to wildtype are 31% on average (Table 4.3), similar to earlier studies with the same energy function [79?]. 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 4.2. 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. ??, 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).

For the same PDZ domain, the lowest energy sequence/structure combination is shown in figure ??, 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 SH3 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

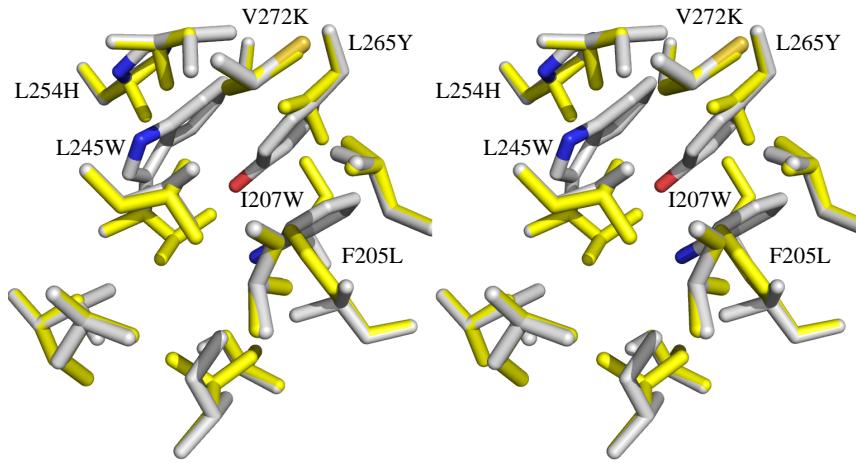


Figure 4.2 – The lowest energy sequence/structure combination is shown for the 2BYG PDZ domain (stereo view; selected sidechains only; white sticks), superimposed on the Xray structure (yellow sticks); the sidechains shown are the ones in the sequence logo, Fig.2. The main mutated positions are labelled; other positions are either not mutated or mutated within the ILV group. Figure produced with pymol [72]

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.

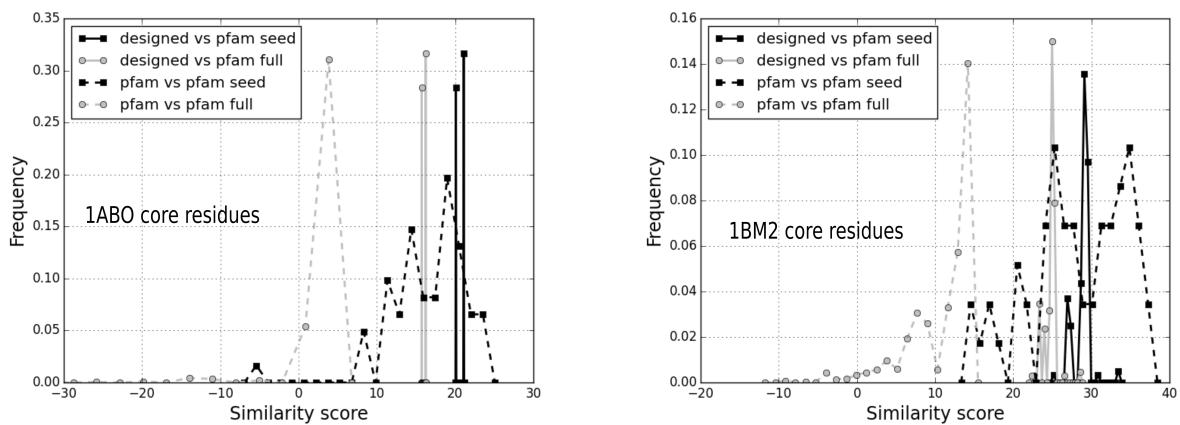


Figure 4.3 – Histogram of Blosum40 similarity scores to Pfam sequences for the core region of two designed proteins : 1ABO (SH3) and 1BM2 (SH2). The similarity between Pfam sequences is also shown, considering either the Pfam seed alignment or the much larger full alignment.

Table 4.3 – 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 resources 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.

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

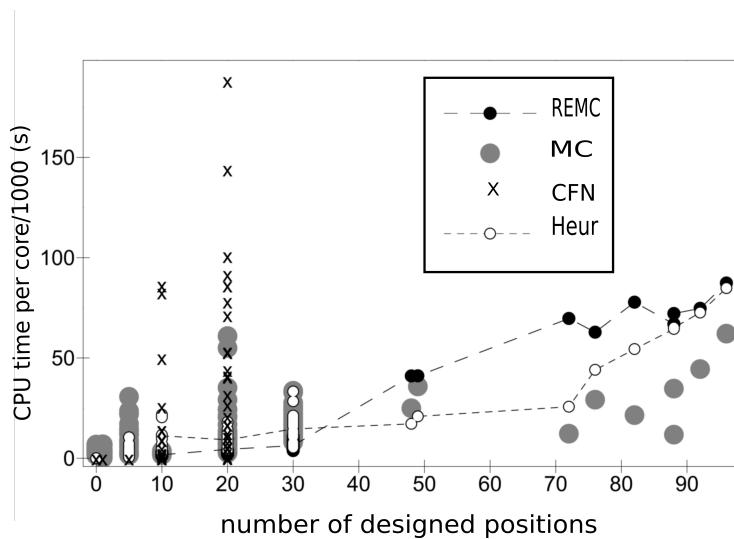


Figure 4.4 – Run times for different test calculations and search methods. CPU times per core are shown ; only the REMC calculations use multiple cores (OpenMP parallelization). CFN results are only given for 20 designed positions or fewer. REMC was only done for the larger calculations. Results are 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.

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.

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. ???. With zero positions, only rotamers are optimized (at all positions in the protein). With one, we systematically

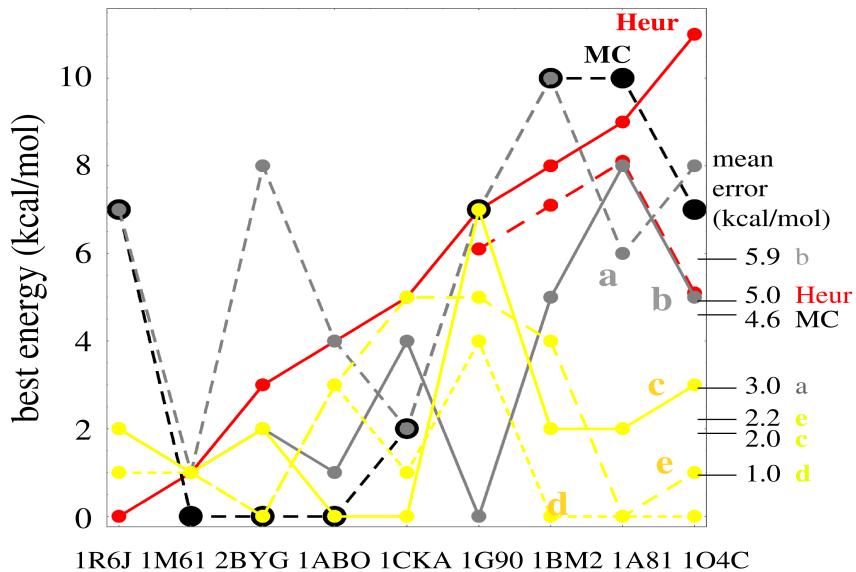


Figure 4.5 – Comparison between selected heuristic, MC, and REMC protocols for whole protein design. The best energy obtained with each protocol is shown for the nine test proteins. Zero represents the overall best energy for each protein. The mean “error” is indicated on the right for each protocol (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.

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.

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. ?? and Table 4.4. 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

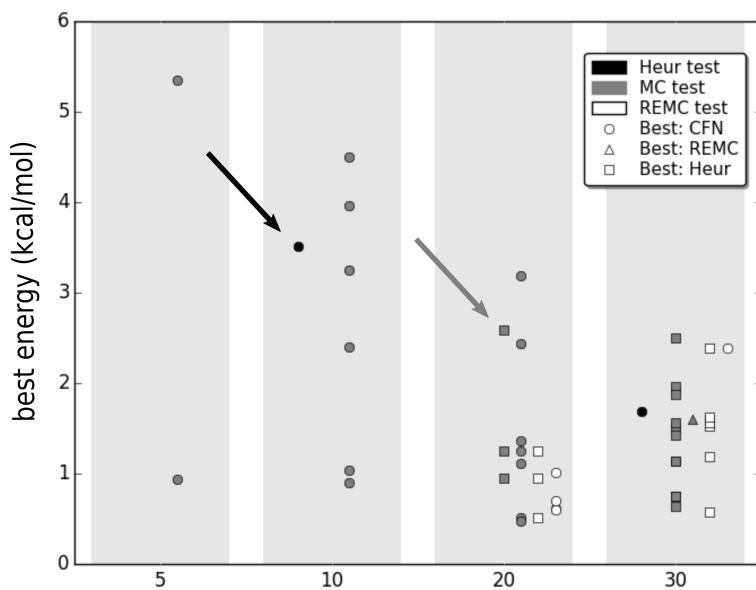


Figure 4.6 – The lowest energies obtained with the different exploration methods for 5-, 10-, 20-, and 30-position design. Differences with respect to the GMEC or the overall best energy are shown, excluding the smallest values (less than 0.4 kcal/mol above the best energy). The color of each point indicates 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. The dashed red line shows heuristic results with a more aggressive protocol (330,000 heuristic cycles) for the four rightmost proteins.

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 [?], 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

4.3. Results

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.

Table 4.4 – 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.
2819	91(15)	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	
3037	104(8)	1O4C 2	gmec	0.	0.1046	
		1O4C 3	gmec	0.	0.1519	
		1O4C 4	gmec	0.	0.1545	
		1O4C 5	gmec	0.	0.1753	
		1R6J 1	gmec	0.	2.4022	0.3986
2773	82(10)	1R6J 2	gmec	0.	1.0398	0.3049
		1R6J 3	gmec	0.	0.0106	
		1R6J 5	gmec	0.	0.0162	
		2BYG 1	5.7485	0.	0.0337	
2888	97(15)	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.

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 [51, 88, 30], 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 4.5. 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, half were rerun with a third, more recent CFN protocol [?] ; 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 4.5. 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 4.6 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 [79?]. 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 [79?].

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 4.5 – 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 4.4. gmec* indicates the more aggressive protocol. ^aBetween CFN/Heur.

Table 4.6 – 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. They often take the form of MC-based simulated annealing (SA) [1, 58] ; various forms of biased or quenched Monte

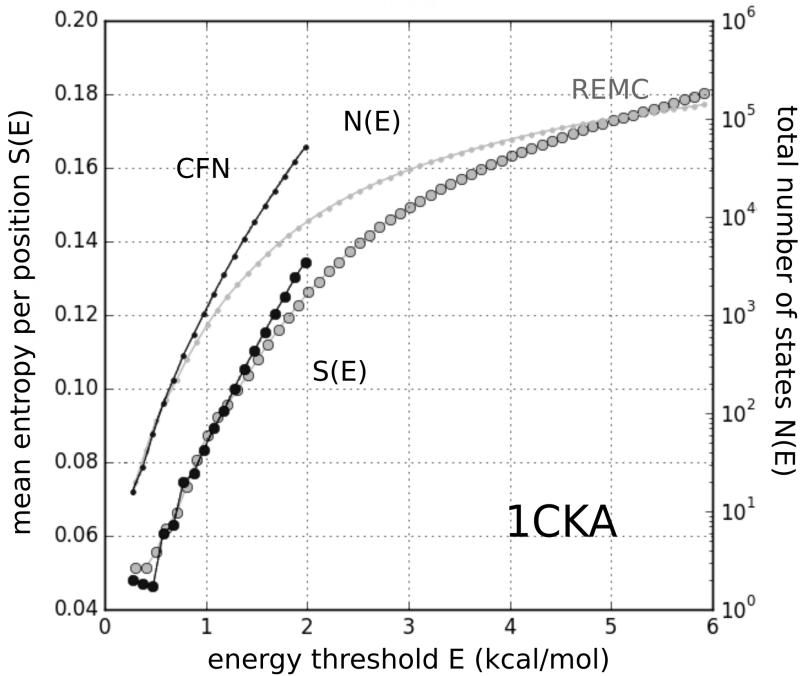


Figure 4.7 – Sequence entropy $S(E)$ and number of states $N(E)$ within a given energy range E above the GMEC for the 1CKA SH3 domain. All positions were allowed to vary except Gly/Pro. The entropy (large dots) is a single position sequence entropy, Eq. (4.13), averaged over all the variable positions. 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.

Carlo are also popular [59, 61–63]. For example, MC moves can be chosen to preferentially affect residues that have a high local energy [62] 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 [32] has similarities to quenched MC. These particular quenched or biased stochastic methods do not sample a Boltzmann 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 [64–66], 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 [21, 67], and can handle slight departures from a purely additive energy function (ones including 3- or 4-body energy terms) [68, 69], they cannot readily handle arbitrary energy functions or a flexible backbone. Other stochastic

4.4. Concluding Discussion

methods preserve the Boltzmann character of the statistical ensemble, including plain Monte Carlo, Replica Exchange MC, and configuration bias MC methods [28, 58]. 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 [?]. 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) [79?]. Results with a more advanced protein force field and Generalized Born solvent are expected to be even better [28?]. 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 4.12 et 4.13. Pour les sélections 20-actives et 30-actives, il a fallu descendre le seuil à 1 pour obtenir les 45 voisinages forts proches de l'objectif, il manque une ou deux positions dans quelques cas, voir les tables 4.14 et 4.15.

Table 4.7 – 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 4.8 – 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 4.9 – 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 4.10 – 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
1G9O 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
1G9O 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
1G9O 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
1G9O 4	10 11 13 14 15 31 32 34 38 40 41 42 43 46 48 49 50 51 54 57 58 60 61 62 68 89 90 91 92 94 95 96
1G9O 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
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

Application aux 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 [96–100]. 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 [101, 102]. Des domaines PDZ et leurs ligands redessinés ont été utilisés pour élucider les principes du repliement des protéines et de l’évolution [103–105]. Ainsi, ces domaines avec leurs ligands peptidiques fournissent d’excellents « benchmarks » pour tester les méthodes informatiques elles-mêmes [106, 107].

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 [108]. 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 des paramètres essentiels 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, des valeurs expérimentales. Pour cela, nous choisissons les E_t^r qui maximisent la probabilité de Boltzmann d'un ensemble de séquences expérimentales. C'est à dire, nous retenons 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 . Comme nous voulons le maximum de \mathcal{L} sur les E_t^r , nous appelons à \mathcal{L} la vraisemblance des E_t^r [109]. Nous avons :

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

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 les 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

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

Ainsi, pour maximiser \mathcal{L} , nous choisissons les E_t^r tels 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 [109], nous l'appelons méthode linéaire :

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

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.

- 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 rapide de l'état de l'optimisation :

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

Alors, à chaque itération on applique quatre fois la règle 5.10. Les trois premières fois 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 le quatrième cycle, au terme duquel les énergies sont mises à jour.

- La troisième méthode, utilisée précédemment [89, 90], 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 une 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 exposée accessible au solvant de l'atome i , σ_i est un paramètre qui représente la préférence de chaque atome à être exposé ou caché du 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 [110, 111] : non polaire -5,

aromatique -40, polaire -80, ionique -100. 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 [57] : non polaire -3, aromatique -1, polaire -9, ionique -9. 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 [94], qui est implémenté dans le programme Xplor [46] et expliqué en 2.3.1. Les simulations MC utilisent une constante diélectrique pour la protéine $\epsilon_p = 4$ ou 8 (voir la partie Résultats). Nous utilisons deux variantes du modèle GB, la méthode NEA pour « Native Environment Approximation » et la méthode FDB ou « Fluctuating Dielectric Boundary » [48]. Elles sont présentées respectivement en 1.3.5 et 2.3.3.

Le champ de force utilisé, Amber ff99SB [16], 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 [112].

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 (voir plus bas).

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, estimée pour une conformation dépliée où chaque acide aminé interagit uniquement avec lui-même et avec le solvant.

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. Pendant, la maximisation de la vraisemblance, E_c^r est optimisé tandis que δE_t^r est fixe. 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 soit des classes d'acides aminés, soit 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, 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 supérieur 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

5.4. Séquences expérimentales et modèles structuraux

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

Proteine	NHREF	INAD	GRIP	Syntenin	DLG2	PSD95	CASK	TIAM1
NHREF	$2 \cdot 10^{-66}$ (100)	$5 \cdot 10^{-10}$ (40)	$2 \cdot 10^{-3}$ (25)	$3 \cdot 10^{-7}$ (25)	$2 \cdot 10^{-11}$ (35)	$1 \cdot 10^{-12}$ (30)	$5 \cdot 10^{-5}$ (25)	$9 \cdot 10^{-7}$ (35)
INAD	$5 \cdot 10^{-10}$ (40)	$3 \cdot 10^{-68}$ (100)	$2 \cdot 10^{-7}$ (27)	[18]	$2 \cdot 10^{-8}$ (27)	$9 \cdot 10^{-14}$ (46)	$4 \cdot 10^{-6}$ (35)	[16]
GRIP	$2 \cdot 10^{-3}$ (25)	$2 \cdot 10^{-7}$ (27)	$3 \cdot 10^{-67}$ (100)	[21]	$3 \cdot 10^{-14}$ (36)	$2 \cdot 10^{-10}$ (37)	$9 \cdot 10^{-12}$ (30)	$5 \cdot 10^{-5}$ (35)
Syntenin	$3 \cdot 10^{-7}$ (25)	[18]	[21]	$1 \cdot 10^{-59}$ (100)	[17]	$1 \cdot 10^{-6}$ (32)	$7 \cdot 10^{-3}$ (32)	[18]
DLG2	$2 \cdot 10^{-11}$ (35)	$2 \cdot 10^{-8}$ (27)	$3 \cdot 10^{-14}$ (37)	[17]	$7 \cdot 10^{-71}$ (100)	$3 \cdot 10^{-15}$ (37)	$2 \cdot 10^{-7}$ (28)	$5 \cdot 10^{-5}$ (41)
PSD95	$1 \cdot 10^{-12}$ (30)	$9 \cdot 10^{-14}$ (46)	$2 \cdot 10^{-10}$ (36)	$1 \cdot 10^{-6}$ (32)	$3 \cdot 10^{-15}$ (37)	$4 \cdot 10^{-70}$ (100)	$1 \cdot 10^{-7}$ (27)	$6 \cdot 10^{-4}$ (33)
Cask	$5 \cdot 10^{-5}$ (25)	$4 \cdot 10^{-6}$ (35)	$9 \cdot 10^{-12}$ (30)	$7 \cdot 10^{-3}$ (32)	$2 \cdot 10^{-7}$ (28)	$1 \cdot 10^{-7}$ (27)	$7 \cdot 10^{-61}$ (100)	$5 \cdot 10^{-4}$ (33)
Tiam1	$9 \cdot 10^{-7}$ (35)	[16]	$5 \cdot 10^{-5}$ (35)	[18]	$5 \cdot 10^{-5}$ (41)	$6 \cdot 10^{-4}$ (33)	$5 \cdot 10^{-4}$ (33)	$1 \cdot 10^{-68}$ (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 5.2, 5.4, 5.6, 5.8, 5.10 et 5.12.

Table 5.4 – Sélection des homologues.

protéines	nombre	E-value	% identité
NHREF	62	$\leq 10^{-32}$	67-95
INAD	42	$\leq 10^{-10}$	38-95
GRIP	48	$\leq 10^{-45}$	84-95
Syntenin	85	$\leq 10^{-43}$	85-95
DLG2	43	$\leq 10^{-41}$	78-95
PSD95	50	$\leq 10^{-46}$	81-95
Cask	126	$\leq 10^{-27}$	60-85
Tiam1	50	$\leq 10^{-22}$	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 exposé 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.

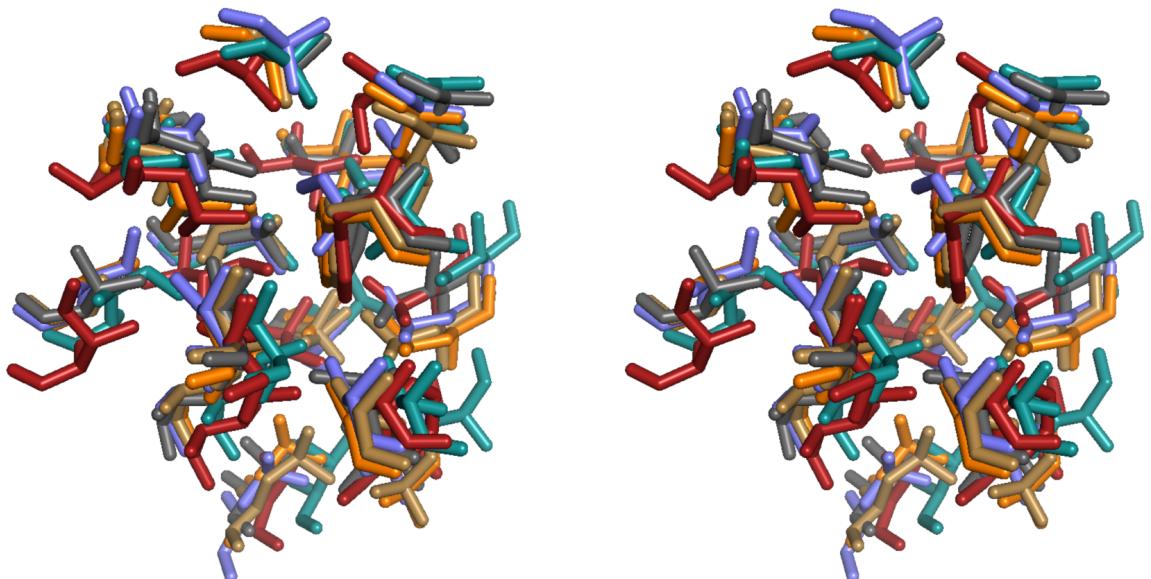
Protein	NHREF	INAD	GRIP	Syntenin	DLG2	PSD95	Cask	Tiam1
NHREF	326	64	15	15	59	112	49	1
INAD	64	221	56	-9	88	107	25	9
GRIP	15	56	378	24	65	87	90	39
Syntenin	15	-10	24	311	-26	22	42	-18
DLG2	59	88	65	-26	325	110	24	22
PSD95	112	107	87	22	110	325	66	21
Cask	49	25	90	42	23	66	308	37
Tiam1	1	10	39	-18	22	21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

5.4. Séquences expérimentales et modèles structuraux

1G90 RMLPRLCCLEK.GPNGYGFHLHGEKGKL.....GQYIRLVEPGSPAEKAG.LLAGDRIVEVNGENVEKETHQQVVSRIRAALNAVRLLVVDPETDEQL
 1IHJ GELIHMVTLGKSFVICIVRGEVKDSPNTKTGIFIKGIVPDSPAHLCGRLKVGDRILSNGKDVRNSTEQAVIDLKEADFKIELEIQTFL
 1N7E GAIYTVELKR.YGGPLGITISGTEEP.....FDPIISSLTKGLAERTGAIHIGDRILAINSSSLKGKPLSEAIHLLQMAGETVTLKIKKQTDQPASS
 1R6J GAMDPRTITMHDSTGHVGFIKRN.....GKITSIVKDSSAARNG.LLTEHNICEINGQNVIGLKDSQIADILSTSGTVVTITIMPAF
 2BYG FQSMTVVEIKLFK.GPKGLGFSIAGGVGNQH.IPGDNSIYVTKIIDGGAAQKDGRQVGDRLLMVNNSYLSSEEVTHEEAVAILKNTSEVVYLKGKPTT
 3K82 EDIPREPRRIVIHR.GSTGLGFNIVGGEGE.....GIFISFILAGGPADLSGELRKGDQILSVNGVDLRNASHEQAIAALKNAGOTVTIIAQYKPEEYSRFEA
 CASK RSRLVQFQKNTDEPMGIFTLKMNELN.....HCVARIMHGGMIHRQGTLHVGDIEIREINGISVANQTVEQLQKMLREMRSITFKIVP
 .TIAM1 GAMGKVTHSIIEKSDTAADTYGFSLSSVEED.....GIRRLYVNSVKETGLASKKG.LKAGDEILEINNRAADALNSSMLKDFLSQP..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 coeurs 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.

5.4. Séquences expérimentales et modèles structuraux

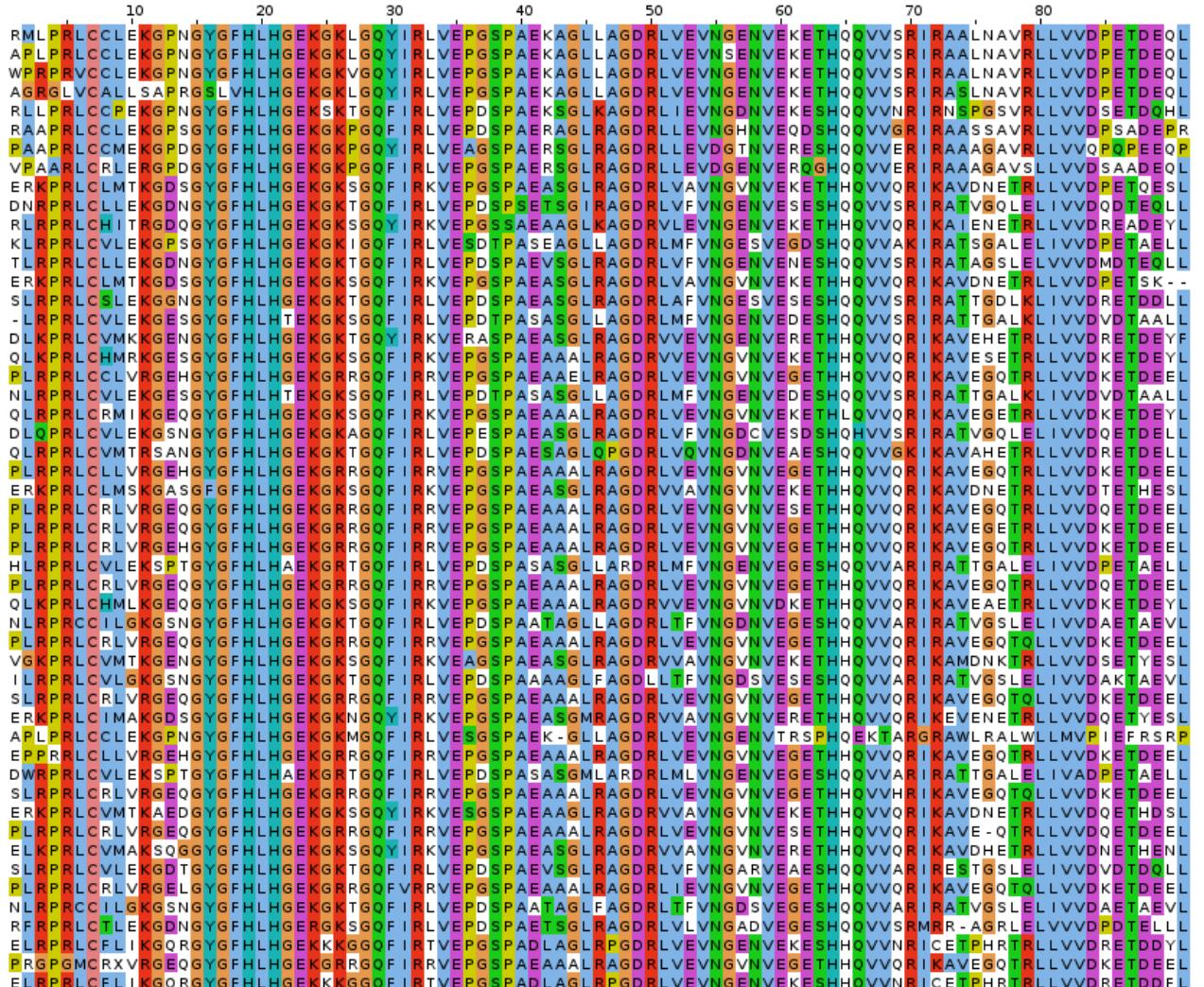


Figure 5.2 – L’alignement de notre sélection de séquences homologues à la protéine NHREF (code PDB : 1G9O)

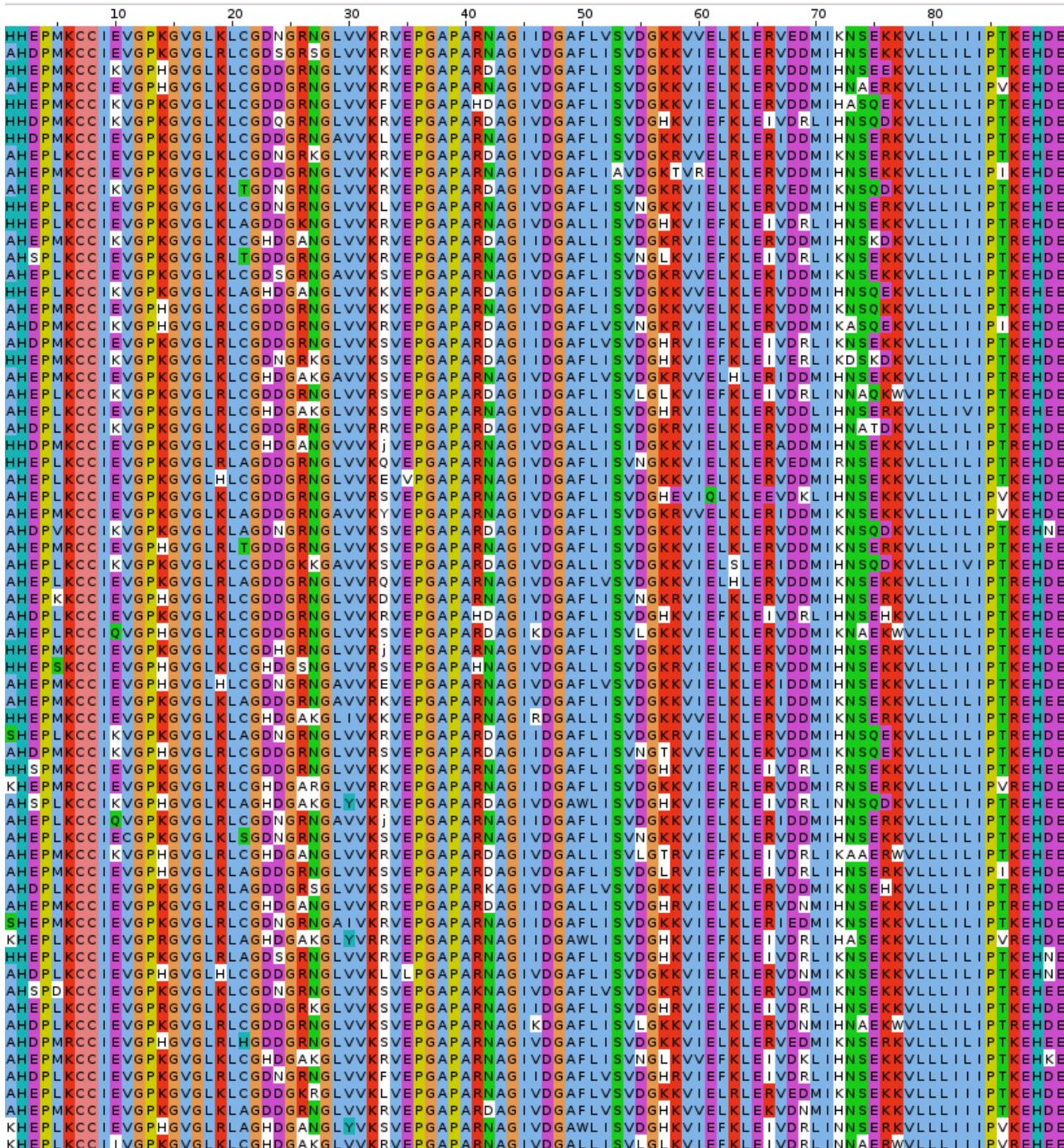


Figure 5.3 – Une sélection de séquences protéiques, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine NHREF (code PDB : 1G9O), modèle NEA

5.4. Séquences expérimentales et modèles structuraux

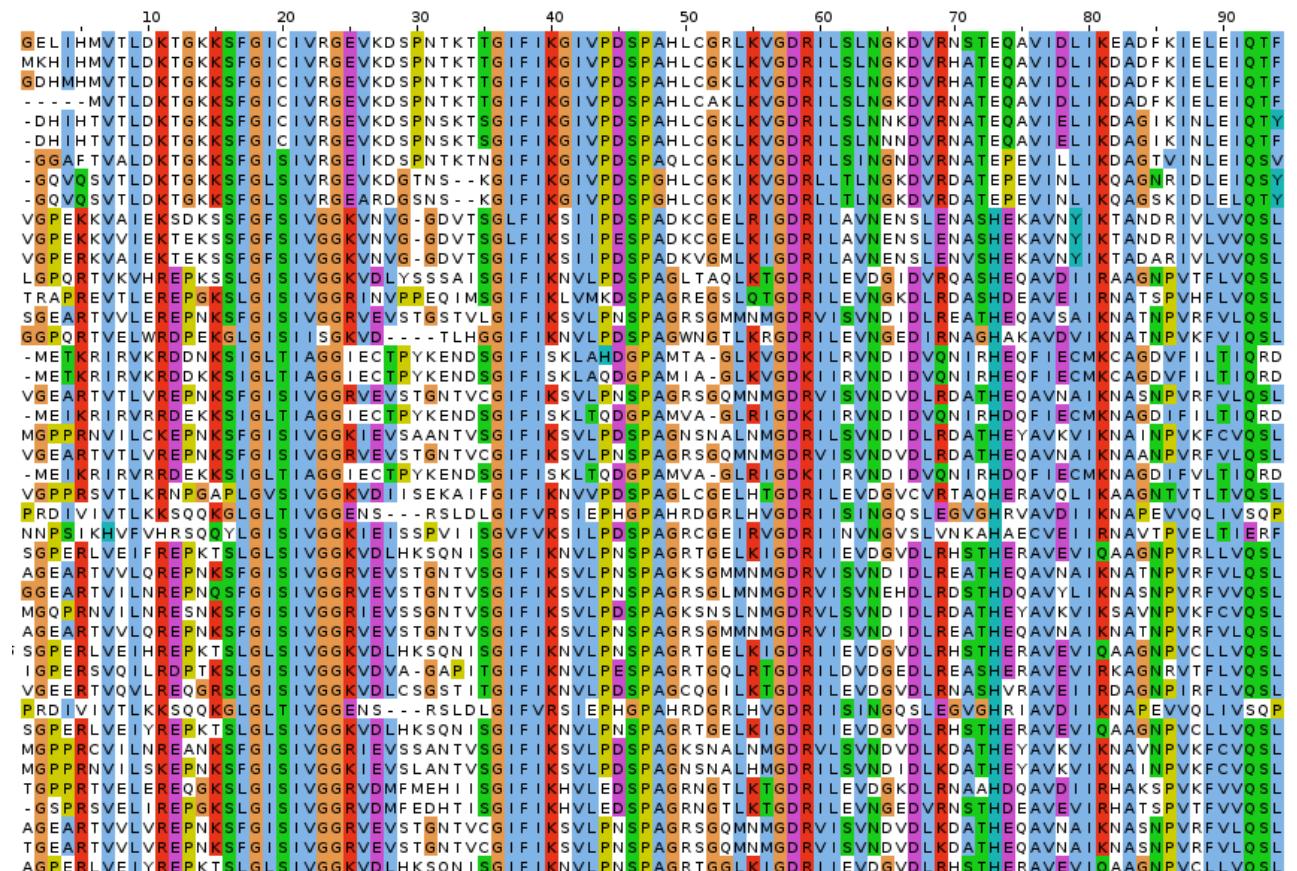


Figure 5.4 – L'alignement de notre sélection de séquences homologues à la protéine INAD (code PDB : 1IHJ)

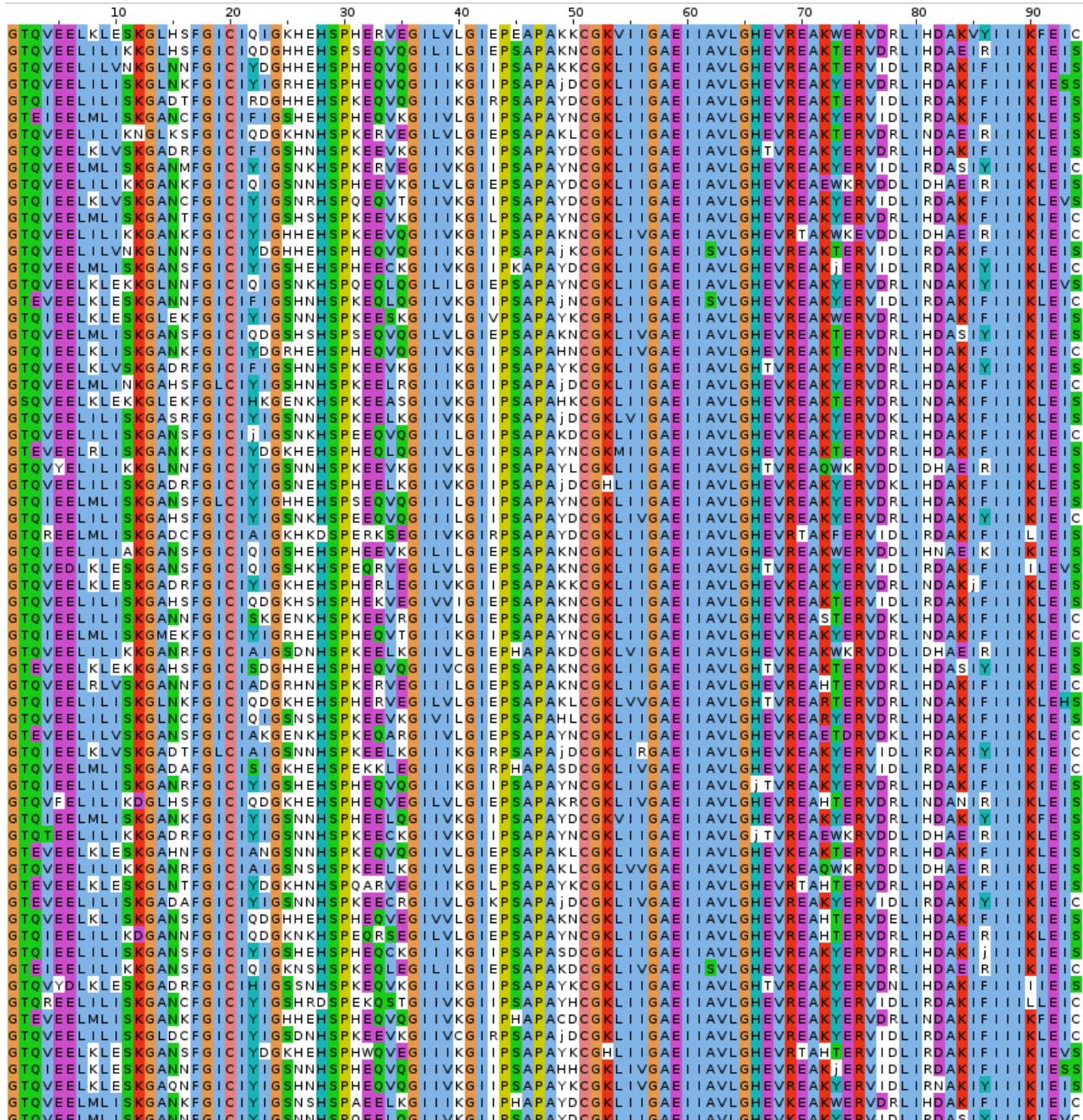


Figure 5.5 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine INAD (code PDB : 1IHJ), modèle NEA

5.4. Séquences expérimentales et modèles structuraux

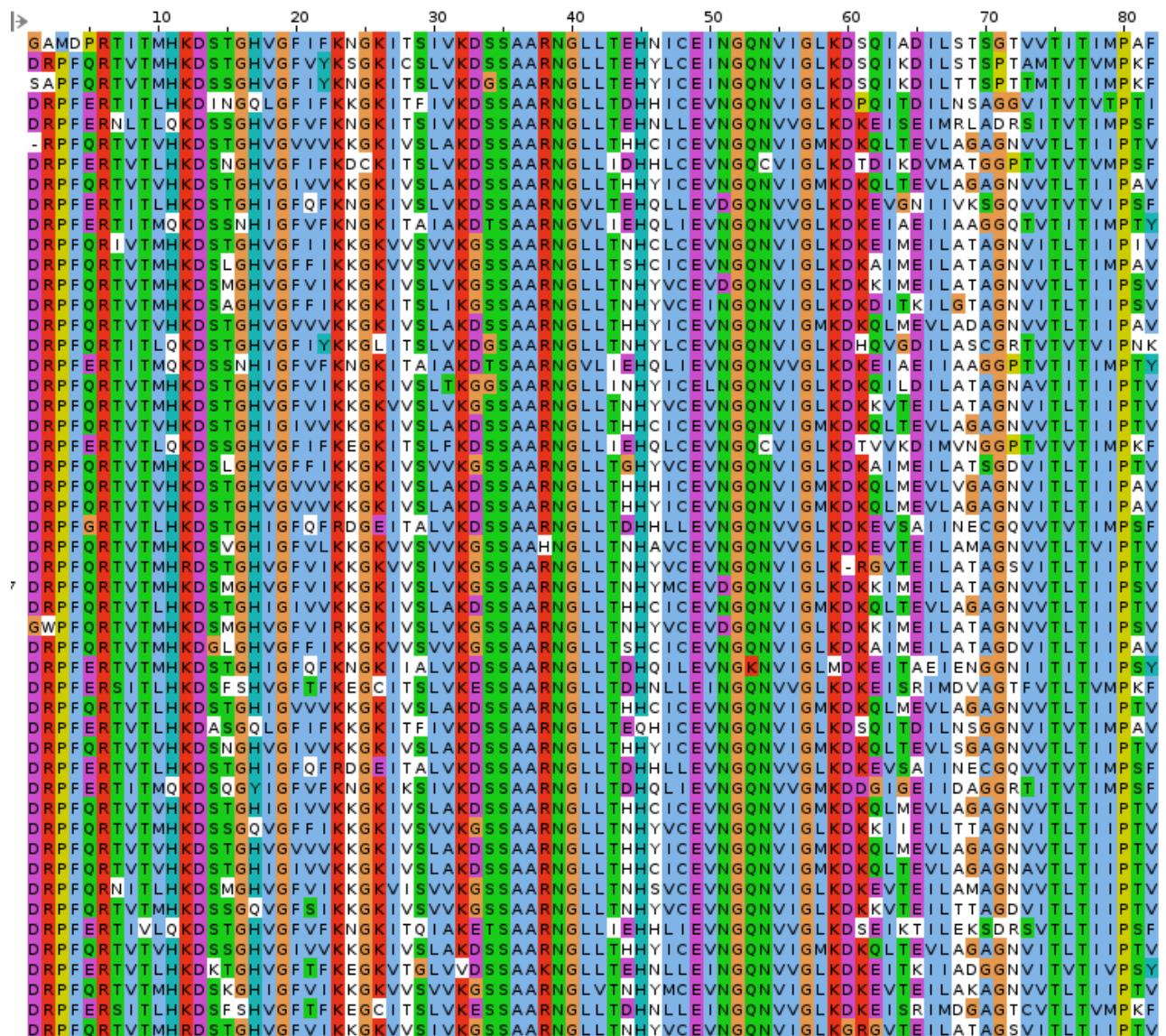


Figure 5.6 – L'alignement de notre sélection de séquences homologues à la protéine Syntenin (code PDB : 1R6J)

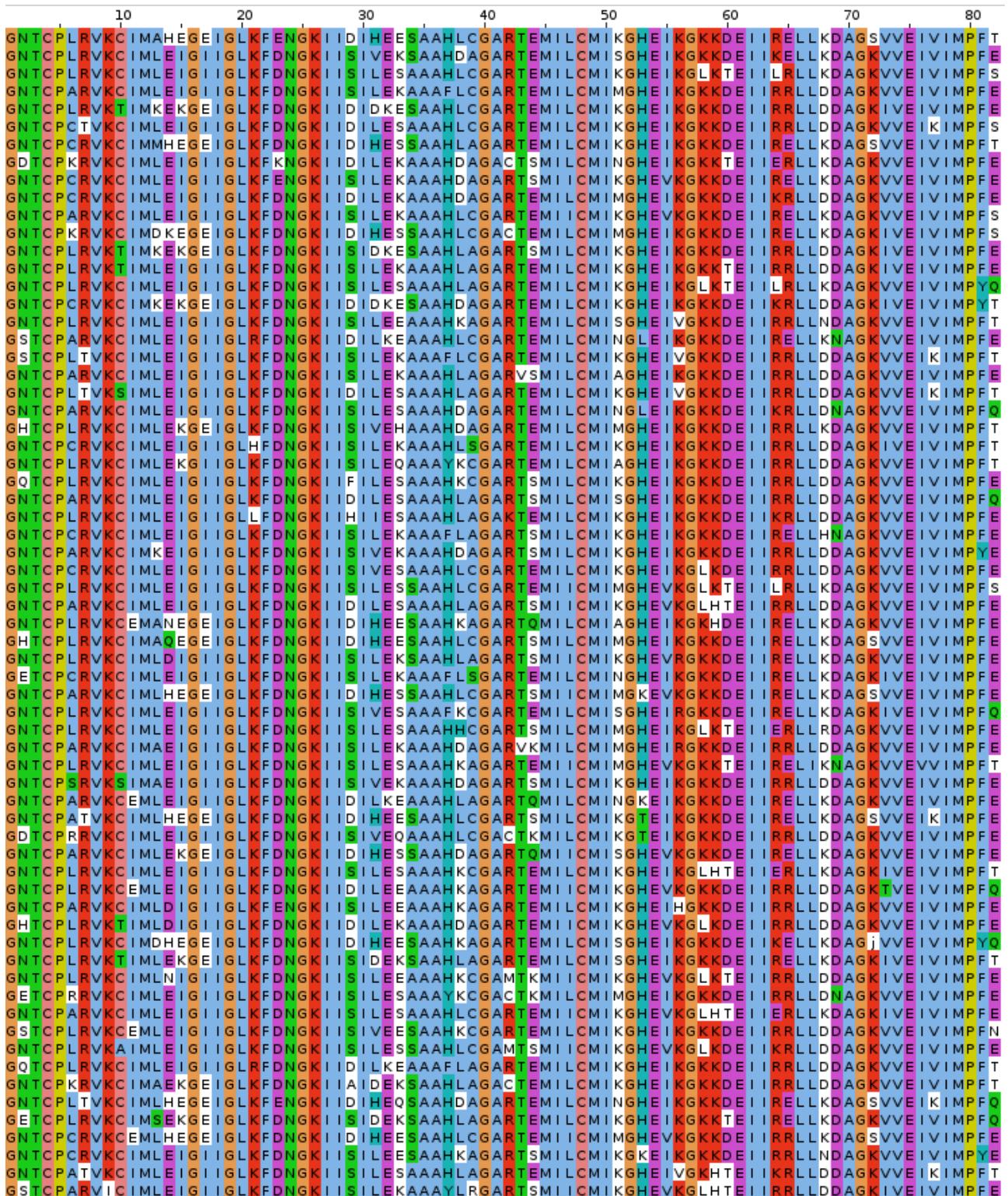


Figure 5.7 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine Syntenin (code PDB : 1R6J), modèle NEA

5.4. Séquences expérimentales et modèles structuraux

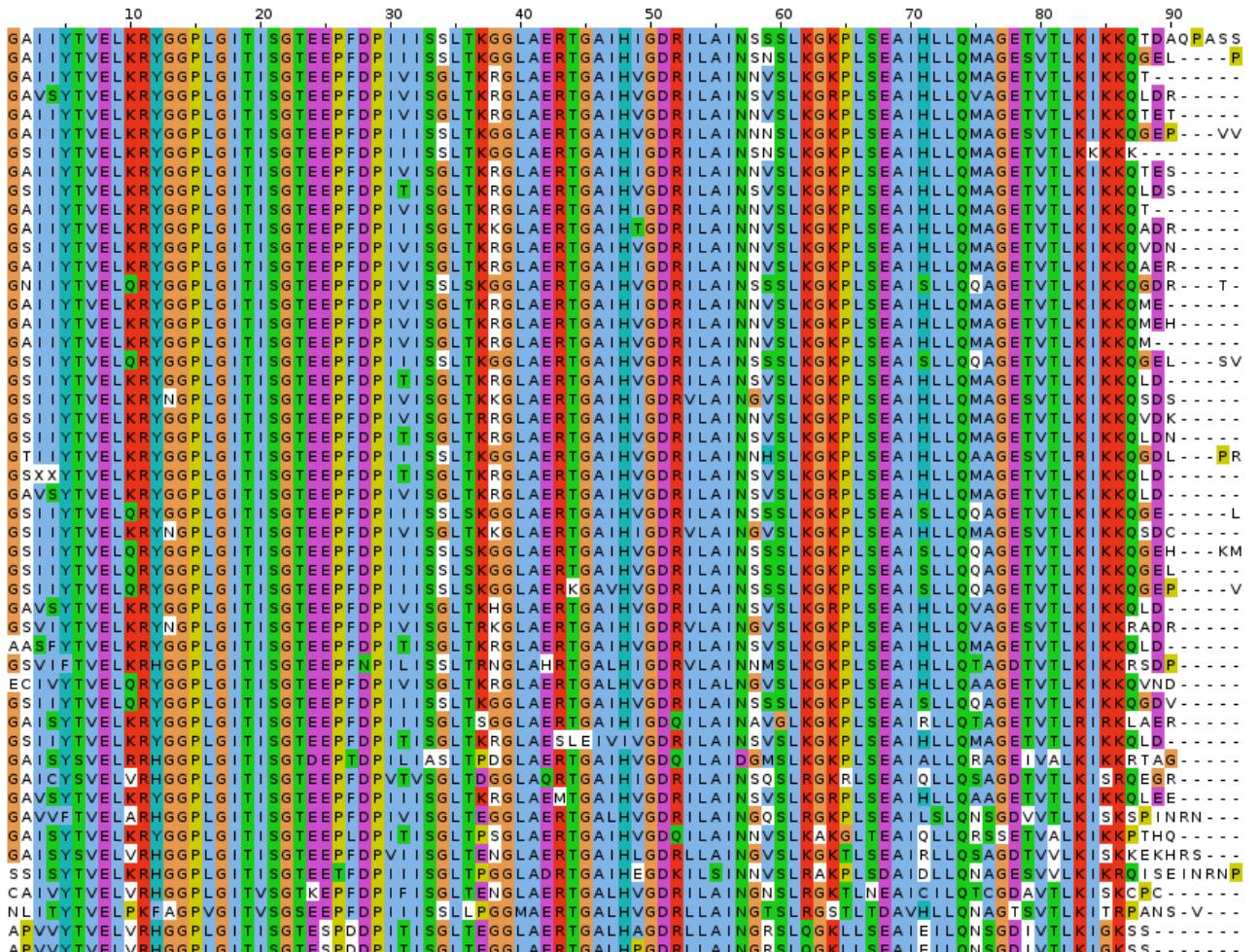


Figure 5.8 – L'alignement de notre sélection de séquences homologues à la protéine GRIP (code PDB : 1N7E)

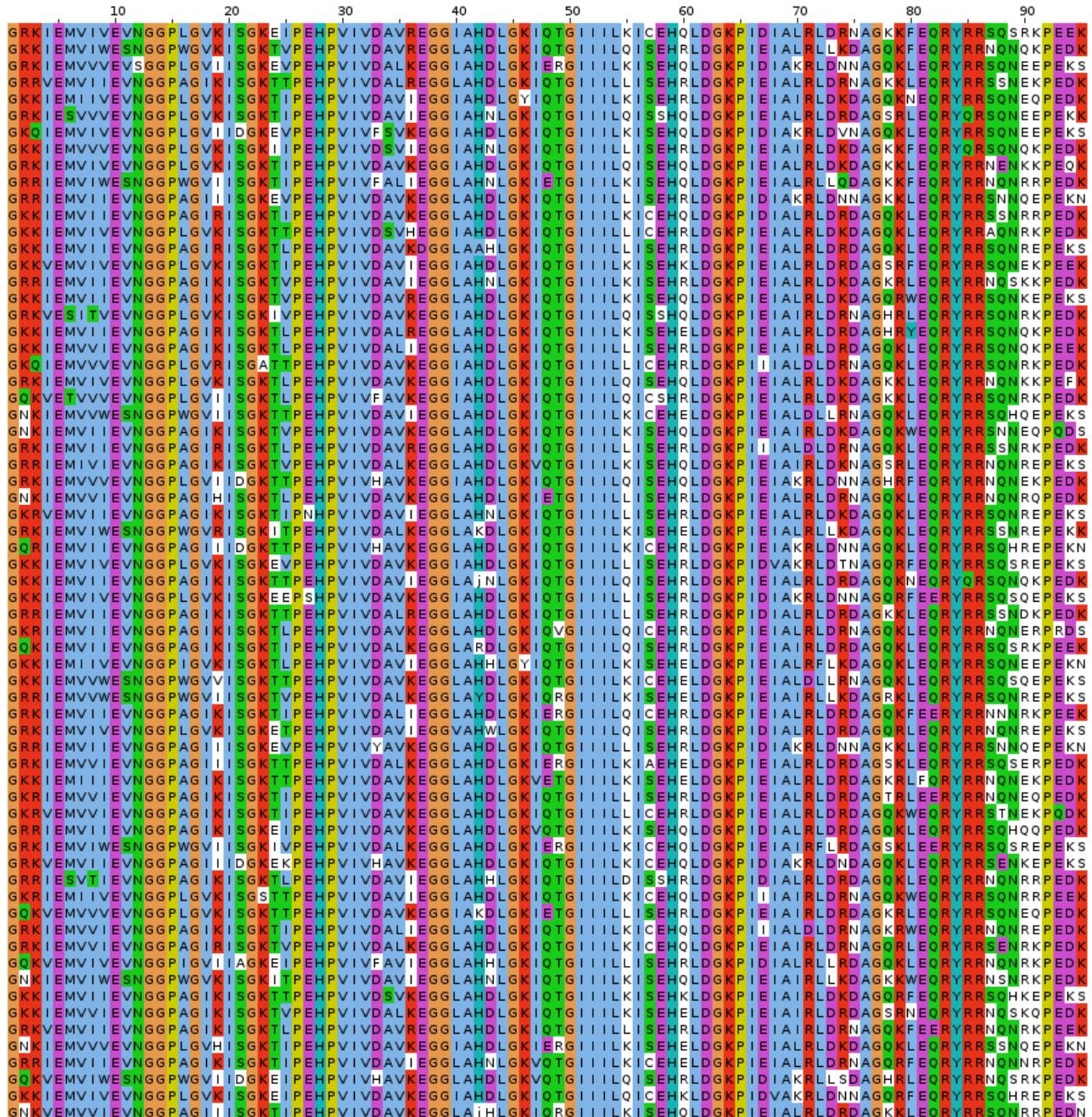


Figure 5.9 – Une sélection de séquences protéiques, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine GRIP (code PDB : 1N7E), modèle NEA

5.4. Séquences expérimentales et modèles structuraux

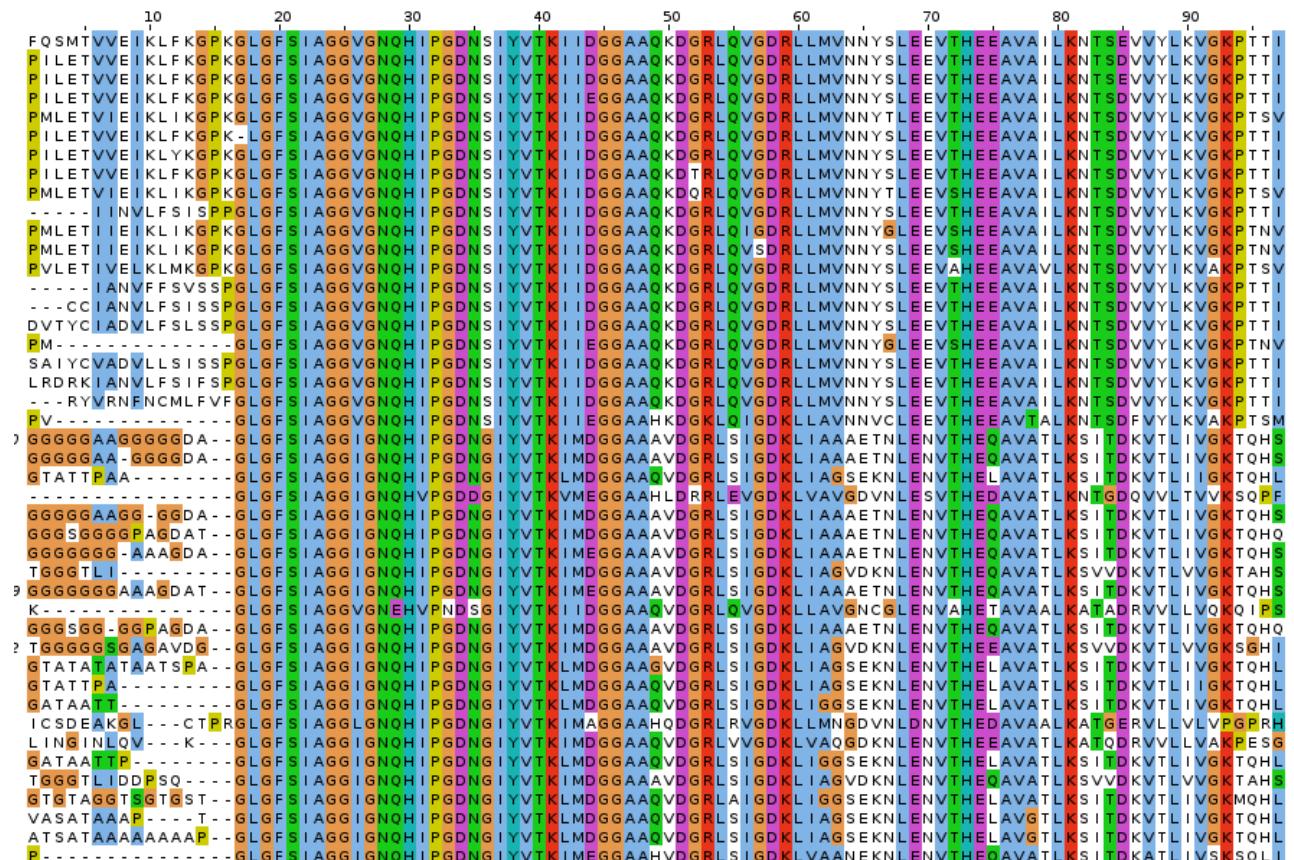


Figure 5.10 – L'alignement de notre sélection de séquences homologues à la protéine DLG2 (code PDB : 2BYG)

Chapitre 5. Application aux domaines PDZ

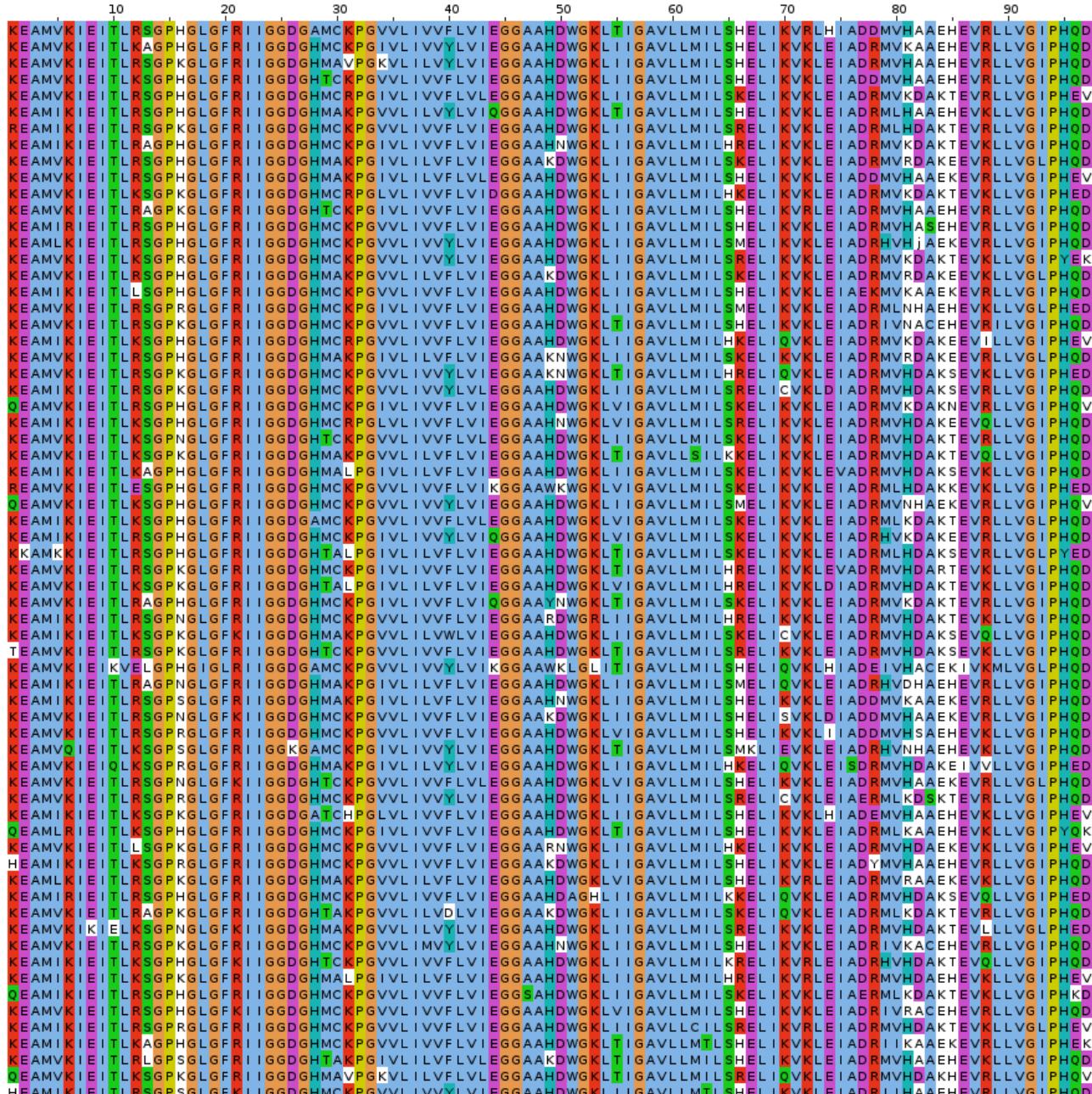


Figure 5.11 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine DLG2 (code PDB : 2BYG), modèle NEA

5.4. Séquences expérimentales et modèles structuraux

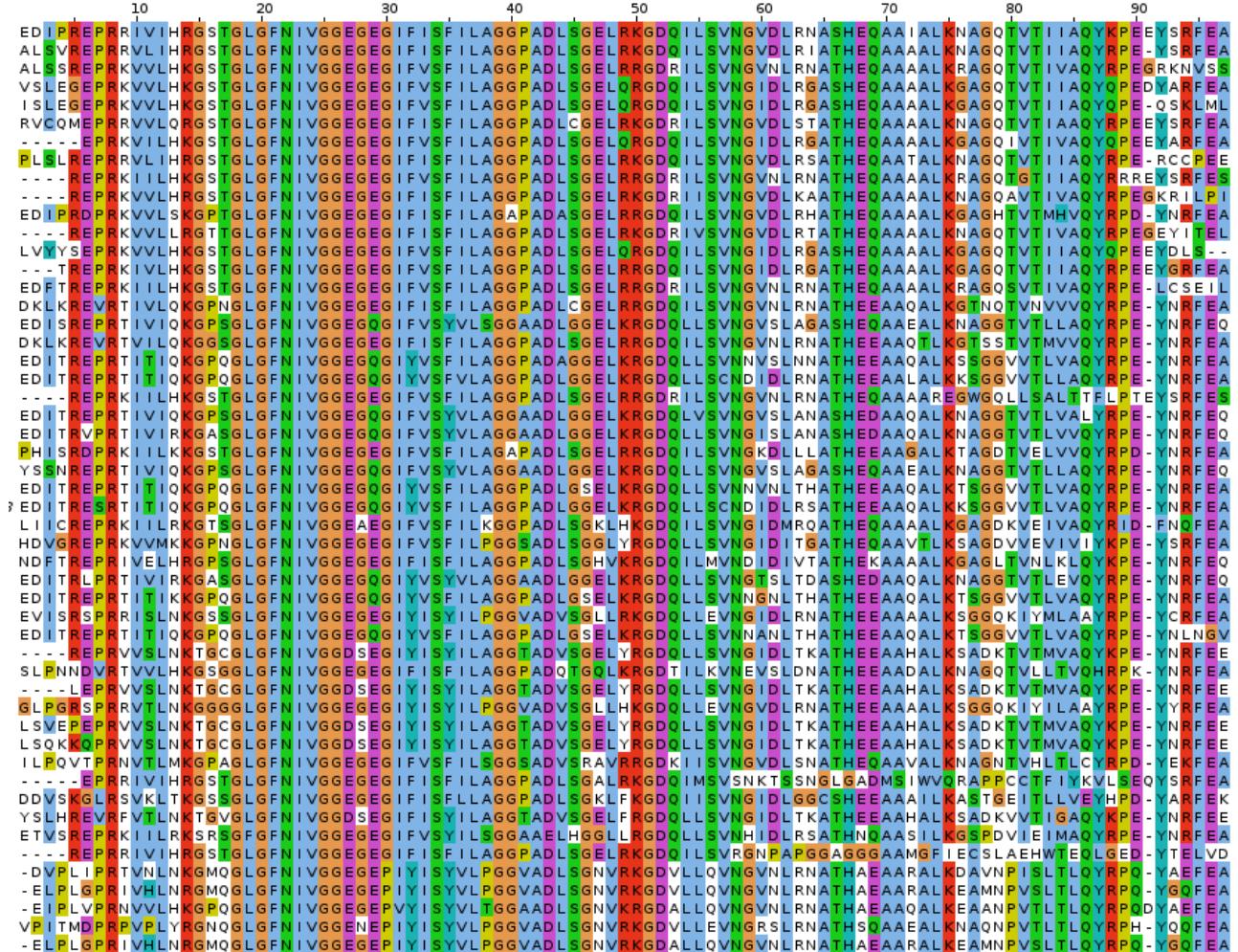


Figure 5.12 – L’alignement de notre sélection de séquences homologues à la protéine PSD95 (code PDB : 3K82)

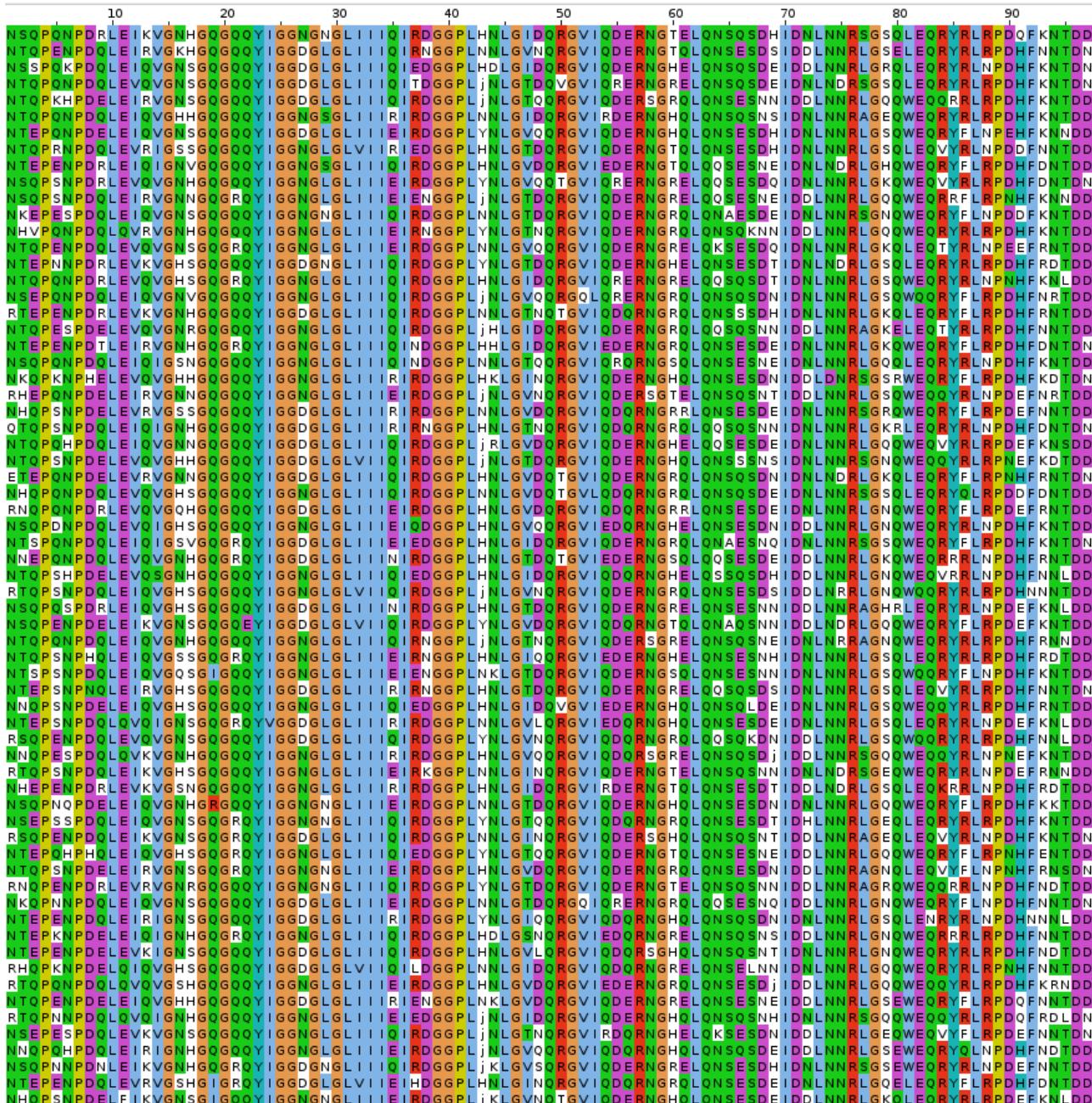


Figure 5.13 – Une sélection de séquences protéins, parmi les 10 000 séquences de meilleure énergie, obtenues avec le backbone de la protéine PSD95 (code PDB : 3K82), modèle NEA

5.4. Séquences expérimentales et modèles structuraux

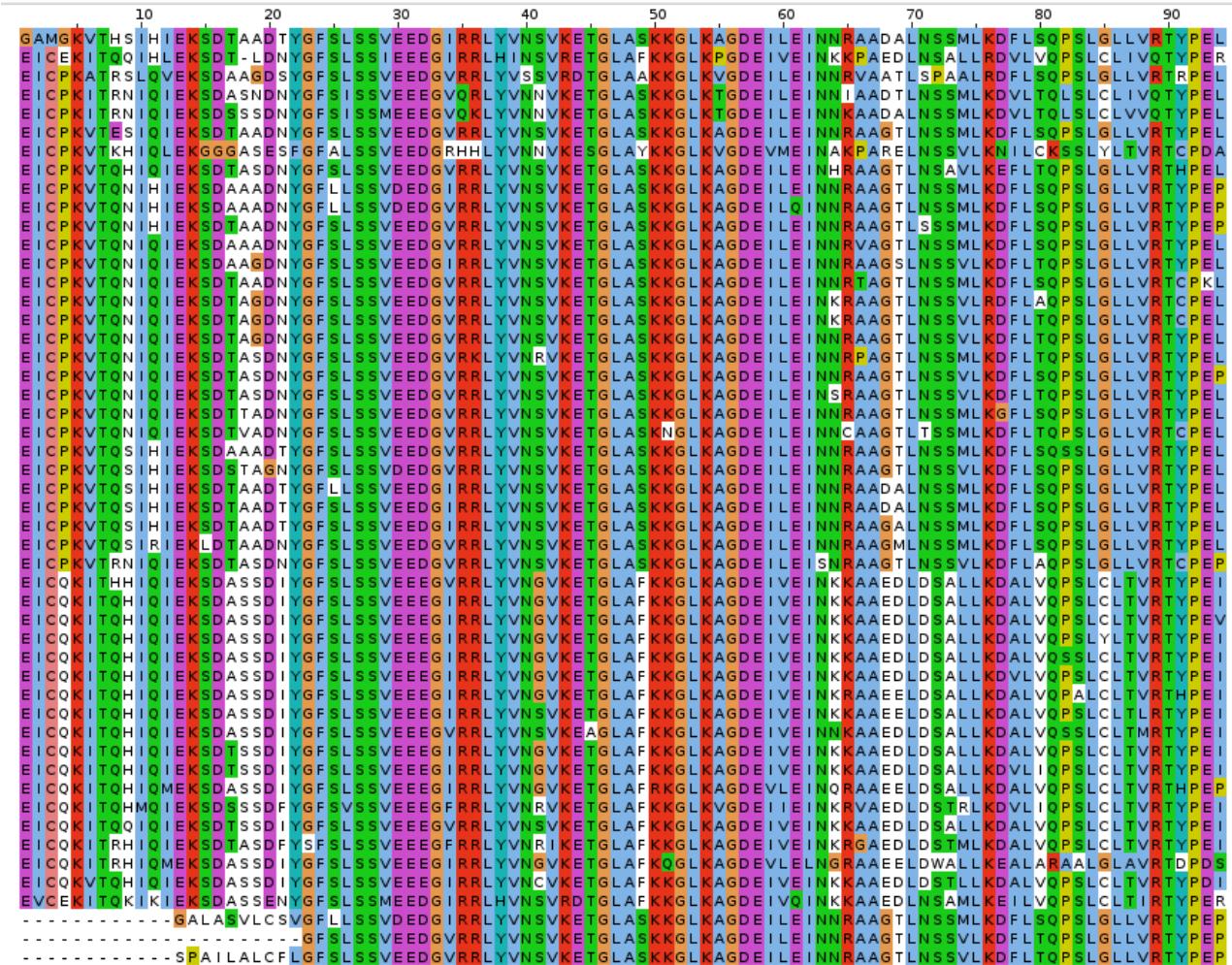


Figure 5.14 – L'alignement de notre sélection de séquences homologues à la protéine Tiam1 (code PDB)

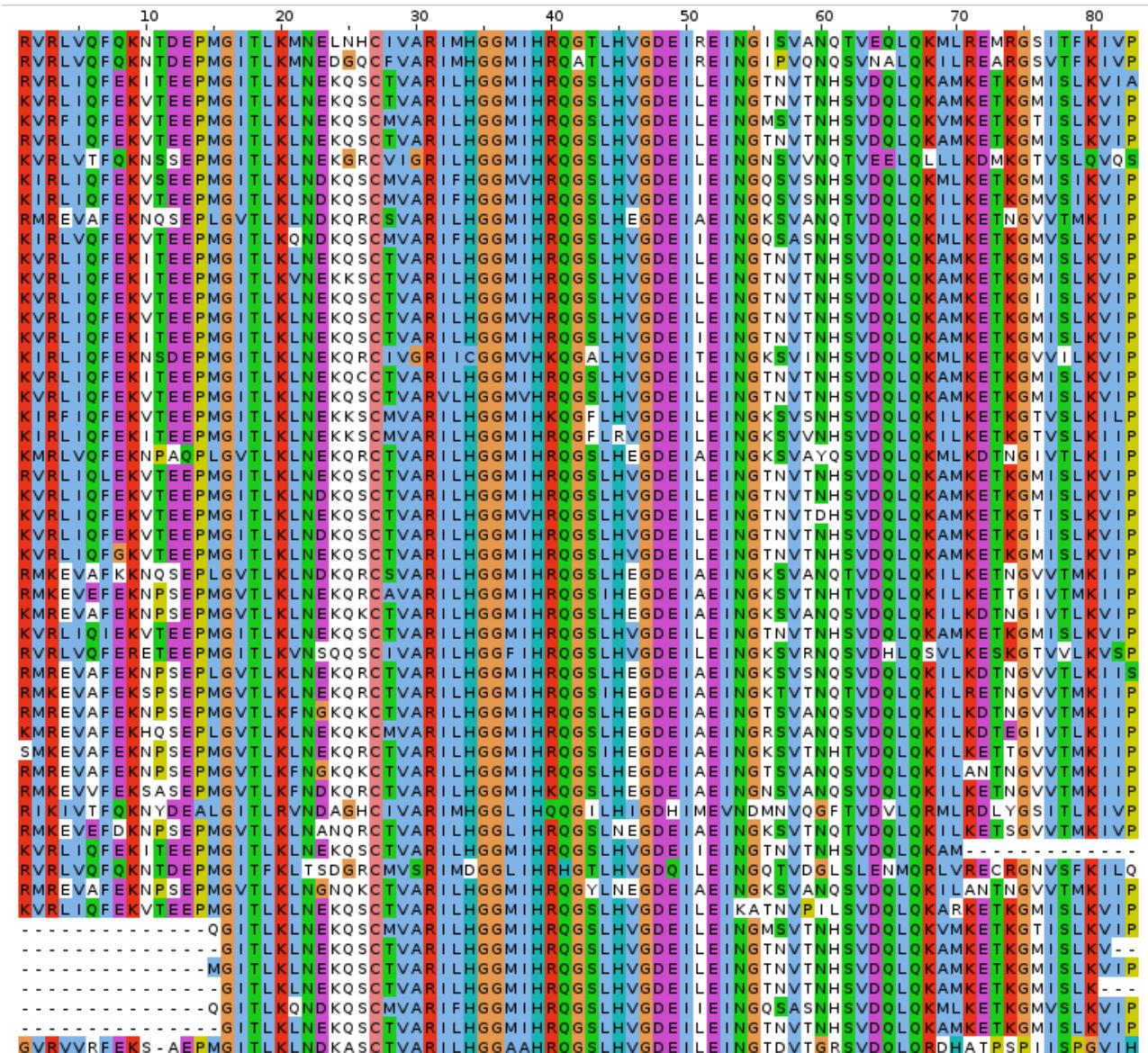


Figure 5.15 – L’alignement de notre sélection de séquences homologues à la protéine Cask (code PDB)

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 [113]. 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 [88, 79] et détaillée en 2.4.2.

5.5.2 Simulation Monte Carlo

La production des séquences est réalisée avec Proteus [90]. 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 [108]. Les simulations sont faites en utilisant la version 2015.38.58158 de la suite 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 [92, 93] qui tente de classer les séquences selon la base de données structurelle SCOP [59], 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

5.6. Résultats du modèle NEA

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 dans pour chaque classe, qui est calculés 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	Enfoui	Exposé	Enfoui	Exposé
ALA	10,9	4,6	11,1	4,4	17,0	12,0	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	(0,1)	7,3	(-1,4)	7,3
ASP	4,3	6,0	4,5	5,6	6,7	16,7	5,6	16,7
GLU	2,5	6,8	11,9	17,9	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	5,5	12,2	2,2	(0,0)	6,5	(1,8)
HIP	1,2	5,0	1,0	5,2	1,1	5,2	5,6	5,6
HIE	0,0	1,2	0,0	5,0	0,1	(-0,1)	0,4	(0,6)
HID	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
ILE	16,0	4,2	16,9	4,1	52,1	14,0	4,1	14,0
VAL	16,5	50,7	5,4	14,0	16,7	(1,4)	5,6	(0,0)
LEU	18,2	4,4	18,5	4,3	(1,4)	4,3	(0,0)	4,3
LYS	2,5	2,5	10,9	10,9	1,5	1,5	13,0	(2,1)
MET	0,9	0,9	1,5	1,5	1,6	(0,7)	1,4	1,4
ARG	2,8	2,8	8,7	8,7	2,5	2,5	6,1	6,1
SER	5,3	5,3	7,6	7,6	4,3	4,3	8,7	8,7
							(1,1)	(1,1)
PHE	4,1	4,1	2,4	2,4	4,5	4,6	2,1	2,1
TRP	0,0	0,0	0,0	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
GLY	0,8	0,9	3,1	4,9	0,0	0,0	0,0	0,0
							(-0,9)	(-4,9)
PRO	0,1	1,8			0,0	0,0	0,0	0,0

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 échantillonnées 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.

Protein	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

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 5.3, 5.5, 5.7, 5.9, 5.11 et 5.13. Les homologues naturels pour les huit protéines sont présentés aux figures 5.2, 5.4, 5.6, 5.8,

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 10^{-13}	10000	2,2 10^{-3}	10000
INAD	85/94	7,4 10^{-14}	10000	3,7 10^{-3}	10000
GRIP	84/95	2,2 10^{-10}	10000	1,2 10^{-3}	10000
Syntenin	76/82	7,3 10^{-13}	10 000	1,8 10^{-3}	10 000
DLG2	86/97	1,3 10^{-9}	10 000	9,6 10^{-4}	10 000
PSD95	90/97	3,7 10^{-23}	10 000	5,2 10^{-4}	10 000

5.10, 5.12, 5.14 et 5.15. Comme dans de nombreuses études de CPD antérieures [114, 69], 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 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 atteindre les mêmes niveaux de diversité que l'ensemble Seed, construit pour caractériser la diversité des domaines PDZ et notamment la diversité de leur squelette.

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 5.16). 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 5.17 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

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

Protein	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
6prots	2,42	2,88	
CASK	1,55	1,68	3,15
TIAM1	1,22	1,57	3,15

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 [49]. 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 générerons des séquences Tiam1 et Cask qui sont alors soumises aux tests Superfamily et aux calculs de similarité. La performance de Tiam1 sur la superfamille 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 superfamille, 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 générerons 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 superfamille et de $3,8 \cdot 10^{-10}$ à $8,0 \cdot 10^{-9}$ pour DLG2.

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 Syntenine, nous calculons également les scores de similarité en utilisant les deux modèles. Les scores

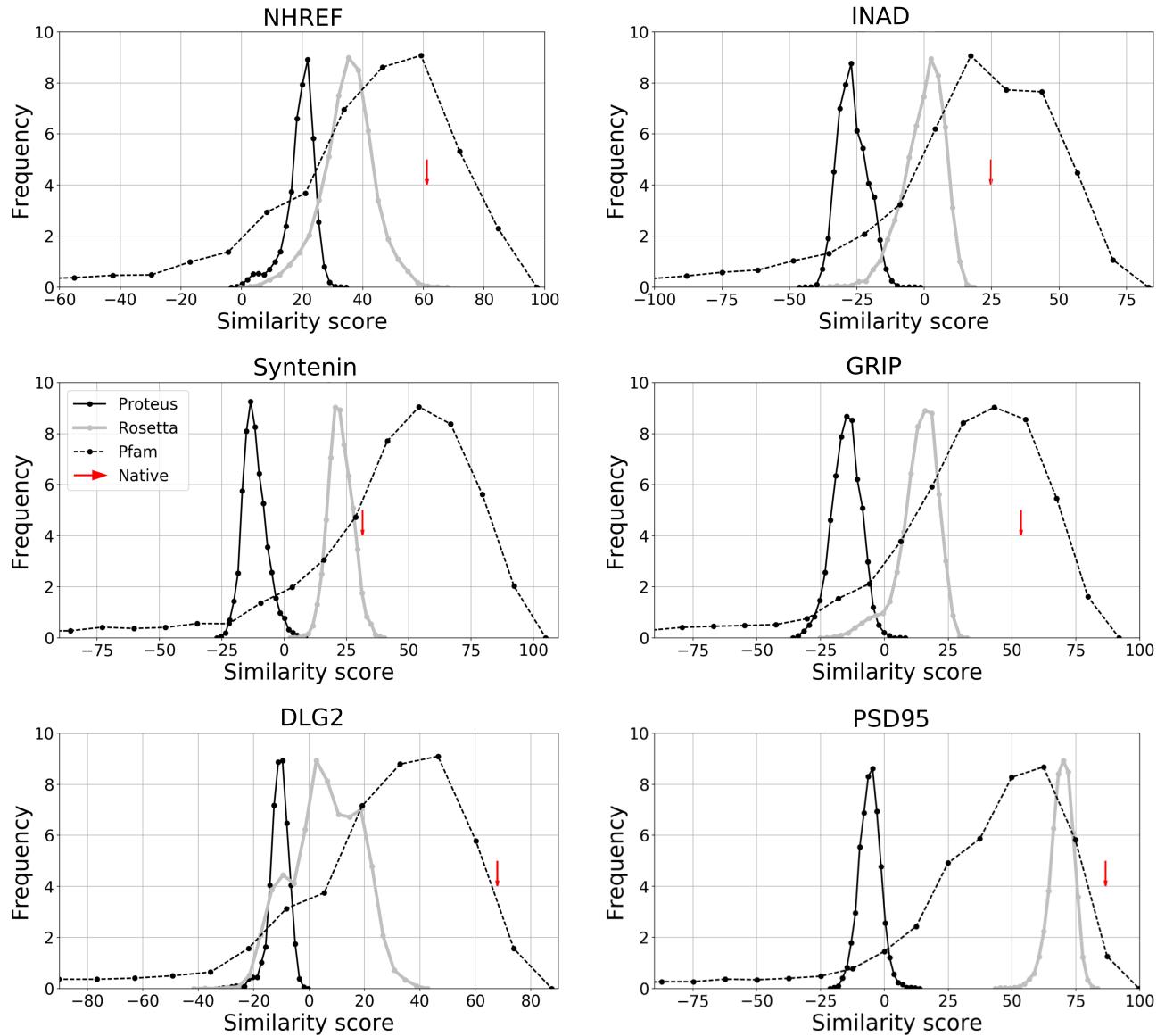


Figure 5.16 – Similarité des séquences des 6 protéines produites par Proteus et Rosetta à l’alignement Pfam RP55, sur l’ensemble des positions.

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.

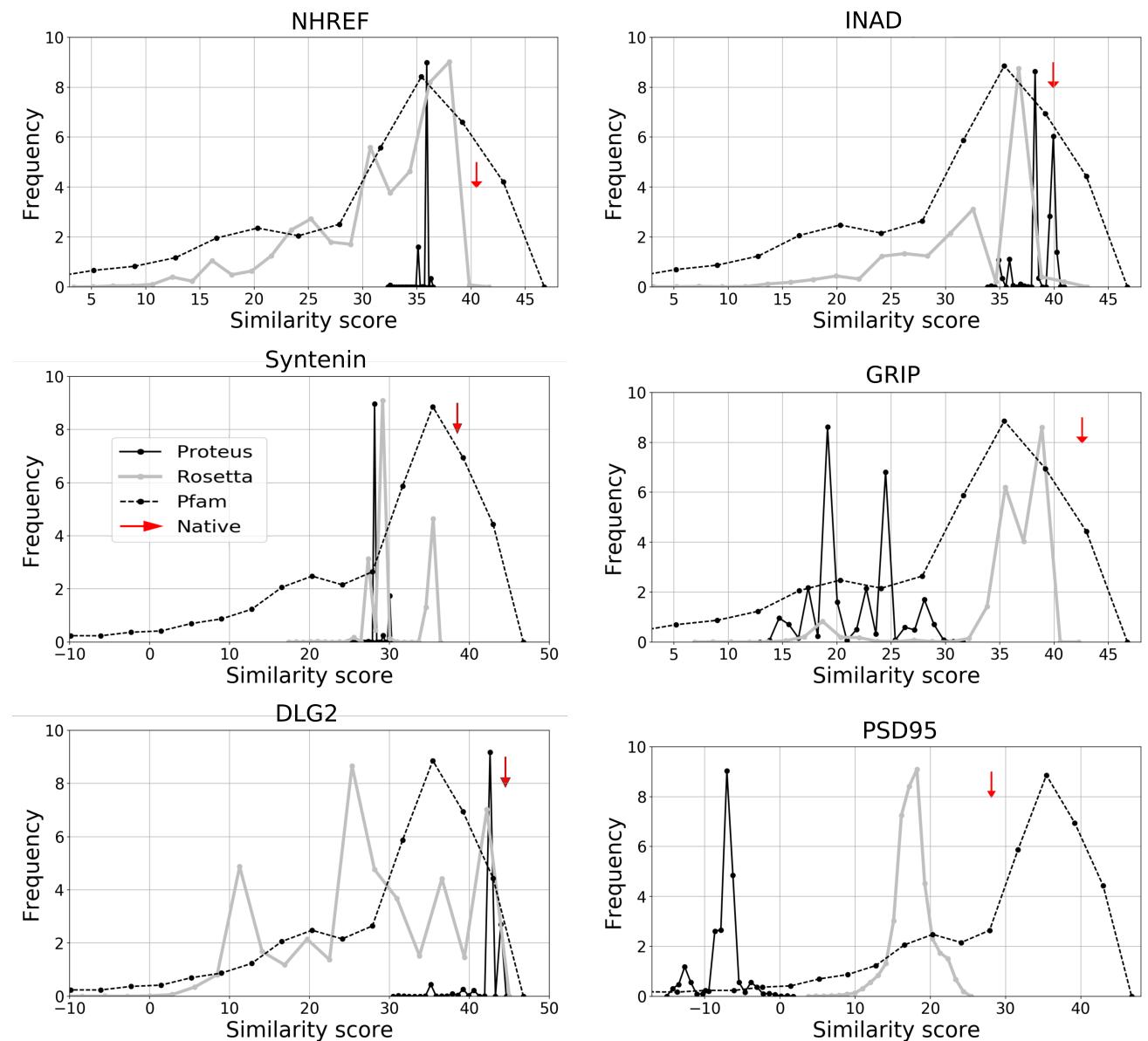


Figure 5.17 – Similarité des séquences des 6 protéines produites par proteus modèle NEA et Rosetta à l’alignement Pfam RP55, sur les positions du cœur hydrophobe.

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 ce variant, 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 générions 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 échantillonnées 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 superfamille 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 superfamille.

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	0,6	11,3
THR	6,2		7,7		4,7	[-0,3]	8,9	[2,3]
SER	4,4	4,4	6,7	6,7	5,2	5,2 [-0,8]	7,9	7,9 [-1,2]
ASP	4,8		6,1		5,8	8,6	8,1	20,6
GLU	2,6	7,4	11,0	17,1	2,8	[-1,2]	12,5	[-3,5]
ASN	3,4		6,9		4,2	6,0	8,8	16,0
GLN	1,7	5,1	5,8	12,7	1,8	[-0,9]	7,2	[-3,3]
HIP	2,0		5,9		0,0		0,4	
HIE	0,0	2,0	0,0	5,9	0,5	0,6 [1,4]	2,7	5,0 [0,9]
HID	0,0		0,0		0,1		1,9	
ILE	12,4		4,1		11,2		0,6	
VAL	21,1	50,3	5,1	14,0	21,2	49,4 [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	1,0 [0,3]	2,2	2,2 [-0,4]
LYS	3,4	3,4	10,8	10,8	4,2	4,2 [-0,8]	12,4	12,4 [-1,6]
ARG	1,1	1,1	9,8	9,8	1,8	1,8 [-0,7]	11,8	11,8 [-2,0]
PHE	4,6		2,4		3,9	3,9	0,0	0,0
TRP	0,0	4,6	0,1	2,5	0,0	0,0 [0,7]	0,0	[2,5]
TYR	2,4	2,4	1,2	1,2	2,0	2,0 [0,4]	0,0	0,0 [1,2]
GLY	1,0		1,9		0,0	0,0	0,0	0,0
PRO	0,1	1,1	1,8	3,7	0,0	0,0 [1,1]	0,0	[3,7]

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.
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

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 sur DLG2, le NEA étant au-dessus pour les deux autres. Il n'y a donc pas d'amélioration

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	10 000	9,6 10 ⁻⁴	10 000
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).

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 5.18.

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). Il s'avère donc que les séquences Rosetta sont nettement plus proches des séquences natives que celles de Proteus, avec sept mutations de moins en moyenne.

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

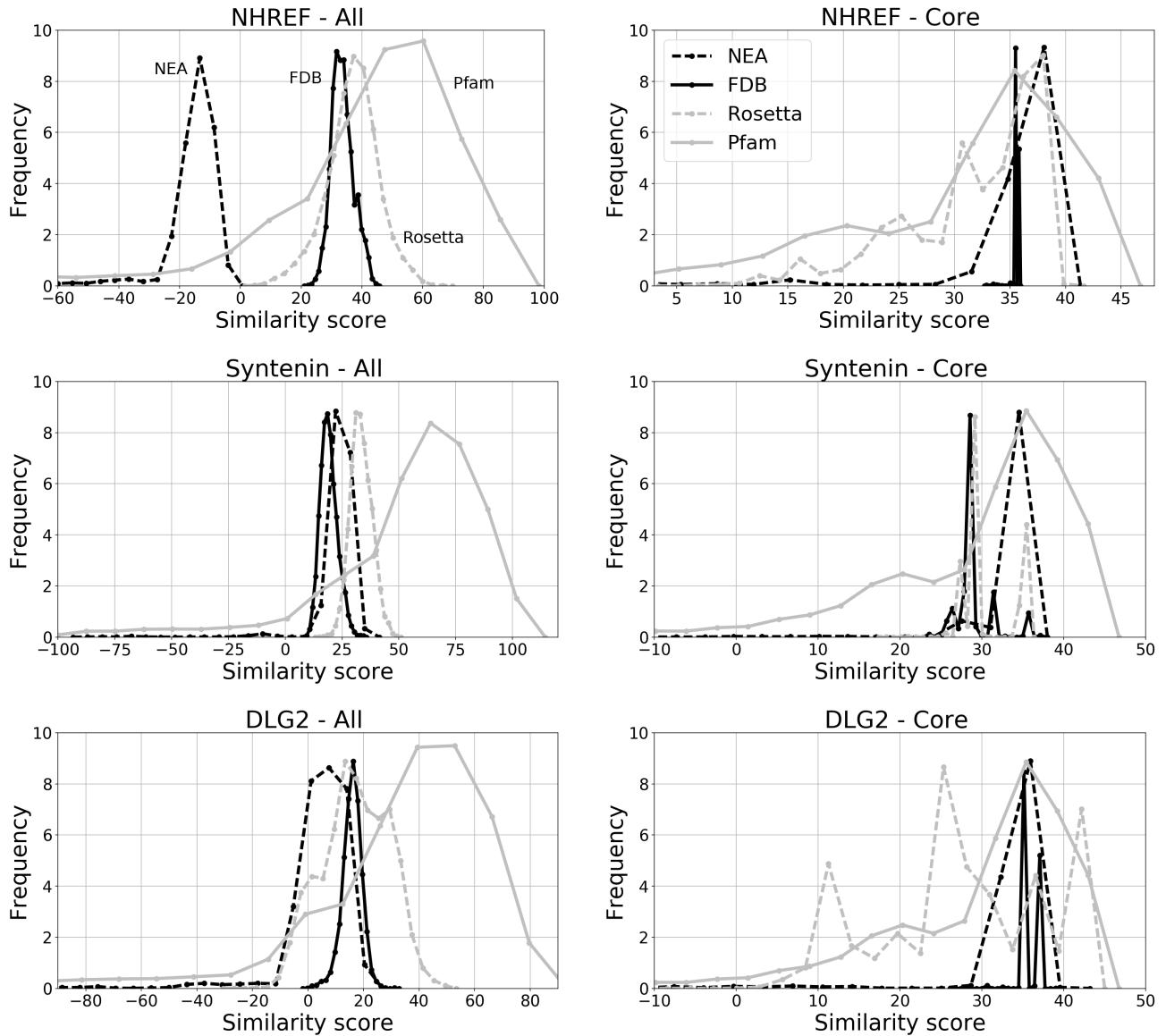


Figure 5.18 – Similarité des séquences Proteus (NEA et FDB), Rosetta et des séquences de l’alignement Pfam RP55, sur toutes les positions à gauche et sur les positions du cœur à droite.

5.7.5 Logos des séquences obtenues

Finalement, nous montrons les séquences obtenues. Elles sont représentées sous forme de logos à la figure 5.19 pour les positions du cœur hydrophobe, et la figure 5.20 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.

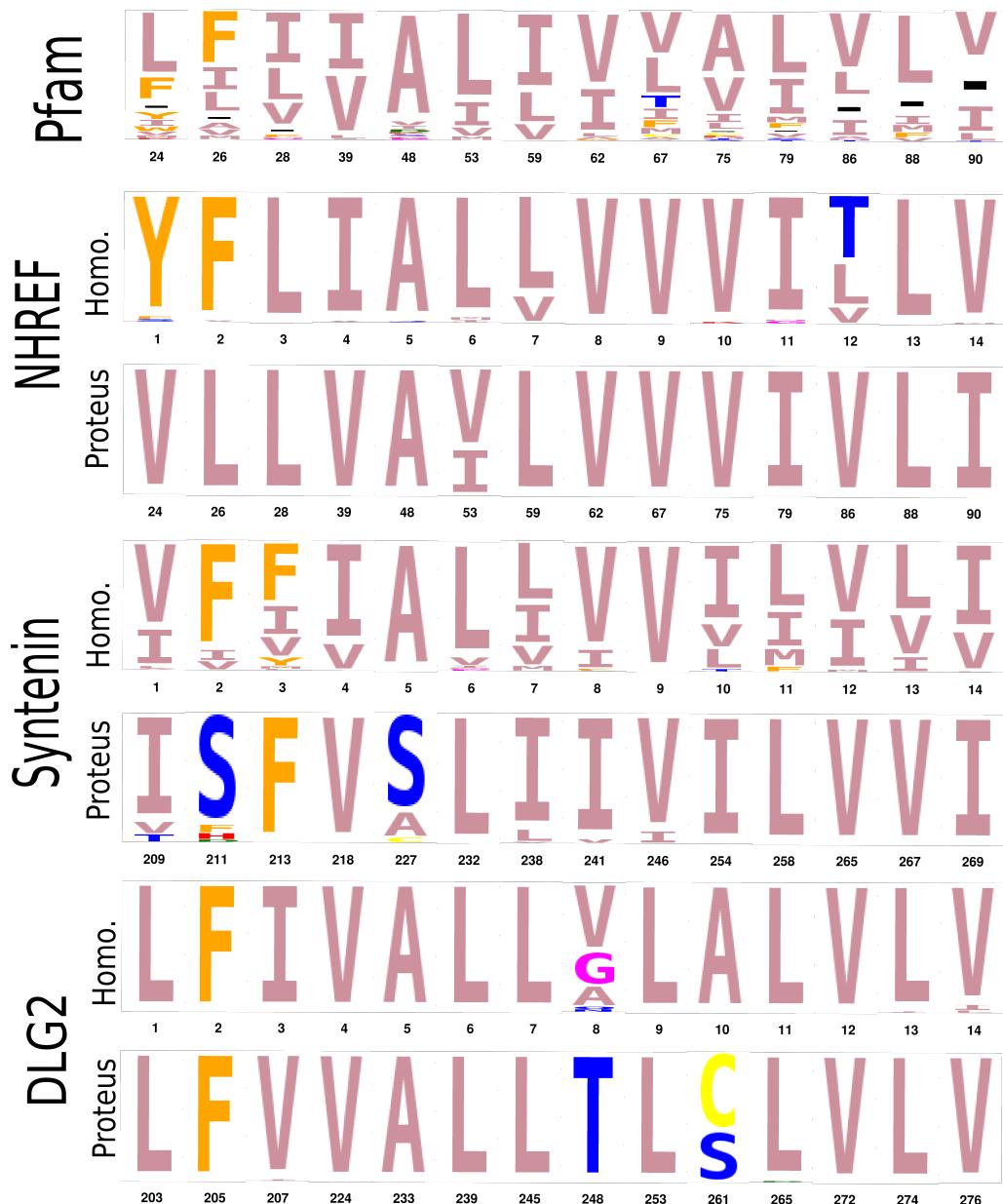


Figure 5.19 – Les séquences conçues par Proteus, les homologues à la native et les séquences naturelles représentées sous forme de logo pour les positions du cœur hydrophobe



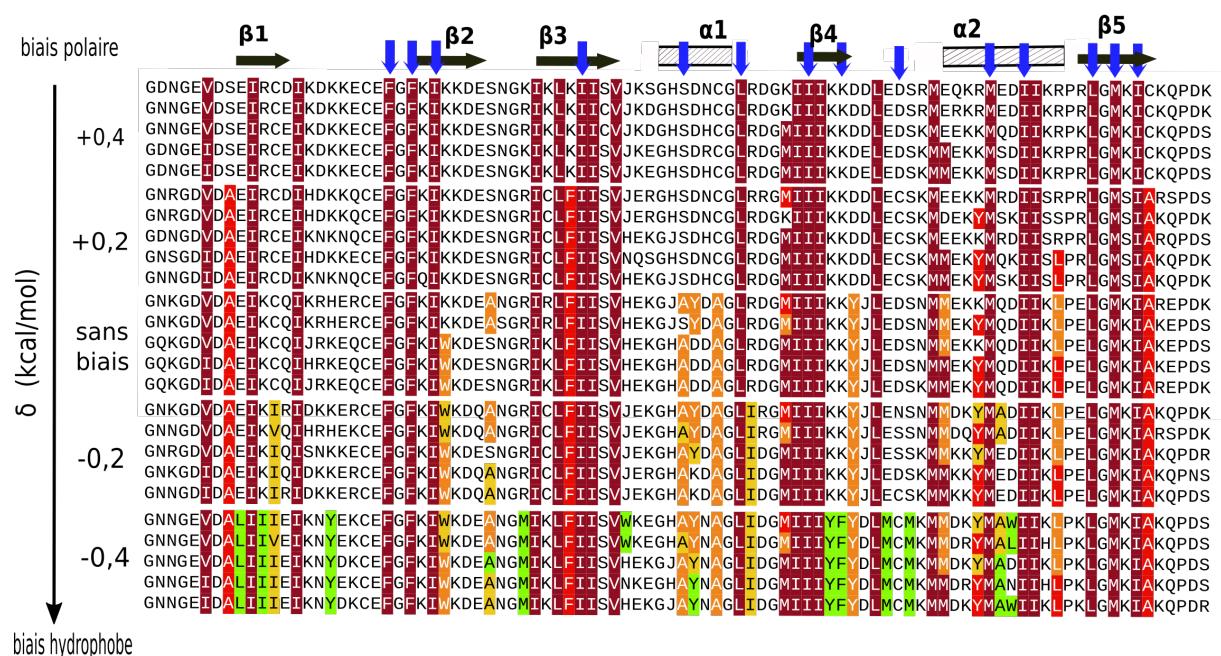
Figure 5.20 – Les séquences conçues par Proteus, les homologues à la native et les séquences naturelles représentées sous forme de logo pour les positions exposées

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 d'énergie 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.

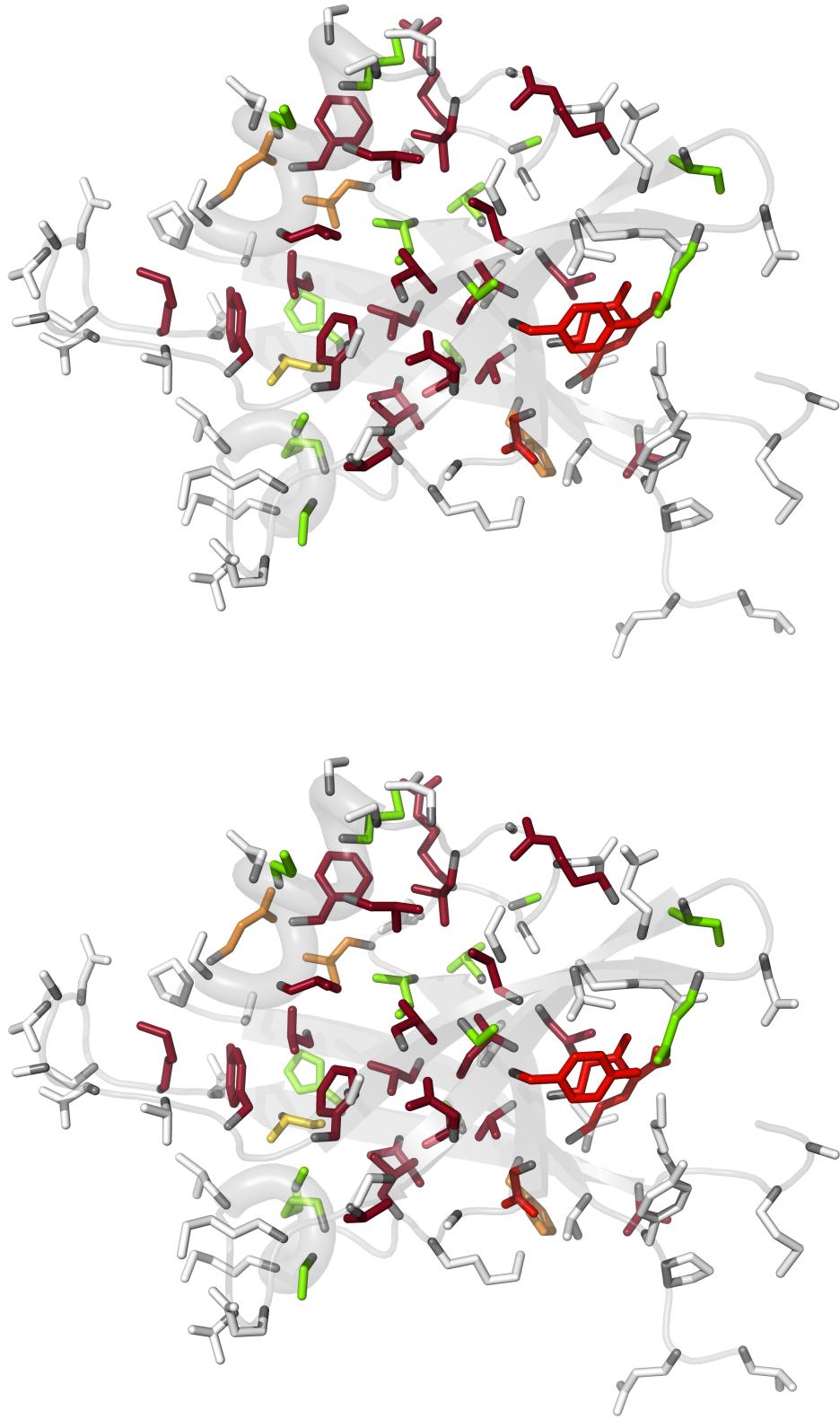
5.8. Application : Croissance du noyau hydrophobe

Figure 5.21 – 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 5.22 – 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.



5.8. Application : Croissance du noyau hydrophobe

Les résultats pour Tiam1 sont présentés à la figure 5.21 et à la figure 5.22. 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 5.24.

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.

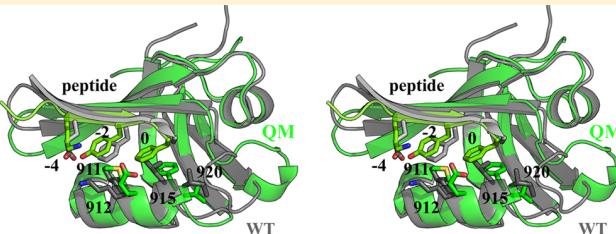
Computational Design of the Tiam1 PDZ Domain and Its Ligand Binding

David Mignon,^{§,†} Nicolas Panel,^{§,†,ID} Xingyu Chen,[†] Ernesto J. Fuentes,[‡] and Thomas Simonson^{*,†,ID}

[†]Laboratoire de Biochimie (CNRS UMR7654), Ecole Polytechnique, Palaiseau, France

[‡]Department of Biochemistry, Roy J. & Lucille A. Carver College of Medicine and Holden Comprehensive Cancer Center, University of Iowa, Iowa City, Iowa 52242-1109, United States

[§] Supporting Information



ABSTRACT: PDZ domains direct protein–protein interactions and serve as models for protein design. Here, we optimized a protein design energy function for the Tiam1 and Cask PDZ domains that combines a molecular mechanics energy, Generalized Born solvent, and an empirical unfolded state model. Designed sequences were recognized as PDZ domains by the Superfamily fold recognition tool and had similarity scores comparable to natural PDZ sequences. The optimized model was used to redesign the two PDZ domains, by gradually varying the chemical potential of hydrophobic amino acids; the tendency of each position to lose or gain a hydrophobic character represents a novel hydrophobicity index. We also redesigned four positions in the Tiam1 PDZ domain involved in peptide binding specificity. The calculated affinity differences between designed variants reproduced experimental data and suggest substitutions with altered specificities.

1. INTRODUCTION

PDZ domains (“Postsynaptic density-95/Discs large/Zonula occludens-1”) are small, globular protein domains that establish protein–protein interaction networks in the cell.^{1–6} They form specific interactions with other, target proteins, usually by recognizing a few amino acids at the target C-terminus. Because of their biological importance, PDZ domains and their interaction with target proteins have been extensively studied and computationally engineered. Peptide ligands have been designed that modulate the activity of PDZ domains involved in various pathologies.^{7–9} Engineered PDZ domains and PDZ ligands have been used to elucidate principles of protein folding and evolution.^{10–13} In addition, these small domains with their peptide ligands provide benchmarks to test the computational methods themselves.^{14–16}

An emerging method that has been applied to several PDZ domains is computational protein design (CPD).^{17–22} Starting from a three-dimensional (3D) structural model, CPD explores a large space of amino acid sequences and conformations to identify protein variants that have predefined properties, such as stability or ligand binding. Conformational space is usually defined by a discrete or continuous library of side chain rotamers and by a finite set of backbone conformations or a specific repertoire of allowed backbone deformations. The energy function that drives CPD usually combines physical and

empirical terms,^{23–25} while the solvent and the protein unfolded state are described implicitly.⁴¹

Here, we considered a simple but important class of CPD models. The energy is a physics-based function of the “MMGBSA” type, which combines a molecular mechanics protein energy with a Generalized Born + surface area implicit solvent. The folded protein is represented by a single, fixed, backbone conformation and a discrete side chain rotamer library. The unfolded state energy depends only on sequence composition, not an explicit structural model. The main adjustable model parameters are the protein dielectric constant ϵ_p , a small set of atomic surface energy coefficients σ_b , and a collection of amino acid chemical potentials, or “reference energies” E_t^r . Each surface coefficient measures the preference of a particular atom type to be solvent-exposed, while each reference energy represents the contribution of a single amino acid of type t to the unfolded state energy. The model is implemented in the Proteus software.^{26–28}

The present physics-based energy function can be compared to more empirical ones, of which the most successful is the Rosetta energy function.^{29–31} The Rosetta function includes a Lennard-Jones repulsion term, a Coulomb term, a hydrogen-bonding term, a Lazaridis–Karplus solvation term,³² and

Received: December 28, 2016

Published: April 10, 2017

63 unfolded state reference energies. It has a large number of
 64 parameters specifically optimized for CPD, which provide
 65 optimal performance, but less transferability and a less
 66 transparent physical interpretation. Proteus also provides
 67 some specific functionalities, such as Replica Exchange Monte
 68 Carlo, various importance sampling methods, and the ability to
 69 compute free energies that are formally exact.^{33–35}

70 We optimized the reference energies E_t^r for the Tiam1 and
 71 Cask PDZ proteins, using a maximum likelihood formalism. We
 72 compared two values of the protein dielectric constant, $\epsilon_p = 4$
 73 and 8. These values gave good results in a systematic study that
 74 compared dielectric constants in the range 1–32.³⁶ The
 75 performance of the model was tested by generating designed
 76 sequences for both proteins and comparing them to natural
 77 sequences, as well as sequences generated with the Rosetta
 78 energy function and software.³⁷ The sequence design was
 79 performed by running long Monte Carlo simulations in which
 80 all protein positions except Gly and Pro were allowed to mutate
 81 freely, leading to thousands of designed protein variants. The
 82 testing included cross-validation, for which the reference
 83 energies were optimized using one set of PDZ domains, then
 84 applied to others. We also performed 100–1000 ns molecular
 85 dynamics (MD) simulations for a few of the sequences
 86 designed with our optimized CPD model, to help assess their
 87 stability. Ten sequences were stable over 100 ns or more and
 88 one over 1000 ns of MD simulation.

89 We then applied the CPD model with optimized parameters
 90 to two problems, which are representative of the two main
 91 areas we are interested in exploring: the plasticity of sequence
 92 space for PDZ domains and designing strong and specific PDZ
 93 ligands. Earlier applications in these areas mostly employed
 94 empirical, knowledge-based energy functions such as the
 95 Rosetta function.^{9,10,13,14} First, we performed a series of
 96 Monte Carlo simulations of two PDZ domains where the
 97 chemical potential of the hydrophobic amino acid types was
 98 gradually increased, artificially biasing the protein composition.
 99 As the hydrophobic bias was increased, hydrophobic amino
 100 acids gradually invaded the protein from the inside out, forming
 101 a hydrophobic core that became larger than the natural one.
 102 The propensity of each core position to become hydrophobic at
 103 a high or low level of bias can be seen as a structure-dependent
 104 hydrophobicity index, which provides information on the
 105 designability or plasticity of the protein core. The second
 106 application consisted in designing four Tiam1 positions known
 107 to be involved in specific target recognition. These four
 108 positions were varied through Monte Carlo simulations of
 109 either the apoprotein or the protein in complex with two
 110 distinct peptide ligands. The simulations were in agreement
 111 with experimental sequences and binding affinities, and suggest
 112 new variants that could have altered specificities. This
 113 application is a step toward the design of strong peptide
 114 binders, which could be of use as reagents or inhibitors *in vitro*
 115 or *in vivo*.

2. THE UNFOLDED STATE MODEL

116 **2.1. Maximum Likelihood Reference Energies.** The
 117 Monte Carlo method employed here generates a Markov chain
 118 of states,^{38,39} such that the states are populated according to a
 119 Boltzmann distribution. The energy employed is not the folded
 120 protein's energy, but rather its *folding* energy, that is, the
 121 difference between its folded and unfolded state energies.³³
 122 One possible elementary move is a “mutation”, we modify the
 123 side chain type $t \rightarrow t'$ at a chosen position i in the folded

protein, assigning a particular rotamer r' to the new side chain.¹²⁴
 We consider the same mutation in the unfolded state. For a
 125 particular sequence S , the unfolded state energy has the form:¹²⁶

$$E^u = \sum_{i \in S} E^r(t_i) \quad (1) \quad 127$$

The sum is over all amino acids; t_i represents the side chain
 128 type at position i . The type-dependent quantities $E^r(t) \equiv E_t^r$ are
 129 referred to as “reference energies”; they can be thought of as
 130 effective chemical potentials of each amino acid type. The
 131 energy change due to a mutation has the form:¹³²

$$\begin{aligned} \Delta E &= \Delta E^f - \Delta E^u \\ &= (E^f(\dots t'_i, r'_i \dots) - E^f(\dots t_i, r_i \dots)) - (E^r(t'_i) - E^r(t_i)) \end{aligned} \quad (2) \quad 133$$

where ΔE^f and ΔE^u are the energy changes in the folded and
 134 unfolded state, respectively. The reference energies are essential
 135 parameters in the simulation model. Our goal here is to choose
 136 them empirically so that the simulation produces amino acid
 137 frequencies that match a set of target values, for example
 138 experimental values in the Pfam database. Specifically, we will
 139 choose them so as to maximize the probability, or likelihood of
 140 the target sequences.¹⁴¹

Let S be a particular sequence. Its Boltzmann probability is¹⁴²

$$p(S) = \frac{1}{Z} \exp(-\beta \Delta G_S) \quad (3) \quad 143$$

where $\Delta G_S = G_S^f - E_S^u$ is the folding free energy of S , G_S^f is the
 144 free energy of the folded form, $\beta = 1/kT$ is the inverse
 145 temperature, and Z is a normalizing constant (the partition
 146 function). We then have¹⁴⁷

$$\begin{aligned} kT \ln p(S) &= \sum_{i \in S} E^r(t_i) - G_S^f - kT \ln Z \\ &= \sum_{t \in aa} n_S(t) E_t^r - G_S^f - kT \ln Z \end{aligned} \quad (4) \quad 148$$

where the sum on the right is over the amino acid types and¹⁴⁹
 $n_S(t)$ is the number of amino acids of type t within the sequence¹⁵⁰
 S .¹⁵¹

We now consider a set \mathcal{S} of N target sequences S ; we denote¹⁵²
 \mathcal{L} the probability of the entire set, which depends on the model
 parameters E_t^r ; we refer to \mathcal{L} as their likelihood.⁴⁰ We have¹⁵³

$$\begin{aligned} kT \ln \mathcal{L} &= \sum_S \sum_{t \in aa} n_S(t) E_t^r - \sum_S G_S^f - NkT \ln Z \\ &= \sum_{t \in aa} N(t) E_t^r - \sum_S G_S^f - NkT \ln Z \end{aligned} \quad (5) \quad 154$$

where $N(t)$ is the number of amino acids of type t in the whole
 155 data set \mathcal{S} . The normalization factor or partition function Z is a
 156 sum over all possible sequences R :¹⁵⁷

$$\begin{aligned} Z &= \sum_R \exp(-\beta \Delta G_R) \\ &= \sum_R \exp(-\beta G_R^f) \prod_{t \in aa} \exp(\beta n_R(t) E_t^r) \end{aligned} \quad (6) \quad 158$$

In view of maximizing \mathcal{L} , we consider the derivative of Z with
 159 respect to one of the E_t^r :¹⁶⁰

$$\frac{\partial Z}{\partial E_t^r} = \sum_R \beta n_R(t) \exp(-\beta G_R^f) \prod_{s \in aa} \exp(\beta n_R(s) E_s^r) \quad (7) \quad 161$$

162 We then have

$$\frac{kT}{Z} \frac{\partial Z}{\partial E_t^r} = \frac{\sum_R n_R(t) \exp(-\beta \Delta G_R)}{\sum_R \exp(-\beta \Delta G_R)} = \langle n(t) \rangle \quad (8)$$

164 The quantity on the right is the Boltzmann average of the
165 number $n(t)$ of amino acids t over all possible sequences. In
166 practice, this is the average population of t we would obtain in a
167 long MC simulation. As usual in statistical mechanics,⁴¹ the
168 derivative of $\ln Z$ with respect to one quantity (E_t^r) is equal to
169 the ensemble average of the conjugate quantity ($\beta n_S(t)$).
170 A necessary condition to maximize $\ln \mathcal{L}$ is that its derivatives
171 with respect to the E_t^r should all be zero. We see that

$$\frac{1}{N} \frac{\partial}{\partial E_t^r} \ln \mathcal{L} = \frac{1}{N} \sum_s n_s(t) - \langle n(t) \rangle = \frac{N(t)}{N} - \langle n(t) \rangle \quad (9)$$

172

173 so that

$$\mathcal{L} \text{ maximum} \Rightarrow \frac{N(t)}{N} = \langle n(t) \rangle, \quad \forall t \in \text{aa} \quad (10)$$

175 Thus, to maximize \mathcal{L} , we should choose $\{E_t^r\}$ such that a long
176 simulation gives the same amino acid frequencies as the target
177 database.

178 **2.2. Searching for the Maximum Likelihood.** We will
179 use two methods to approach the maximum likelihood $\{E_t^r\}$
180 values, starting from a current guess $\{E_t^r(n)\}$. With the first
181 method, we step along the gradient of $\ln \mathcal{L}$, using the update
182 rule:⁴⁰

$$\begin{aligned} E_t^r(n+1) &= E_t^r(n) + \alpha \frac{\partial}{\partial E_t^r} \ln \mathcal{L} \\ &= E_t^r(n) + \delta E(n_t^{\text{exp}} - \langle n(t) \rangle_n) \end{aligned} \quad (11)$$

184 Here, n is an iteration number; α is a constant; $n_t^{\text{exp}} = N(t)/N$ is
185 the mean population of amino acid type t in the target database;
186 $\langle \cdot \rangle_n$ indicates an average over a simulation done using the
187 current reference energies $\{E_i^r(n)\}$, and δE is an empirical
188 constant with the dimension of an energy, referred to as the
189 update amplitude. This update procedure is repeated until
190 convergence. We refer to this method as the linear update
191 method.

192 The second method, used previously,^{26,27} employs a
193 logarithmic update rule:

$$E_t^r(n+1) = E_t^r(n) - kT \ln \frac{\langle n(t) \rangle_n}{n_t^{\text{exp}}} \quad (12)$$

195 where kT is a thermal energy, set empirically to 0.5 kcal/mol (1
196 cal = 4.184 J). We refer to this as the logarithmic update
197 method. Both the linear and logarithmic update methods
198 converge to the same optimum, specified by eq 10.

199 In the later iterations, some E_t^r values tended to converge
200 slowly, with an oscillatory behavior. Therefore, we sometimes
201 used a modified update rule, where the $E_t^r(n+1) - E_t^r(n)$
202 value computed with the linear or logarithmic method for
203 iteration n was mixed with the value computed at the previous
204 iteration, with the $(n-1)$ value having a weight of $1/3$ and the
205 current value a weight of $2/3$. At each iteration, we typically ran
206 500 million steps (per replica) of Replica Exchange Monte
207 Carlo.

3. COMPUTATIONAL METHODS

208 **3.1. Effective Energy Function for the Folded State.** The energy matrix was computed with the following effective
209 energy function for the folded state:
210

$$\begin{aligned} E &= E_{\text{bonds}} + E_{\text{angles}} + E_{\text{dihedral}} + E_{\text{impr}} + E_{\text{vdw}} + E_{\text{Coul}} \\ &\quad + E_{\text{solv}} \end{aligned} \quad (13)$$

212 The first six terms in eq 13 represent the protein internal
213 energy. They were taken from the Amber ff99SB empirical
214 energy function,⁴² slightly modified for CPD. The original
215 backbone charges were replaced by a unified set, obtained by
216 averaging over all amino acid types and adjusting slightly to
217 make the backbone portion of each amino acid neutral.⁴³ The
218 last term on the right of eq 13, E_{solv} , represents the contribution
219 of solvent. We used a “Generalized Born + Surface Area”, or
220 GBSA implicit solvent model:⁴⁴

$$\begin{aligned} E_{\text{solv}} &= E_{\text{GB}} + E_{\text{surf}} = \frac{1}{2} \left(\frac{1}{\epsilon_W} - \frac{1}{\epsilon_p} \right) \\ &\quad \sum_{ij} q_i q_j (r_{ij}^2 + b_i b_j \exp[-r_{ij}^2/4b_i b_j])^{-1/2} + \sum_i \sigma_i A_i \end{aligned} \quad (14)$$

222 Here, ϵ_W and ϵ_p are the solvent and protein dielectric
223 constants; r_{ij} is the distance between atoms i,j and b_i is the
224 “solvation radius” of atom i .^{44,45} A_i is the exposed solvent
225 accessible surface area of atom i ; σ_i is a parameter that reflects
226 each atom’s preference to be exposed or hidden from solvent.
227 The solute atoms were divided into four groups with specific σ_i
228 values. The values were -5 (nonpolar), -40 (aromatic), -80
229 (polar), and -100 (ionic) cal/mol/Å². Hydrogen atoms were
230 assigned a surface coefficient of 0. Surface areas were computed
231 by the Lee and Richards algorithm,⁴⁶ implemented in the
232 XPLOR program,⁴⁷ using a 1.5 Å probe radius. The MC
233 simulations used a protein dielectric of $\epsilon_p = 4$ or 8 (see
234 Results).

235 In the GB energy term, the atomic solvation radius b_i
236 approximates the distance from i to the protein surface and is
237 a function of the coordinates of all the protein atoms. The
238 particular b_i form corresponds to a GB variant we call GB/
239 HCT, after its original authors,⁴⁴ with model parameters
240 optimized for use with the Amber force field.⁴⁵ Since b_i
241 depends on the coordinates of all the solute atoms,⁴⁴ an
242 additional approximation is needed to make the GB energy
243 term pairwise additive and to define the energy matrix.^{27,48} We
244 use a “Native Environment Approximation”, or NEA, in which
245 the solvation radius b_i of each particular group (backbone, side
246 chain or ligand) is computed ahead of time, with the rest of the
247 system having its native sequence and conformation.^{27,48}

248 The surface energy contribution E_{surf} is not pairwise additive
249 either, because in a protein structure, surface area buried by one
250 side chain may also be buried by another. To make this energy
251 pairwise, we used the method of Street et al.⁴⁹ In this method,
252 the buried surface of a side chain is computed by summing over
253 the neighboring side chain and backbone groups. For each
254 neighboring group, the contact area with the side chain of
255 interest is computed, independently of other surrounding
256 groups. The contact areas are then summed. To avoid
257 overcounting the buried surface area, a scaling factor is applied
258 to the contact areas involving buried side chains. Previous
259 studies showed that a scaling factor of 0.65 works well.^{45,48}

3.2. Reference Energies in the Unfolded State. In the CPD model, the unfolded state energy depends on the sequence composition through a set of reference energies E_t^r (eq 1). Here, the reference energies were assigned based on amino acid types t , taking into account also the position of each amino acid in the folded structure, through its buried or solvent-exposed character. Thus, for a given type (Ala, say), there were two distinct E_t^r values: a buried and an exposed value. This is so even though the reference energies are used to represent the unfolded, not the folded state. This procedure is supported by three assumptions. First, we assume residual structure is present in the unfolded state, so that amino acids partly retain their buried/exposed character. Second, we hypothesize that the unfolded state model compensates in a systematic way for errors in the folded state energy function, so that the folded structure contributes indirectly to the reference energies. Third, this strategy makes the model less sensitive to variations in the length of surface loops, and to the proportion of surface vs buried residues, which can vary widely among homologues (see below). As a result, the model should be more transferable within a protein family.

Distinguishing buried/exposed positions doubles the number of adjustable E_t^r parameters. Conversely, to reduce the number of adjustable parameters, we group amino acids into homologous classes (given in Results). Within each class c , and for each type of position (buried or exposed), the reference energies have the form

$$E_t^r = E_c^r + \delta E_t^r \quad (15)$$

Here, E_c^r is an adjustable parameter while δE_t^r is a constant, computed as the molecular mechanics energy difference between amino acid types within the class c , assuming an unfolded conformation where each amino acid interacts only with itself and with solvent. Specifically, we ran MC simulations of an extended peptide (the Syndecan1 peptide; see below) and computed the average energies for each amino acid type at each peptide position (excluding the termini). We took the differences between amino acid types and averaged them over the peptide positions. During likelihood maximization, E_c^r was optimized while δE_t^r was held fixed. To optimize the E_c^r values, we applied the linear or logarithmic method while the target frequencies corresponded to the experimental frequencies of the amino acid classes, n_c^{exp} , rather than of the individual types (n_t^{exp} , above).

3.3. Experimental Sequences and Structural Models. We considered the Tiam1 and Cask PDZ domains, whose crystal structures are known (PDB codes 4GVD and 1KWA,³ respectively). They both belong to the class II binding motif, which recognizes the pattern $\Phi\text{-X-}\Phi$ at the C-terminus of its peptide ligand, where Φ is a hydrophobic amino acid. To define the target amino acid frequencies for likelihood maximization, we collected homologous sequences for each PDZ domain. We identified homologous sequences by using the Blast tool to search the Uniprot database, with the sequences taken from the PDB file as the query and the Blosum62 scoring matrix. We retained homologues with a sequence identity, relative to the query, above a 60% threshold and below an 85% threshold. If two homologues had a mutual sequence identity above 95%, one of the two was viewed as redundant and was discarded. This led to 50 Tiam1 and 126 Cask homologue sequences. The two sets of homologues are referred to as \mathcal{H}_T and \mathcal{H}_C , respectively. For each of the sets, say \mathcal{H} , we average over all homologues and all positions to obtain a computation of the

overall amino acid frequencies. The averaging is done separately for buried and exposed positions. The resulting amino acid frequencies are denoted $\{f_t^b(\mathcal{H}), f_t^e(\mathcal{H})\}$, where the subscript t represents an amino acid type and the superscripts b, and e refer to buried and exposed positions, respectively. Finally, the sets of mean frequencies derived from \mathcal{H}_T and \mathcal{H}_C were themselves averaged, giving the overall target amino acid frequencies, $f_t^b = (f_t^b(\mathcal{H}_T) + f_t^b(\mathcal{H}_C))/2$ for each type t , and similarly for the exposed positions. Distinct target frequencies were thus obtained for buried and exposed positions.

Model parametrization and testing were mostly done for the apo state of each protein. However, for Cask, no apo X-ray structure was available at the beginning of this work, so a holo-like structure was used, where the peptide binding site is occupied by the C-terminus of another PDZ domain in the crystal lattice; the apo state was then modeled by removing this peptide. For the PDZ domain of Tiam1, we also used a holo structure then modeled the apo state by removing the peptide. For this PDZ domain, the backbone rms deviation between the apo and holo X-ray structures is just 0.5 Å; therefore, we expect the CPD model to be transferable between apo/holo Tiam1 states. For additional testing, we also considered two class I PDZ domains, syntenin and DLG2 (second PDZ domain in both cases), which recognize the pattern S/T-X-Φ at the C-terminus of its peptide ligand. Their X-ray structures are 1R6J and 2BYG, respectively. In both these structures, the peptide ligand was not cocrystallized, but the peptide binding site of each PDZ domain was partly occupied by the C-terminus of another protein molecule in the crystal lattice. The structures employed are listed in Table 1.

351 t1

Table 1. Test Proteins

protein name ^a	PDB code	residue numbers	no. active positions ^c
syntenin(2)	1R6J	192–273	72
DLG2(2)	2BYG	186–282	82
Cask	1KWA ^b	487–568	74
Tiam1	4GVD ^b	837–930	84

^aIn parentheses: number of the PDZ domain within the protein.

^bHolo or holo-like structures. ^cThe number of non-Gly, non-Pro positions, which can mutate during the design simulations.

To carry out the Monte Carlo design calculations, the structures were prepared and energy matrices were computed using procedures described previously.^{15,50} Two missing segments in the Tiam1 PDZ domain (residues 851–854 and 868–869) were built using the Modeler program.⁵¹ The peptide ligand was removed from the PDB structure for most of the design calculations before computing the energy matrix. For each pair of amino acid side chains, the interaction energy was computed after 15 steps of energy minimization, with the backbone held fixed and only the interactions of the pair with each other and the backbone included.²⁶ This short minimization alleviates the discrete rotamer approximation. Side chain rotamers were described by a slightly expanded version of the library of Tuffery et al.,⁵² which has a total of 254 rotamers (summed over all amino acid types). This expanded library includes additional hydrogen orientations for OH and SH groups.⁴⁸ This rotamer library was chosen for its simplicity and because it gave very good performance in side chain

370 placement tests, comparable to the specialized Scwrl4 program
 371 (which uses a much larger library).^{53,54}

372 **3.4. Monte Carlo Simulations.** Sequence design was
 373 performed with Proteus, which runs long Monte Carlo (MC)
 374 simulations where selected amino acid positions can mutate
 375 freely. The choice of mutating positions is user-defined and
 376 depends on the specific design challenge. Four different choices
 377 occurred in the present work. First, to optimize the reference
 378 energies, we did simulations where about half of the positions
 379 could mutate at a time. Second, the optimized models were
 380 tested in simulations where all positions except Gly and Pro
 381 were free to mutate. Hydrophobic titration of two PDZ
 382 domains also employed this choice. Third, to produce designed
 383 sequences to test through molecular dynamics, we did MC
 384 simulations where Gly, Pro, and 11 positions closely involved in
 385 peptide binding were held fixed, while all other positions were
 386 allowed to mutate. Fourth, in the second Tiam1 application,
 387 only four positions in the protein could mutate. In all these
 388 cases (with two exceptions), mutations occurred randomly,
 389 subject only to the MMGBSA energy function that drives the
 390 simulation. In only two cases, an additional, “experimental”
 391 energy term was used to explicitly bias the simulation to stay
 392 close to the natural, Pfam sequences.

393 The Monte Carlo simulations used one- and two-position
 394 moves, where either rotamers, amino acid types, or both
 395 changed. For two-position moves, the second position was
 396 selected among those that had a significant interaction energy
 397 with the first (i.e., there was at least one rotamer conformation
 398 where their unsigned interaction energy was 10 kcal/mol or
 399 more). In addition, sampling was enhanced by Replica
 400 Exchange Monte Carlo (REMC), where several MC simu-
 401 lations (“replicas” or “walkers”) were run in parallel, at different
 402 temperatures. Periodic swaps were attempted between the
 403 conformations of two walkers i, j (adjacent in temperature).
 404 The swap was accepted with the probability

$$405 \text{acc}(\text{swap}_{ij}) = \text{Min}[1, e^{(\beta_i - \beta_j)(\Delta E_i - \Delta E_j)}] \quad (16)$$

406 where β_i, β_j are the inverse temperatures of the two walkers and
 407 $\Delta E_i, \Delta E_j$ are the changes in their folding energies due to the
 408 conformation change.^{55,56} We used eight walkers, with thermal
 409 energies kT_i that range from 0.125 to 3 kcal/mol, spaced in a
 410 geometric progression: $T_{i+1}/T_i = \text{constant}$.⁵⁵ Simulations were
 411 done with the proteus program (which is part of the Proteus
 412 package).²⁷ REMC was implemented with an efficient, shared-
 413 memory, OpenMP parallelization.³³

414 One simulation of Tiam1 and one of Cask were done that
 415 included an “experimental”, biasing energy term, which
 416 penalized sequences that had a low similarity to a reference,
 417 experimental set. The bias energy had the form

$$418 \delta E_{\text{bias}} = c \sum_i (S_i^{\text{rand}} - S(t_i)) \quad (17)$$

419 where the sum extends over the amino acid positions i ; t_i is the
 420 side chain type at position i ; $S(t_i)$ is the (dimensionless)
 421 Blosum40 similarity score versus the corresponding position in
 422 the Pfam RPSS sequence alignment; S_i^{rand} is the mean score
 423 (versus the same Pfam column) for a random type (where all
 424 types are equiprobable), and $c = 0.5$ kcal/mol.

425 **3.5. Rosetta Sequence Generation.** Monte Carlo
 426 simulations were also performed using the Rosetta program
 427 and energy function.³⁷ The simulations were done using
 428 version 2015.38.58158 of Rosetta (freely available online),

429 using the command `fixbb -s Tiam1.pdb -resfile Tiam1.res -nstruct 10000 -ex1 -ex2 -linme- m_ig 10` where the ex1 and ex2 options activate an enhanced rotamer search for buried side chains, the last option (`linmem_ig`) corresponds to on-the-fly energy calculation, and default parameters were used otherwise. Gly and Pro residues present in the wildtype protein were not allowed to mutate, and positions that do mutate could not change into Gly or Pro (as with the Proteus design simulations). Simulations were run for each PDZ domain until 10 000 unique low energy sequences were identified, corresponding to run times of about 5 min per sequence on a single core of a recent Intel processor, for a total of 10 h (per protein) using 80 cores. This was comparable to the cost of the Proteus calculations (energy matrix plus Monte Carlo simulations).

430 **3.6. Sequence Characterization.** Designed sequences were compared to the Pfam alignment for the PDZ family, using the Blosum40 scoring matrix and a gap penalty of -6. This matrix is appropriate for comparing rather distant homologues (CPD and Pfam sequences in this case). Each Pfam sequence was also compared to the Pfam alignment, which allowed comparison between the designed sequences and a typical pair of natural PDZ domains. For these Pfam/Pfam comparisons, if a test PDZ domain T was part of the Pfam alignment, the T/T self-comparison was left out, to be more consistent with the designed/Pfam comparisons. The Pfam alignment was the “RPSS” alignment, consisting of 12 255 sequences. Similarities were computed separately for the 14 core residues and 16 surface residues, defined by their near-complete burial or exposure (listed in [Results](#)) and for the entire protein.

431 Designed sequences were submitted to the Superfamily library of Hidden Markov Models,^{57,58} which attempts to classify sequences according to the Structural Classification of Proteins, or SCOP.⁵⁹ 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. The hmmscan program was executed using an E-value threshold of 10^{-10} and a total of 15 438 models to represent the SCOP database.

432 To compare the diversity in the designed sequences with the diversity in natural sequences, we used the standard, position-dependent sequence entropy,⁶⁰ computed as follows:

$$433 S_i = - \sum_{j=1}^6 f_j(i) \ln f_j(i) \quad (18)$$

434 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 employed six residue classes, corresponding to the following groups: {LVIMC}, {FYW}, {G}, {ASTP}, {EDNQ}, and {KRH}. This classification was obtained by a cluster analysis of the BLOSUM62 matrix,⁶¹ and by analyzing residue–residue contact energies in proteins.⁶² To obtain a sense for how many amino acid types appeared at a typical position, we report the residue entropy in its exponential form, $\exp(S)$ (which ranges from 1 to 6), averaged over the protein chain.

435 **3.7. Protein:Peptide Binding Free Energies.** For the Tiam1 PDZ domain, we used design calculations in the presence and absence of a bound peptide to obtain estimates of

489 the binding free energy differences between protein variants. If
 490 a given sequence S was sampled in both the apo and holo
 491 states, we computed the mean energy $\langle E_{\text{holo}}(S) \rangle$, $\langle E_{\text{apo}}(S) \rangle$ in
 492 each of the two states by averaging over the sampled
 493 conformations. Then, we took the difference

$$\Delta\Delta E(S, S') = (\langle E_{\text{holo}}(S') \rangle - \langle E_{\text{apo}}(S') \rangle) \\ - (\langle E_{\text{holo}}(S) \rangle - \langle E_{\text{apo}}(S) \rangle) \quad (19)$$

494

495 as our estimate of the binding free energy difference between
 496 the variants S and S' . We also computed binding free energy
 497 differences between groups of homologous sequences, say S
 498 and S' , by pooling the homologous sequences sampled in
 499 either the apo or holo state, then averaging over the
 500 conformations sampled and taking the energy difference
 $\Delta\Delta E(S, S')$.

501 **3.8. Molecular Dynamics Simulations.** Wildtype and a
 502 quadruple mutant Tiam1 and 10 sequences designed with
 503 Proteus were subjected to MD simulations with explicit solvent
 504 and no peptide ligand. The starting structures were taken from
 505 the MC trajectory or the crystal structure (wildtype protein and
 506 quadruple mutant: PDB codes 4GVD and 4NXQ) and slightly
 507 minimized with harmonic restraints to maintain the backbone
 508 geometry. The protein was immersed in a large box of
 509 nonoverlapping waters. The solvated system was truncated to
 510 the shape of a truncated octahedral box using the Charmm
 511 graphical interface or GUI.⁶³ The minimum distance between
 512 protein atoms and the box was 15 Å and the final models
 513 included about 11 000 water molecules. A few sodium or
 514 chloride ions were included to ensure overall electroneutrality.
 515 The protonation states of histidines were assigned to be neutral,
 516 based on visual inspection. MD was done at room temperature
 517 and pressure, using a Nose-Hoover thermostat and baro-
 518 stat.^{64,65} Long-range electrostatic interactions were treated with
 519 a Particle Mesh Ewald approach.⁶⁶ The Amber ff99SB force
 520 field and the TIP3P model⁶⁷ were used for the protein and
 521 water, respectively. Simulations were run for 100–1000 ns,
 522 depending on the sequence, using the Charmm and NAMD
 523 programs.^{68,69}

4. RESULTS

524 **4.1. Experimental structures and sequences.** Three
 525 dimensional (3D) structures of the four test PDZ domains are
 526 shown in Figure 1A. Fourteen core residues (identified visually)
 527 superimposed well between the structures, while loops and
 528 chain termini displayed large deviations. The Tiam1 α_2 helix is
 529 rotated slightly outward compared to the other three
 530 structures.⁷⁰ Figure 1B illustrates the similarity between pairs
 531 of PDZ domains, as determined by the rms deviation between
 532 structurally aligned C_α atoms and the pairwise sequence
 533 identities. The rms deviations are between 1.0 and 2.1 Å and
 534 the sequence identities between 17 and 33%. The Tiam1/Cask
 535 sequence identity is 33% and their structural deviation is 1.7 Å
 536 based on 42 aligned C_α atoms. The syntenin and DLG2
 537 structures are more similar, with a structural deviation of 1.0 Å
 538 based on 60 aligned C_α atoms.

539 Sequence conservation within the four PDZ domains and a
 540 subset of the Pfam seed alignment is shown in Figure 2. The 14
 541 positions used to define the hydrophobic core are highly,
 542 although not totally conserved within the Pfam seed alignment.
 543 Arginine, Lys, and Gln appear at some of the positions, since in
 544 small proteins such as PDZ domains, the long hydrophobic

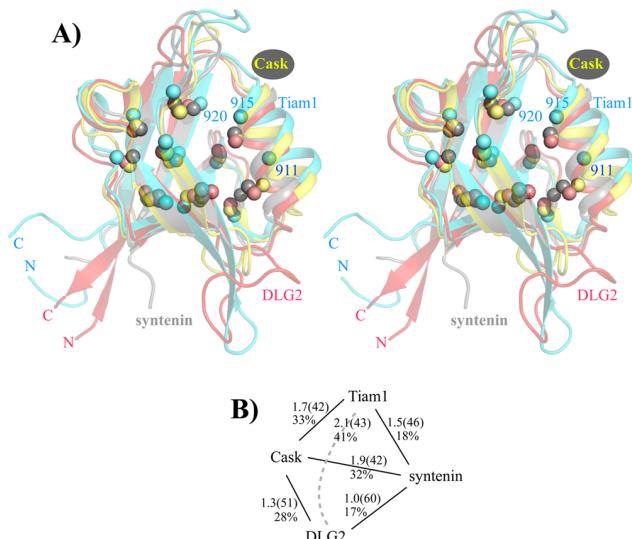


Figure 1. (A) Three dimensional view of four PDZ domains. The C_β atoms of 14 hydrophobic core residues are shown as spheres. Three core positions designed in this work are labeled (Tiam1 numbers). (B) Cluster representation of the PDZ domains studied. The links between domains are labeled with the percent identity scores and backbone rms deviations (Å); the number in parentheses is the number of aligned C_α atoms used to compute the rms deviation.

portion of these side chains can be buried in the core while still allowing the polar tip of the side chain to be exposed to solvent. A few Asp and Glu residues also appear, in places where the sequence alignment may not reflect closely the 3D side chain superposition.

4.2. Optimizing the Unfolded State Model. We optimized the reference energies E_t^r for Tiam1 and Cask, using their natural homologues to define the target amino acid frequencies. The protein dielectric constant ϵ_p was either 4 or 8. The E_t^r optimizations all converged to within 0.05 kcal/mol after about 20 iterations for most amino acid types, and to within 0.1 kcal/mol for the others (the weakly populated types), using either the linear or the logarithmic method (eq 11 or 12). Table 2 indicates the final reference energies. The E_t^r values were compared to, and agreed qualitatively with the energies computed from an extended peptide structure, which provides a less empirical model of the unfolded state. Table 3 compares the amino acid frequencies from the natural homologues and the simulations using parameters optimized with $\epsilon_p = 8$. Results obtained using parameters optimized with $\epsilon_p = 4$ are given in Supporting Information. The theoretical population of the different amino acid classes agreed well with experiment, with rms deviations of about 1%, for both the exposed and buried positions. The agreement for the amino acid types was less good, with rms deviations of 3.9%/2.4% (buried/exposed positions). The intraclass frequency distributions depend explicitly on the energy offsets δE_t^r defined within each class, which were computed with molecular mechanics (see Methods, eq 15).

4.3. Assessing Designed Sequence Quality. Family Recognition Tests. Proteus design simulations used Replica Exchange Monte Carlo with eight replicas and 750 million steps per replica, at thermal energies kT that ranged from 0.125 to 3 kcal/mol. All positions (except Gly and Pro) were allowed to mutate freely into all amino acid types except Gly and Pro. The simulations were done with the MMGBSA energy function,

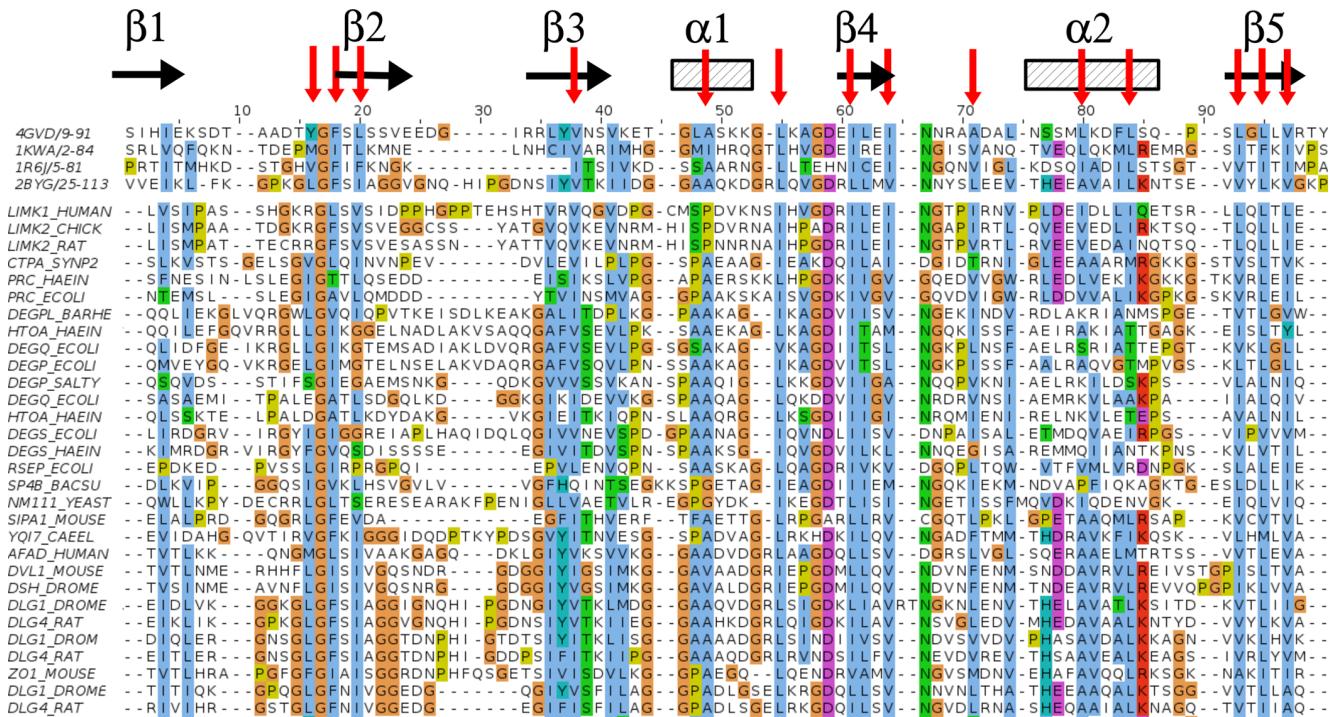


Figure 2. Alignment of natural PDZ sequences. The top four sequences were tested in this work. The others are the first 30 sequences from the Pfam seed alignment. Fourteen hydrophobic core positions are indicated by red arrows and the secondary structure elements are shown for reference. The Clustal color scheme is used, in which conserved amino acids are colored according to their physical chemical properties.

Table 2. Unfolded State Reference Energies E_t^r (kcal/mol)

residues	peptide ^a	design, $\epsilon_p = 8$		design, $\epsilon_p = 4$
		buried	exposed	
ALA	0.00	0.00	0.00	0.00
CYS	-0.85	-0.85	-0.85	-0.60
THR	-5.44	-5.44	-5.44	-8.22
SER	-6.43	-3.71	-4.74	-5.68
ASP	-17.28	-11.90	-15.88	-20.31
GLU	-17.35	-11.97	-15.95	-17.67
ASN	-12.25	-7.82	-10.22	-17.70
GLN	-11.50	-7.07	-9.47	-14.35
HIS ^b	9.02	12.53	9.73	13.52
HIS _e ^b	6.98	10.49	7.69	9.90
HIS _d ^b	7.35	10.86	8.06	10.62
ARG	-36.90	-32.00	-35.18	-54.08
LYS	-11.71	-6.76	-10.17	-8.41
ILE	4.22	4.63	3.63	5.30
VAL	-0.15	0.26	-0.74	-0.89
LEU	-0.53	-0.12	-1.12	-0.97
MET	-1.78	-2.05	-2.40	-1.11
PHE	-3.98	-0.23	-4.17	0.66
TRP	-5.96	-2.21	-6.15	0.17
TYR	-10.09	-5.80	-9.82	-7.87

^aEnergies within an extended peptide structure (averaged over positions). ^bHis protonation states.

assigned to the correct family: 91% for Tiam1 and 100% for s88 Cask, with E-values of around 10^{-3} for the family assignments. s89 These values are similar to Rosetta (90 and 98% family s90 recognition for Tiam1 and Cask). Changing the protein s91 dielectric constant to $\epsilon_p = 4$ gave somewhat poorer results s92 for Tiam1, with 53% of the sequences designed with Proteus s93 correctly recognized by Superfamily. s94

Sequences and Sequence Diversity. Tiam1 and Cask s95 sequences predicted by Proteus and Rosetta as well as natural s96 sequences are shown in Figure 3 for the 14 core residues and in s97 f3 Figure 4 for the 16 surface residues (Tiam1 only). The s98 f4 sequences are represented as sequence logos. As seen in many s99 previous CPD studies,^{30,72} agreement with experiment for the s100 core residues is very good, while agreement for the surface s101 residues is much poorer. The behavior of surface positions was s102 also probed by designing each position individually, with the s103 rest of the protein free to explore rotamers but not mutations s104 (“mono-position” design). The corresponding logo (Figure 4) s105 shows an excess of Arg and Lys residues, suggesting that the s106 CPD reference energies are not yet fully optimal, despite the s107 extensive empirical E_t^r tuning. Sequence similarity scores are s108 given in the next subsection. s109

The diversity of the natural and designed sequences was s110 characterized by a mean, exponential sequence entropy (see s111 Methods), which corresponds to a mean number of sampled s112 sequence classes per position. For example, a value of 2 at a s113 particular position indicates that amino acids from two of the s114 six classes are present at that position within the set of analyzed s115 sequences. An overall average value of two indicates that on s116 average, two amino acid classes are present at any position s117 within the analyzed sequences. For reference, the Pfam RPSS s118 set of 12 255 natural sequences has a mean entropy of 3.4. s119 Pooling the designed Tiam1 and Cask sequences gave an s120

without any bias toward natural sequences or any limit on the s82 number of mutations. The 10 000 sequences with the lowest s83 energies among those sampled by any of the MC replicas were s84 retained for analysis, along with the 10 000 Rosetta sequences. s85 These sequences were analyzed by the Superfamily fold s86 recognition tool^{58,71} (Table 4). With a protein dielectric s87 constant of 8, we obtained a high percentage of sequences

Table 3. Amino Acid Composition (%) of Natural and Designed PDZ Proteins^a

type	natural sequences				designed sequences			
	buried		exposed		buried		exposed	
type	type	class	type	class	type	class	type	class
A	5.9		4.6		4.1		7.2	
C	1.5	11.2	1.2	13.4	8.6	12.7 [1.5]	5.8	13.6 [0.2]
T	3.8		7.6		0.0		0.6	
S	4.7	4.7	10.2	10.2	4.9	4.9 [0.2]	10.7	10.7 [0.5]
D	3.5		6.2		7.4		8.0	
E	6.1	9.6	10.5	16.7	2.0	9.4 [-0.2]	8.1	16.1 [-0.6]
N	1.9	2.7	7.4		1.8		8.6	
Q	0.8		8.7	16.1	1.0	2.8 [0.1]	8.5	17.1 [1.0]
H ⁺	0.7		4.7		0.1		1.8	
H _e	0.0	0.7	0.0	4.7	0.6	0.9 [0.2]	2.2	4.5 [-0.2]
H _d	0.0		0.0		0.2		0.5	
I	15.7		4.1		25.1		8.4	
V	13.5	49.6	5.5	14.4	12.8	46.7 [-2.9]	3.3	15.3 [0.9]
L	20.4		4.8		8.8		3.6	
M	5.0	5.0	1.4	1.4	5.9 [0.9]	5.9	1.4 [0.0]	1.4
K	6.5	6.5	10.1	10.1	5.5	5.5 [-1.0]	10.8	10.8 [0.7]
R	1.8	1.8	9.5	9.5	2.2	2.2 [0.4]	9.1	9.1 [-0.4]
F	5.0	5.0	0.4		3.2		0.3	
W	0.0		0.0	0.4	2.3	5.5 [0.5]	0.2	0.5 [0.1]
Y	2.9	2.9	0.9	0.9	3.4	3.4 [0.5]	0.9	0.9 [0.0]
G	0.0		1.7		0.0		0.0	
P	0.3	0.3	0.4	2.1	0.0	0.0 [-0.3]	0.0	0.0 [-2.1]

^aCompositions are given for buried/exposed positions, for individual amino acid types (left) and for classes (right); values in brackets (right) are the deviations between design and experiment per class. The experimental target set included the Tiam1 and Cask homologues.

Table 4. Fold Recognition of Designed Sequences by Superfamily

protein	design model	match/seq length ^a	superfamily		family	
			E-value ^b	success no. ^c	E-value ^b	success no. ^c
Tiam1	Proteus, $\epsilon_p = 4$	53/94	1.0×10^{-4}	10000	7.0×10^{-2}	5259
Cask	Proteus, $\epsilon_p = 4$	76/83	5.1×10^{-7}	10000	1.6×10^{-2}	10000
syntenin	Proteus, $\epsilon_p = 8$	69/91	$1.3e \times 10^{-2}$	9999	$4. \times 10^{-3}$	9999
DLG2	Proteus, $\epsilon_p = 8$	85/97	8.0×10^{-9}	10000	5.0×10^{-3}	10000
Tiam1	Proteus, $\epsilon_p = 8$	64/94	1.2×10^{-4}	9920	5.2×10^{-2}	9058
Cask	Proteus, $\epsilon_p = 8$	71/83	3.2×10^{-7}	10000	8.2×10^{-3}	10000
Tiam1	Rosetta	65/94	4.4×10^{-4}	9035	$2.8e \times 10^{-2}$	9030
Cask	Rosetta	68/83	2.8×10^{-5}	9832	7.5×10^{-3}	9832
syntenin	Rosetta	76/82	7.3×10^{-13}	10000	1.8×10^{-3}	10000
DLG2	Rosetta	86/97	1.3×10^{-9}	10000	9.6×10^{-4}	10000

^aThe average match length for sequences recognized by Superfamily and the total sequence length. ^bAverage E-values for superfamily assignments to the correct SCOP superfamily/family. ^cThe number of designed sequences (out of 10000 tested) assigned to the correct SCOP superfamily/family.

entropy of 2.2 with Rosetta and 2.0 with Proteus, indicating that these two backbone geometries cannot accommodate as much diversity as the much larger RPSS set. Taking the 10 000 lowest energy sequences sampled with the room temperature Monte Carlo replica (instead of the 10 000 lowest energies

sampled collectively by all replicas at all temperatures) and pooling Tiam1 and Cask as before gave a higher overall entropy of 2.9 with Proteus. With Rosetta, entropy in the core was only slightly below the average over all positions. With Proteus, it

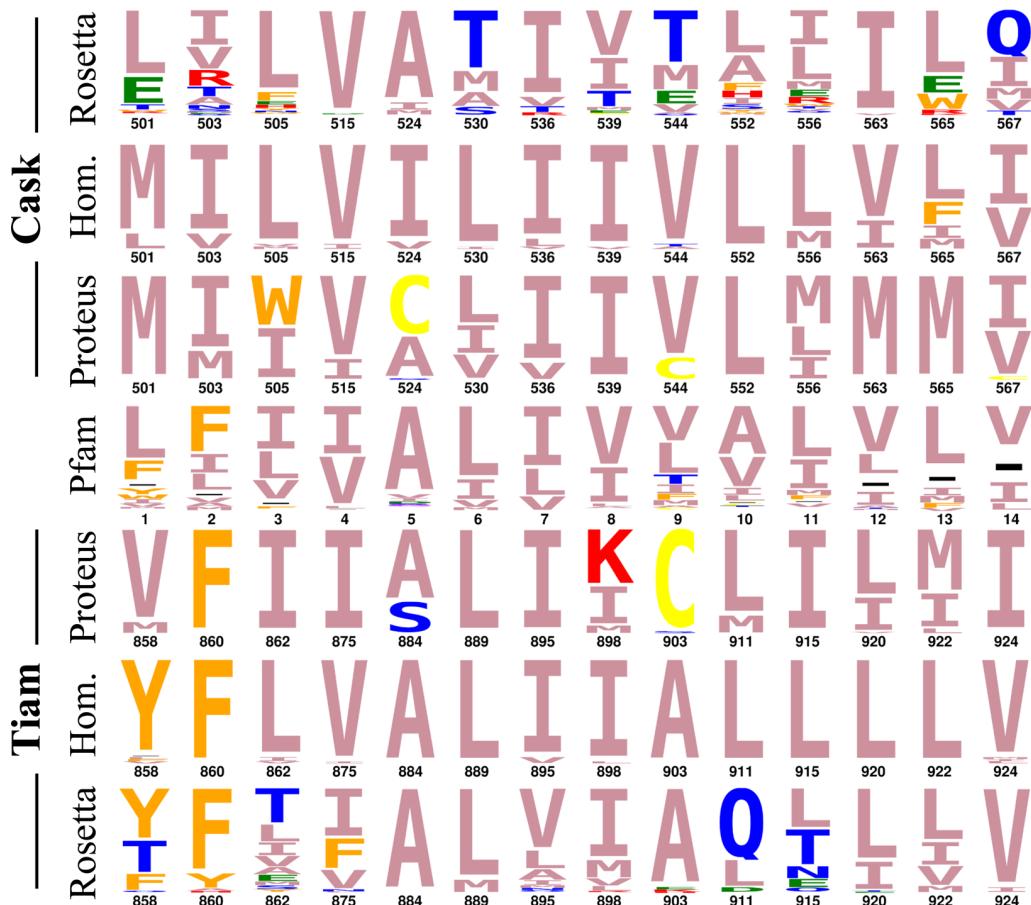


Figure 3. Sequence logos for the conserved hydrophobic core of designed and natural Tiam1 and Cask sequences. “Hom.” corresponds to the homologues that make up our target set of sequences (used for E_t optimization). “Pfam” corresponds to the Pfam seed alignment. Proteus sequences were generated with model $\epsilon_p = 8$. The height of each letter is proportional to the abundance of each type at the corresponding position in the Proteus/Rosetta simulations or the natural sequences. The color of each letter is determined by the physical chemical properties of each amino acid type.

630 was distinctly lower (1.25). For the Pfam-RP55 sequences, it
631 was 1.8.

632 **Blosum Similarity Scores.** Figure 5 shows the computed
633 Blosum40 similarity scores between designed and natural
634 sequences. With Proteus, for both Tiam1 and Cask, the overall
635 similarities overlapped with the bottom of the peak of the
636 natural scores, and were comparable to the values for the
637 Rosetta sequences. For the surface residues, shown separately,
638 similarity to the natural sequences was low (scores below zero),
639 both for Proteus and Rosetta. With a protein dielectric constant
640 of 4, Proteus performed about as well as with $\epsilon_p = 8$, giving
641 almost the same similarity averaged over all Tiam1 and Cask
642 positions, for example.

643 While the similarity scores vs Pfam with Proteus were
644 comparable to Rosetta (Figure 5), the identity scores vs the
645 wildtype sequence were significantly higher with Rosetta.
646 Identity scores excluding (respectively, including) Gly and
647 Pro positions (which did not mutate) were 20% (28%) for
648 Proteus vs 26% (34%) for Rosetta. Evidently, for Tiam1 and
649 Cask, Rosetta performed ≈ 5 fewer mutations than Proteus.

650 For certain applications, we may need to specifically explore a
651 sequence space region very similar to Pfam, beyond the
652 similarity provided by an MMGBSA energy. This can be
653 achieved by adding to the energy an “experimental,” or bias
654 energy term that explicitly favors high sequence scores. Figure 5

includes results that used such a biased energy term: by 655 construction, it led to very high similarity scores. A bias energy 656 term could also be used to limit the total number of mutations. 657

658 **4.4. Cross-Validation Tests.** As a first cross-validation test,
659 we applied the reference energies optimized using Tiam1 and
660 Cask homologues (with $\epsilon_p = 8$) to two other PDZ domains:
661 DLG2 and syntenin. The superfamily scores were comparable
662 to those obtained for Tiam1 and Cask, with 100% family
663 recognition (Table 4). Sequences designed with Rosetta for
664 DLG2 and syntenin also gave 100% family recognition. For
665 further cross-validation, we optimized reference energies using
666 an alternate set of PDZ domains: DLG2, syntenin, PSD95,
667 GRIP, INAD, and NHERF. Target frequencies were defined by
668 a small set of their natural homologues. We used $\epsilon_p = 8$. To
669 distinguish the new and initial model variants, we refer to the
670 new variant as the $n = 6$ model (it uses six PDZ domains for
671 parametrization), and the initial model as the “T+C” model (it
672 used Tiam1 and Cask). The new, $n = 6$ reference energies were
673 then used to produce designed Tiam1 and Cask sequences,
674 which were subjected to Superfamily tests and similarity
675 calculations. The Superfamily performance for Tiam1 was
676 slightly degraded, compared to the previous, T + C model. The
677 Tiam1 Superfamily score decreased from 90.6% to 76.6% for
678 family recognition. The Cask score was unchanged. Histograms
679 of Blosum similarity scores (Supporting Information) show that
680



Figure 4. Sequence logos for 16 surface positions in Tiam1. Same representation as Figure 3. The “mono-position” results are from a set of simulations where only one amino acid at a time could mutate, the rest of the protein having its native sequence (see text). Amino acid colors as in Figure 3.

the overall scores for Tiam1 and Cask with $n = 6$ were very similar to the T + C model, while the scores for the core positions were actually shifted to higher, not lower values. For DLG2 and syntenin, we also computed similarity scores using both the initial, T+C parametrization and the new, $n = 6$ parametrization. The similarity scores with the T+C model were slightly poorer than with the $n = 6$ model, as expected. The overall score decreased by about 20 points for syntenin and about 10 points for DLG2 (Supporting Information). Overall, the cross-validated models degraded performance slightly. Thus, for any PDZ domain of interest, it may be preferable to optimize reference energies specifically for that domain, rather than transferring values parametrized using other PDZ domains.

4.5. Stability of Designed Sequences in Molecular Dynamics Simulations. As another test of the design model, 10 Tiam1 sequences designed with Proteus were subjected to molecular dynamics simulations (MD) using an explicit solvent environment. These sequences were obtained using Proteus with either $\epsilon_p = 8$ or the less polarizable value $\epsilon_p = 4$. Although no peptide ligand was present during the design simulations, 11 positions in the binding pocket that make close contact with the peptide when it is present were not allowed to mutate. This was done to allow future experimental testing of designed sequences by a peptide binding assay. Among the 2500 lowest energy designed sequences, we narrowed down the choice of sequences using the following four criteria: (a) sequences should have a nonneutral isoelectric point, (b) they should be assigned to the correct SCOP family by Superfamily with good E -values, (c) they should have good Pfam similarity scores, and (d) they should have at most 15 mutations that drastically change the amino acid type compared to the wildtype protein (such a change is defined by a Blosum62 similarity score between the two amino acid types of -2 or less). Applying these criteria reduced the number of sequences to 66 from the

$\epsilon_p = 8$ model and 45 from the $\epsilon_p = 4$ model. In addition, we eliminated sequences that had two mutations that created a buried cavity and those that had net protein charges of $+6$ or more (which could lead to protein instability). A total of six sequences were chosen for further analysis. We refer to them as sequences 1–6 or seq-1, ..., seq-6. Sequences 1, 2, 4, and 5 were modified further manually to eliminate charged residues in the exposed loop 852–856 (lysines were changed manually to alanine), giving sequences 1', 2', 4', and 5'. The 10 sequences are shown in Figure 6A. Using these sequences as queries to search Uniprot with Blast, the top hits were either Tiam1 mammalian orthologs (including human Tiam1) or uncharacterized proteins, with identity scores between 35 and 40% and Blast E -values of around 10^{-8} – 10^{-7} (except for one sequence which gave hits with lower E -values of around 10^{-10}).

All 10 sequences were subjected to MD simulations with explicit solvent. Initial simulations were run for 100 ns, with all 10 sequences exhibiting good stability. Six were extended to lengths of 500 or 1000 ns. The wildtype protein (WT) was also simulated for 1000 ns. The WT sequence appeared stable over the entire simulation, judging by its rms deviations from the WT X-ray structure and from its own mean MD structure (Figure 6B). The mean MD structure had a backbone rms deviation of 1.0 Å from the WT X-ray structure (excluding 3–4 residues at each terminus and one very flexible loop, residues 850–857). During the MD trajectory, the rms deviation from the mean MD structure varied in the range 1–1.5 Å, without any visible drift (Figure 6B). A weakly stable quadruple mutant (QM) with an unfolding free energy of just 1 kcal/mol⁷⁰ was also simulated for 1000 ns. The mean MD structure of QM had a backbone rms deviation of 1.6 Å relative to the QM X-ray structure (4NXQ). Note that the X-ray structure includes a peptide ligand, whereas the MD simulation represents the apo state. The average MD structure of QM (Figure 6C) exhibited some unwinding of the N-terminus of the α_2 helix. During the

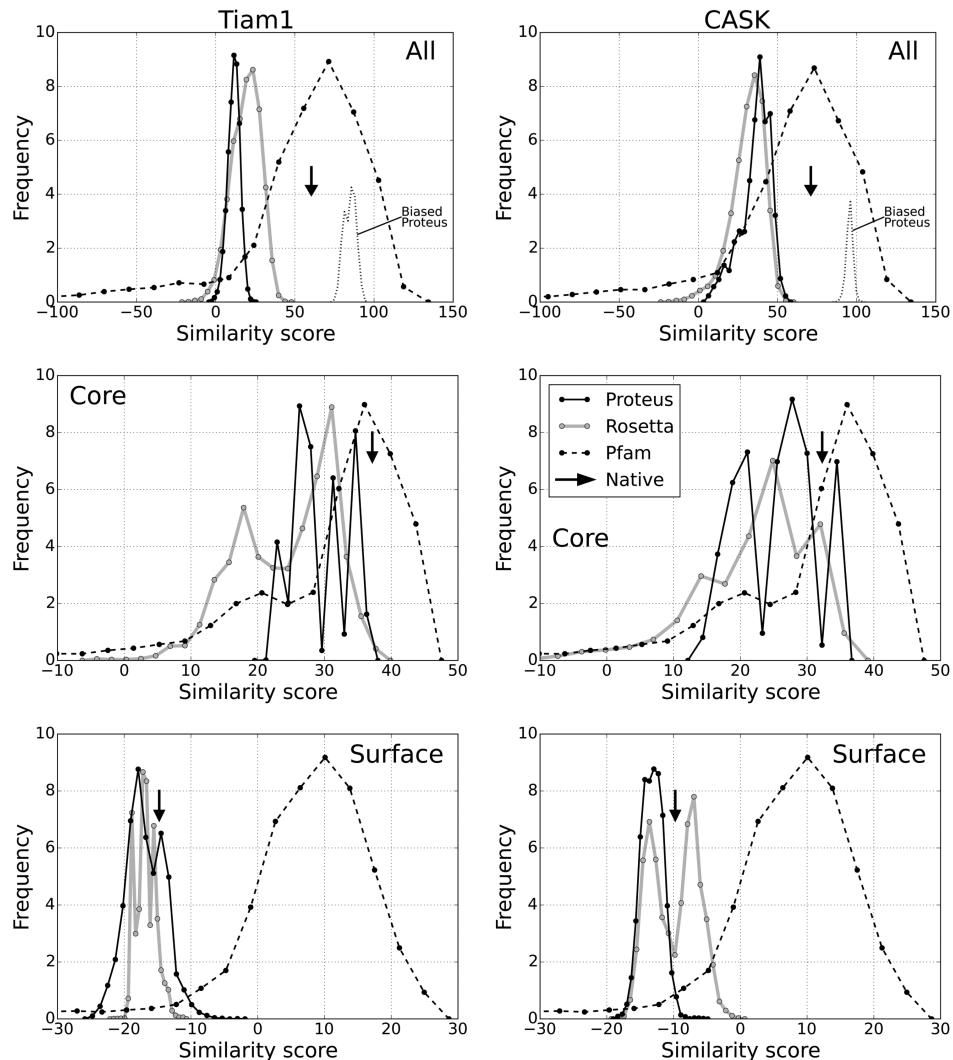


Figure 5. Histogram plots showing similarity scores for designed PDZ sequences. Similarity scores for Tiam1 (left) and Cask (right), relative to the Pfam-RPSS alignment. The scores were computed for all positions (top), 14 core positions (middle), and 16 surface positions (bottom). Values are shown for Proteus, Rosetta, and Pfam sequences (all compared to RPSS). The similarity score of the wildtype sequence is indicated in each panel by a vertical arrow. The top panels include results for Proteus simulations where a bias energy was included, which explicitly favors sequences that are similar to Pfam (dotted lines, labeled “Biased Proteus”). Notice that the designed sequences represented in each top, middle, or bottom panel were the same; only the positions included in the similarity score calculation differ between panels.

QM simulation, the structure had deviations from its average MD structure in the 0.8–1.2 Å range (Figure 6B) and appeared stable.

Sequences 1, 3, 4, and 5 were simulated for 500 ns; sequence 3 moved away from the mean MD structure toward the end of the simulation; the other three sequences appeared stable (Figure 6B). Sequence 2 (or seq-2) appeared stable up to almost 1000 ns (Figure 6B). The mean seq-2 structure exhibited a shortening of the β strands 2 and 3. Note that in the holo state, strand 2 makes direct contact with the peptide ligand. The rms deviations between the average MD structure of seq-2 and the WT and QM X-ray structures were 1.5 and 1.6 Å, respectively. During the seq-2 MD simulation, the rms deviation of seq-2 from its average MD structure varied in the range 1.3–2 Å up to almost 1000 ns. At this point, just before 1000 ns, seq-2 underwent a large fluctuation. When the simulation was extended for another 100 ns, the fluctuation largely regressed. More data are needed to determine if this fluctuation signals instability of this designed sequence.

Sequence 6 appeared stable throughout the microsecond MD simulation (Figure 6B). Its mean MD structure had a backbone rms deviation from the WT X-ray structure of just 1.0 Å, the same deviation as the mean WT MD structure. The mean MD structures of seq-6 and WT are superimposed in Figure 6C and are very similar to each other, with a 1.2 Å backbone deviation between them. During the seq-6 MD trajectory, the deviations of seq-6 away from its mean MD structure fluctuated between about 1 and 1.5 Å, without any visible drift over the microsecond MD simulation.

4.6. Application: Growing the PDZ Hydrophobic Core. As a first application of our optimized models, we examined the designability of the Tiam1 and Cask hydrophobic cores. Each PDZ domain was subjected to Replica Exchange Monte Carlo simulations with a succession of biased energy functions that increasingly favored hydrophobic residues. The first simulation included a bias energy term $\delta = 0.4$ kcal/mol (per position) that penalized hydrophobic amino acid types (ILMVAWFY). The final simulation included a bias energy term $\delta = -0.4$ kcal/mol that favored hydrophobic amino acid types (ILMVAWFY).

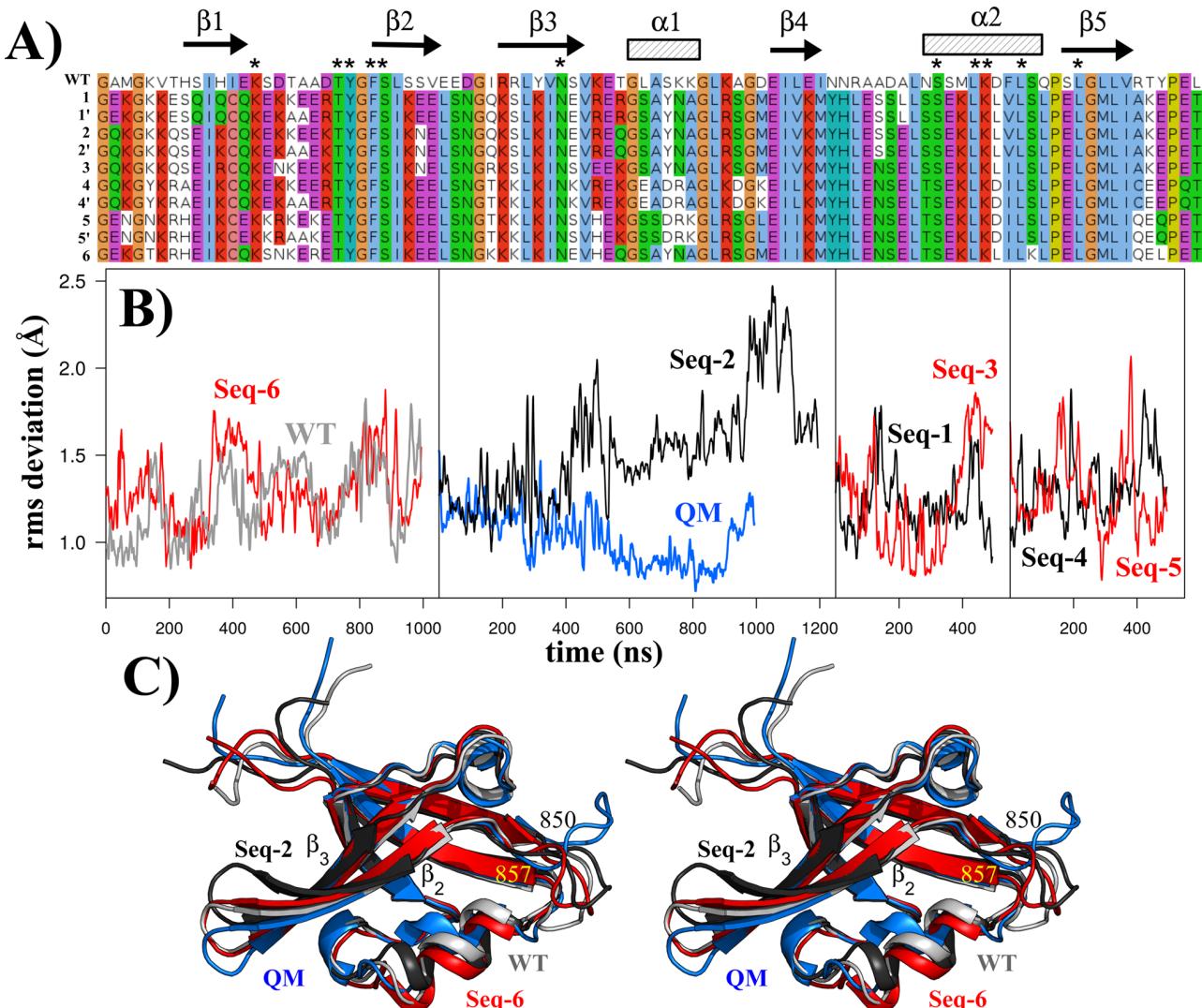


Figure 6. MD simulations of Tiam1 variants designed with Proteus. (A) Sequences of the wildtype (WT) PDZ domain and the 10 designed variants simulated by MD. Asterisks indicate peptide binding residues held fixed during the design simulations. (B) Backbone rms deviations over the course of an MD simulation for WT, QM, and six designed variants relative to the corresponding mean MD structure. (C) Mean MD structures of WT, QM, seq-6, seq-2, seq-1, seq-3, seq-4, and seq-5.

788 mol (per position) that favored hydrophobic types. Inter-
789 mediate bias energy values $\delta = 0.2, 0$, and -0.2 kcal/mol were
790 also simulated. By gradually decreasing the bias energy value δ ,
791 we effectively “titrate in” hydrophobic residues.

792 The results for Tiam1 are illustrated in Figure 7. At the
793 largest δ value, the Tiam1 hydrophobic core was depleted, with
794 10 amino acid positions (out of 94) changed to polar types.
795 The changed positions mostly lie on the outer edge of the core.
796 At the intermediate δ values, the hydrophobic core remained
797 native-like. At the most negative δ value, the hydrophobic core
798 became larger, expanding out toward surface regions, with 14
799 polar positions changed to hydrophobic types. Thus, the
800 numbers of positions changed were approximately symmetric
801 (around ± 12 changes), reflecting the bias. About 2/3 of the
802 changes were in secondary structure elements. Overall, the
803 observed propensities of each position to become polar or
804 hydrophobic in the presence of a large or small penalty bias
805 energy δ can be thought of as a hydrophobic designability
806 index. Here, 11 of the 14 conserved core positions (all but
807 positions 884, 898, and 903) remained hydrophobic even at the

808 highest level of polar bias, along with 13 other positions,
809 indicating that these positions have the highest hydrophobic
810 propensity. Furthermore, 14 positions changed from polar to
811 hydrophobic at the highest bias level, indicating that these
812 positions also have a certain hydrophobic propensity. Results
813 for Cask were similar, with 11 positions changed to polar at the
814 highest polar bias and 9 changed to hydrophobic at the highest
815 hydrophobic bias.

We also derived a parameter to describe the relative number
816 of amino acid type changes per unit bias energy. This parameter
817 was defined as the number δN of residue positions changed
818 from nonpolar to polar, divided by the product of the change
819 δE in bias energy and the mean number N of nonpolar
820 positions at zero bias. We call it the hydrophobic susceptibility,
821 χ_h . For the Tiam1 PDZ domain, this calculation amounts to
822 $\chi_h = \frac{1}{N} \frac{\delta N}{\delta E} = 0.88$ changes (per position) per kcal/mol. For
823 Cask, the susceptibility was $\chi_h = 0.71$ changes per kcal/mol.

4.7. Application to Tiam1: Designing Specificity
824 **Positions.** As a second application, we redesigned four
825

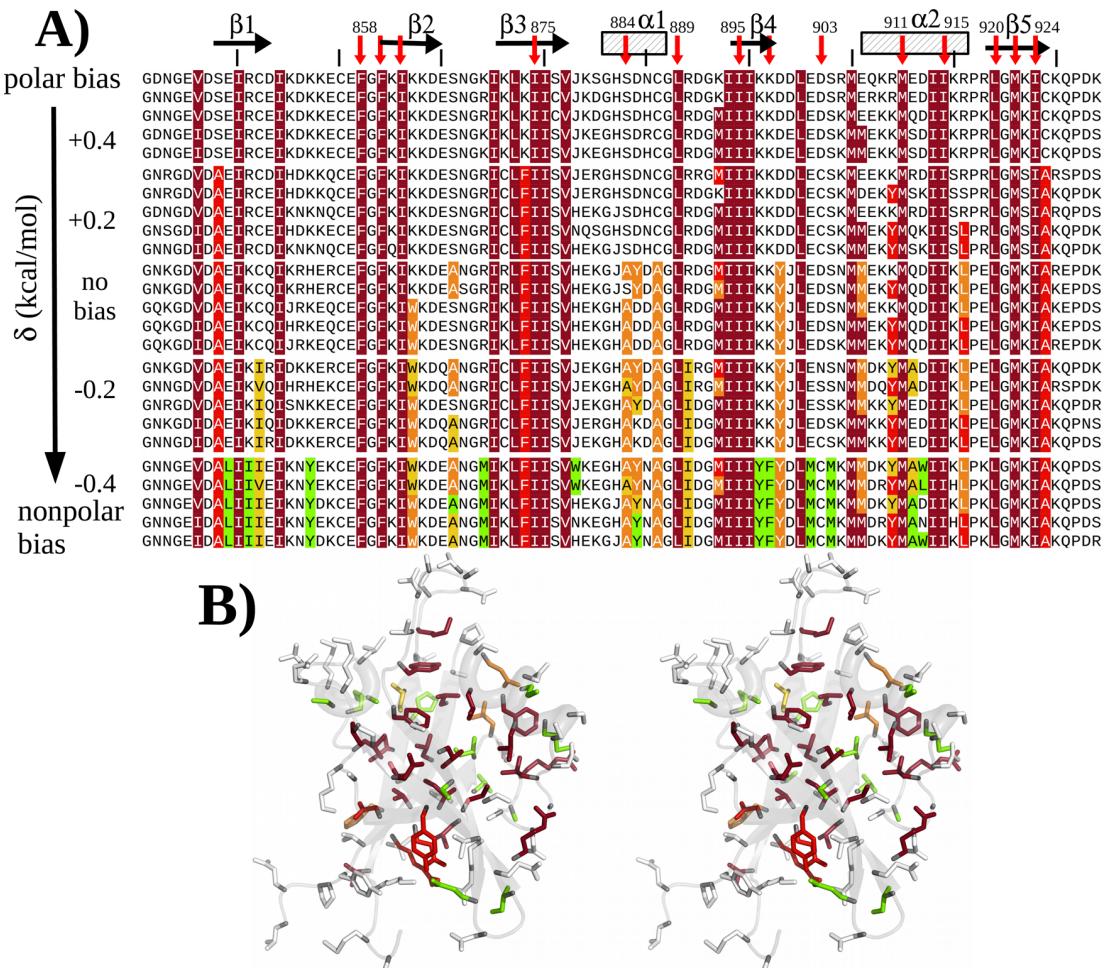


Figure 7. (A) Tiam1 sequences designed with different levels of hydrophobic bias. The top (respectively, bottom) sequences were obtained in the presence of a bias δ opposing (favoring) hydrophobic types. The middle sequences were obtained from Proteus simulations without any bias ($\delta = 0$). For each bias level, five low energy sequences are shown. Hydrophobic positions are colored according to the simulation where they appear first: from brick red (top) to light green (bottom). The 14 hydrophobic core positions are indicated by red arrows. (B) 3D Tiam1 structure (stereo) with residue colors as in (A). The backbone is shown in light gray.

826 amino acid positions in the Tiam1 PDZ domain known to
 827 contribute to peptide binding specificity. Modifying these four
 828 positions (quadruple mutant or QM) in the protein altered the
 829 binding specificity such that QM preferentially bound a Caspr4
 830 peptide relative to a syndecan-1 (Sdc1) peptide.^{70,73} The four
 831 mutations in the QM PDZ domain were L911M, K912E,
 832 L915F, and L920 V. All four positions but Lys912 are part of
 833 the conserved hydrophobic core. The four single and two
 834 double mutants were also characterized experimentally.^{70,73} For
 835 simplicity, we denote the native (WT) sequence as LKLL and
 836 the quadruple mutant (QM) as MEFV. Other variants are
 837 denoted similarly. Replica Exchange MC simulations were
 838 conducted on several structural templates, where all four
 839 positions could mutate simultaneously, into all amino acid types
 840 except Gly and Pro. We used the Proteus model with the lower
 841 dielectric constant, $\epsilon_p = 4$, which gave similarity scores
 842 equivalent to the $\epsilon_p = 8$ model but had a reduced tendency
 843 to bury polar side chains, thanks to its lower dielectric constant.
 844 In addition, no bias energy term was used, only the MMGBSA
 845 energy function. The CPD model used either the wildtype or
 846 the quadruple mutant crystal structure as backbone template
 847 for the design (PDB codes 4GVD and 4NXQ, respectively),
 848 shown in Figure 8. Although these two structures were

determined with the Sdc1 and Caspr4 ligands, they were used here for both holo *and* apo design simulations. The backbone rms deviation between these structures is 0.9 Å, with the main differences in the flexible 850–857 loop near the peptide C-terminus and in helix α_2 . This helix is pushed slightly outward in the mutant complex, to accommodate Phe side chains both at protein position 915 and at the peptide C-terminus. One expectation is that the mutant backbone model (4NXQ) will better describe variants with Phe at position 915 and the wildtype backbone model (4GVD) will better describe variants with a smaller 915 side chain.

We studied six systems: the Tiam1 PDZ domain with either its wildtype or QM backbone X-ray structure, with the syndecan1 or the Caspr4 peptide ligand or no ligand. Results are shown in Figure 8. For all six systems, the native or native-like amino acid types were sampled at all four designed positions. For example, using the wildtype backbone structure (4GVD), Leu911 was preserved in the apo simulations and changed to Ile or Val in the holo simulations. Similarly, holo simulations with the mutant backbone structure (4NXQ) sampled Ile, Leu, and Met. With the mutant backbone, holo simulations sampled somewhat different types at position 911 (Trp, Arg, Lys), which all appear in low amounts at the

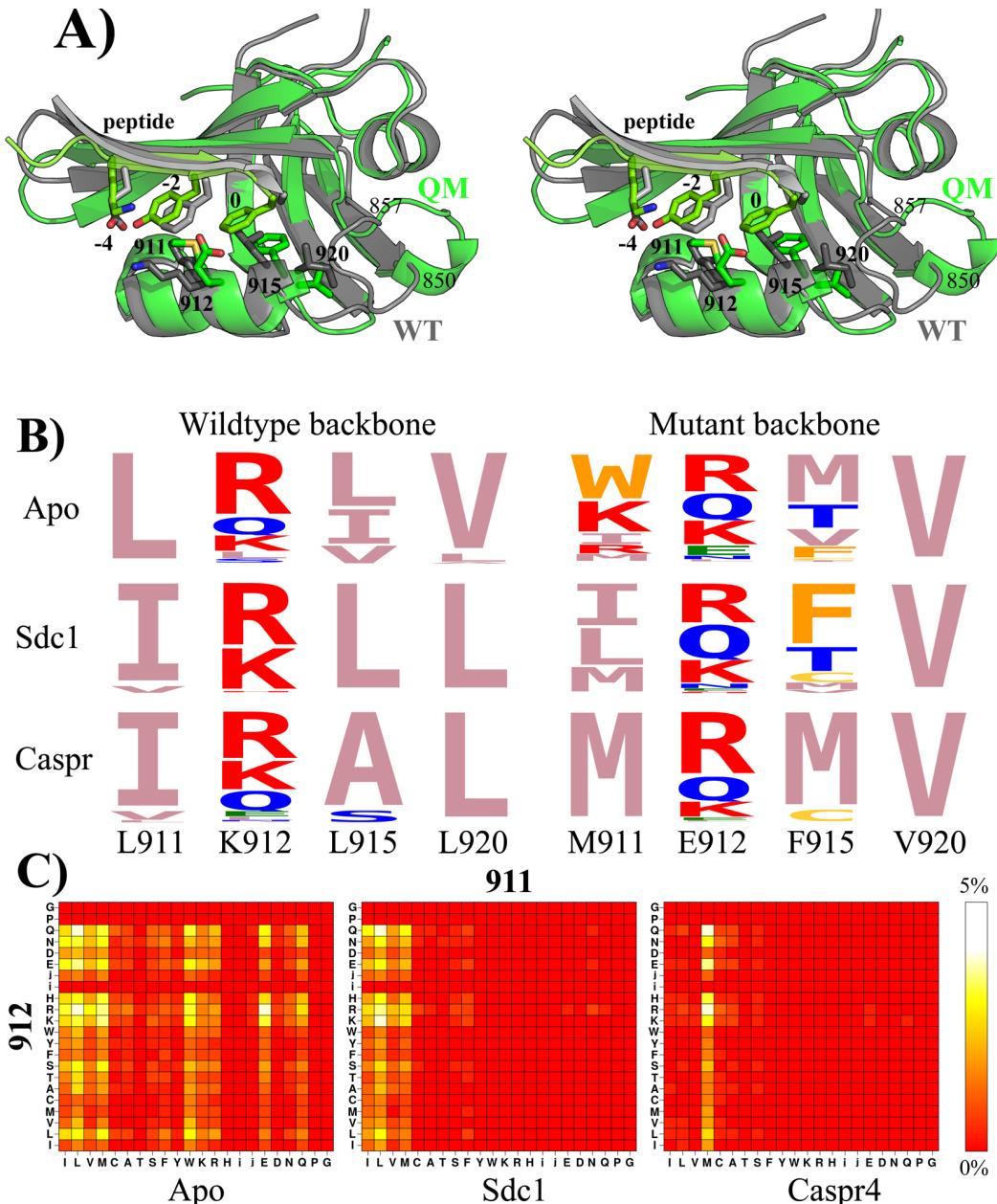


Figure 8. Design of four Tiam1 specificity positions. (A) X-ray structures (stereo) with the wildtype sequence (LKLL; labeled WT; PDB code 4GVD) or the quadruple mutant sequence (MEFV; labeled QM; PDB code 4NXQ), with bound Sdc1 and Caspr4, respectively. The four designed side chains are shown and labeled (both WT and QM mutant types). (B) Logo representation of designed sequences with no ligand (apo) or Sdc1 or Caspr4, using the wildtype (left) or quadruple mutant (right) X-ray structure. (C) Covariance plots for the 911–912 pair: populations of each pair of types are shown as levels of color, with yellow the most highly populated (5%) and red the lowest (0%). Results correspond to design simulations with the wildtype backbone.

corresponding position in the Pfam seed alignment. All the simulations sampled mostly Arg, Lys, Gln and occasionally Glu at position 912, and mostly Leu and Val at position 920, similar to the wildtype sequence. Not surprisingly, agreement with the wildtype sequence was better with the wildtype X-ray structure, while agreement with the mutant sequence was better with the mutant X-ray structure.

Recovery of the precise native and quadruple mutant sequences in the different states (apo and holo) was qualitative. Thus, using the wildtype backbone structure and in the apo state, the MC simulation recovered the wildtype sequence LKLL just 2 kcal/mol above the lowest energy sequence

(KKLV). The LKML homologue was the second best sequence overall, and the homologues IKLL and LKLV were just 1–2 kcal/mol higher in energy. The mutant sequence MEFV did not appear, nor did any close homologues, probably because the wildtype backbone structure cannot accommodate Phe at position 915. Similar results were obtained with the Sdc1 ligand, with the LKLL, IKLL, VKLL, and MKLL sequences all having energies just 1–2 kcal/mol above the best sequence. The MKLL:Sdc1 affinity is known experimentally, and is within 0.1 kcal/mol of the wildtype value.⁷³ Experimentally, the wildtype and mutant sequences have the same binding free energy for Caspr4, and stabilities just 2 kcal/mol apart,

896 suggesting that both should be sampled. Instead, neither was
897 sampled. The closest homologue sequence was IEAV (similar
898 to MEFV), at +2 kcal/mol (relative to the best sequence). This
899 was probably due to steric conflict between position 915 (L or
900 F) and the Caspr4 Phe0 in this backbone geometry.

901 Using the mutant backbone structure (4NXQ) and in the
902 apo state, the room temperature Monte Carlo replica recovered
903 the mutant sequence MEFV at an energy of +5 kcal/mol
904 (relative to the best sequence) and the wildtype sequence
905 LKLL at +7 kcal/mol. Both protein variants are thermodynamically
906 stable; a slightly higher energy to produce LKLL seems
907 reasonable, since the design simulation used the mutant
908 backbone structure, which presumably should favor MEFV.
909 With the Sdc1 ligand, MEFV appeared at an energy of +6 kcal/
910 mol, relative to the best sequence, which was the close wildtype
911 homologue IKLV. VKVL was just 3 kcal/mol higher. With the
912 Caspr4 ligand, the mutant sequence appeared at an energy of
913 +7 kcal/mol, compared to the best sequence, TKMV. Its
914 homologues MQMV and MEMV appeared at +5 kcal/mol.
915 The wildtype LKLL and its close homologues did not appear
916 (indicating poorer energies), while MAFI was the second best
917 sequence overall.

918 A more detailed comparison is possible with the binding
919 affinities of the experimentally characterized mutants.⁷³ The
920 experiments show that (1) affinity changes associated with each
921 position are roughly independent of the other positions
922 (coupling free energies of 0.4 kcal/mol or less between
923 positions); (2) homologous changes to Leu911, Leu915, and
924 Val920 have a very small effect on the affinity; (3) changing
925 Lys912 to Glu reduces binding by about 0.5–1 kcal/mol (for
926 both peptides, possibly due to lost interactions with the Lys
927 methylenes); (4) changing Leu915 to Phe affects binding
928 differently depending on the residue type at position 912 type
929 and the peptide. These properties are mostly reproduced by our
930 simulations. With the wildtype backbone model, considering
931 sequences of the form NKNN (where N ∈ {I,L,V,M}), the
932 mean apo and Sdc1-bound energies are 0.9 ± 0.6 and 1.1 ± 0.5
933 kcal/mol, respectively, which leads to a mean affinity of $0.2 \pm$
934 0.8 kcal/mol (relative to IKLL, taken as a reference): mutations
935 between the amino acid types I, L, V, and M (N to N'
936 mutations) change the Sdc1 affinity very little, consistent with
937 experiment. Comparing the apo and holo energies sampled in
938 our design simulations, we predict that NKNN → NENN
939 mutations lead to affinity changes of +0.75 kcal/mol for both
940 peptides, compared to 0.94 kcal/mol (Sdc1) and 0.55 kcal/mol
941 (Caspr4) experimentally. Similarly, we predict that NKNN →
942 NKFN mutations reduce the affinity by 0.5 kcal/mol for both
943 peptides, compared to 1.2 and 0.8 kcal/mol experimentally.
944 Only for NENN → NEFN mutations do we see larger errors:
945 we predict a 0.5 kcal/mol affinity loss for Sdc1 (vs no loss
946 experimentally) and a 0.9 kcal/mol loss for Caspr4 (vs a 0.5
947 kcal/mol gain experimentally). Specificity changes are predicted
948 to be small, in qualitative agreement with experiment. For
949 example the MKFV → MEFV mutation favors Caspr4, relative
950 to Sdc1, by 0.2 kcal/mol, compared to 0.5 kcal/mol
951 experimentally for the homologous LKLL → LELL mutation.
952 The simulations also gave information on correlations
953 between the four mutating positions. Figure 8C shows
954 covariance plots between positions 911 and 912 for the apo
955 and holo simulations. Position 911 was more diverse in the apo
956 than in either holo state (Sdc1 or Caspr4), while 912 was not
957 very sensitive to the peptide. The computed pairwise
958 correlations among all four protein positions were weak, so

that the covariance plots mostly exhibit horizontal and vertical
959 lines or bands, without noticeable “diagonal” features. This
960 agrees with the experimental affinities of the single, double, and
961 quadruple mutants, where the affinity changes associated with
962 each point mutation were only weakly coupled to the other
963 positions.⁷³

5. DISCUSSION

5.1. Model Limitations. We have parametrized a simple
965 CPD model for PDZ design, suitable for high-throughput
966 design applications and implemented in the Proteus software.
967 For the folded state representation, we use a high-quality
968 protein force field and Generalized Born solvent model. We
969 tested two protein dielectric constants ϵ_p . We used a specific set
970 of X-ray structures for our test proteins, each with a specific
971 backbone conformation. For the side chains, we used a simple,
972 discrete rotamer library and a short minimization of each side
973 chain pair during the energy matrix calculation to alleviate the
974 discrete rotamer approximation. Both the energy function and
975 the rotamer description have been extensively tested and shown
976 to give very good performance for side chain reconstruction
977 tests⁵⁴ (comparable to the popular Scwrl4 program⁵³) and
978 good performance for a large set of protein acid/base
979 constants⁷⁴ (superior to the Rosetta software,⁷⁵ despite its
980 extensive *ad hoc* parameter tuning).

The unfolded state representation used a simple, implicit
981 model, characterized by a set of empirical, amino acid chemical
982 potentials or reference energies. These energies were chosen by
983 a likelihood maximization procedure, formulated here, in order
984 to reproduce the amino acid composition of carefully selected
985 natural homologues. The unfolded state description used here
986 is more refined than previously,⁷⁶ since distinct reference
987 energy values were used for amino acid positions that are
988 buried or exposed in the *folded* state. This method assumes that
989 there is residual structure in the unfolded state, with some
990 positions more buried than others. Furthermore, it should make
991 the parametrization more robust and less sensitive to the size
992 and structure of the natural homologues used to define the
993 target amino acid compositions, because the amino acid
994 frequencies of exposed and buried regions are averaged
995 separately. In principle, this doubles the number of adjustable
996 reference energies. However, we reduced this number by
997 introducing amino acid similarity classes, with one adjustable
998 reference energy per class. To optimize the reference energies,
999 we performed design calculations for each test protein (apo
1000 state) where half of the amino positions could mutate at a time
1001 (excluding Gly and Pro), with distinct simulations for each half.
1002 This way, during parameter optimization, a mutating position
1003 was always surrounded by an environment at least 50%
1004 identical to the wildtype sequence. The design calculations
1005 relied on a powerful and efficient Replica Exchange Monte
1006 Carlo exploration method that used over a half billion steps per
1007 simulation (per replica), and produced thousands of designed
1008 sequences in a single simulation. Reference energy values were
1009 optimized with two different choices for the protein dielectric
1010 constant ϵ_p . The performance levels were similar for both
1011 values.

The model has several limitations, most of which are
1012 widespread in CPD implementations and applications. The first
1013 is the use of protein stability as the sole design criterion,
1014 without explicitly accounting for fold specificity,⁷⁷ protection
1015 against aggregation, or functional considerations like ligand
1016 binding. We note, however, that the Superfamily tests did not
1017

lead to any fold misassignments (sequences that prefer another SCOP fold), so that in practice, fold specificity was achieved. Functional criteria can also be introduced in an *ad hoc* way; for example, the sequences tested by MD were designed with 11 peptide binding residues fixed, to facilitate future experimental studies.

A second model limitation is the use of a fixed protein backbone. In fact, the backbone is not really fixed: rather, certain motions are allowed but modeled *implicitly*, through the use of a protein dielectric constant greater than 1 ($\epsilon_p = 4$ or 8).⁷⁸ This dielectric value means that the protein structure (including its backbone) is allowed to relax or reorganize in response to charge redistribution associated with mutations or side chain rotamer changes. However, the reorganization is modeled implicitly, not explicitly,⁷⁸ and it does not involve motion of the atomic centers or their associated van der Waals spheres. Thus, the backbone cannot reorganize in response to steric repulsion produced by mutations or rotamer changes, nor can it shift to fill space left empty by a mutation. The effect of this approximation was apparent in the design of the four Tiam1 specificity positions, where the designed sequences were sensitive to the particular backbone conformation of the protein and peptide. Specifically, with the wildtype backbone structure, there was no room to insert a Phe side chain at position 915, even though Phe915 is present in the experimental quadruple mutant (which has a slightly different backbone structure). Therefore, the choice of the initial X-ray structural model is important, and several strategies are possible. Here, to parametrize the CPD model, we used X-ray structures solved with a peptide ligand, even though the parametrization simulations and most of the testing were done for the apo proteins. This choice was made partly because the apo/holo PDZ structures are quite similar and partly to make the model more transferable and facilitate applications to peptide binding. Another strategy could have been to parametrize the model using all apo structures, then switch to holo structures for the Tiam1 application.

For whole protein design (such as the hydrophobic titration application), the use of a fixed backbone can be partly counterbalanced by designing two or more PDZ structures. For example, pooling the designed Tiam1 and Cask sequences gave a mean sequence entropy comparable to the experimental Pfam set, and allowed us to recapitulate more sequences than design with just one backbone. In the application to Tiam1 4-position design, the fixed backbone was also counterbalanced by doing calculations separately with two different backbone structures, a holo wildtype and a holo mutant structure. Simulations with the mutant backbone allowed us to obtain mutants having Phe at position 915. Notice that a new method for multibackbone design was recently developed in Proteus, based on a novel, nonheuristic hybrid Monte Carlo method that preserves Boltzmann sampling.³⁵ This method could be applied in the future.

A third limitation of our model is the need, for optimal results, to parametrize the reference energies specifically for a given set of proteins. This step is well-automated and highly parallel. However, it involves several choices that are partly arbitrary. These include the choice of a set of protein domains to represent the protein or family of interest. We also need to choose a similarity threshold to define the target homologues from which we compute the experimental amino acid compositions. Here, we chose to use close homologues of each family member, compute their compositions, then average

over the two family representatives. This method worked well but other choices are possible, and more work is needed to draw definitive conclusions. Also, the monoposition design of Tiam1 showed evidence of some systematic error (Figure 4), with a large fraction of Arg, Lys, and Gln residues types on the protein surface, despite the optimized reference energies. In the future, it may be necessary to relax the intragroup constraints toward the end of the reference energy optimization and/or target smaller numbers of mutating positions, instead of one-half of the protein at a time.

A fourth limitation of our model is the discrete rotamer approximation, which requires some adaptation of the energy function to avoid exaggerated steric clashes; the method used here is the residue-pair minimization method described earlier.^{26,76} A fifth limitation is the use of a pairwise additive solvation model (as in most CPD models). Specifically, the dielectric environment of each residue pair is assumed here to be native-like (so-called “Native Environment approximation” or NEA^{74,76}). This leads to an energy function that has the form of a sum over pairs of residues and that can be precalculated and stored in an energy matrix, which then serves as a lookup table during the subsequent Monte Carlo simulations. Despite this approximation, the model gave good results for a large acid/base benchmark, a problem that is very sensitive to the electrostatic treatment.⁷⁴

Some of these limitations could be removed in future work. In particular, since the energy function is mostly physics-based, it can benefit rapidly from ongoing improvements in protein force fields and solvation models. Thus, the NEA approximation could be removed in the future due to the recent implementation (manuscript in preparation) of a more exact Generalized Born calculation, whose efficiency is comparable to the pairwise approximation.⁷⁹ We have also implemented an improved model for hydrophobic solvation,⁸⁰ which is faster and more accurate than our current surface area energy term (manuscript in preparation).

5.2. Model Testing and Applications. Designed sequences were extensively compared to natural sequences, through fold recognition tests, similarity calculations, and entropy calculations. In the test simulations, we designed the entire protein sequence, so that all positions (except Gly and Pro) could mutate freely, subject only to an overall bias toward the mean, experimental amino acid composition (through the reference energies). Despite the lack of experimental bias or constraints, the resulting sequences had a high overall similarity to the natural, Pfam sequences, as measured by the Blosum40 similarity scores. The scores obtained were mostly comparable to the similarity scores between pairs of Pfam sequences themselves. Thus, the sequences designed with Proteus resemble moderately distant natural homologues. The similarity was very strong for residues in the core of the protein, as observed in previous CPD studies.^{30,72} In contrast, for residues at the protein surface, similarity scores were close to zero, the score one would obtain if one picked amino acid types randomly. Notice that many surface residues are involved in functional interactions, such as the 11 peptide-binding residues in PDZ domains. Surface residues are also selected by evolution to avoid aggregation or unwanted adhesion. Most of these functional constraints are not explicitly accounted for in our design protocol (although the energy function indirectly favors protein solubility). Despite the limited similarity scores for surface regions, fold recognition with the Superfamily tool and the best design models was almost perfect. Earlier fold

recognition tests that used a simpler energy function gave a lower fold recognition rate of about 85% (for a larger and more diverse test set) and lower similarities.^{15,50} Evidently, the combined use of an improved protein force field, Generalized Born solvent, and family specific reference energies leads to designed sequences that are more native-like and presumably better.

The Proteus sequences were also compared to sequences designed with the Rosetta software (using default parameters), which has itself been extensively tested. On the basis of the Blosum similarity scores (vs natural sequences in Pfam) and the fold recognition tests, the Proteus and Rosetta sequences appear to be of about the same quality. However, Rosetta makes fewer mutations than Proteus, so that the identity scores, compared to the corresponding wildtype protein, are about 6% points higher. This means that Proteus mutates about five more positions, on average, per PDZ domain. This number could easily be reduced, by adding to the Proteus energy function an explicit bias energy term that increases with the number of mutations (away from the wildtype sequence). An equivalent bias energy was used above for just two simulations of Tiam1 and Cask (see the “biased Proteus” results in the two upper panels of Figure 5), to illustrate the possibility of using experimentally restrained sampling. It remains to be seen whether a restraint based on the identity score would lead to more stable and realistic designed sequences.

Another attractive route for testing designed sequences is through high-level MD simulations. Here, 10 designed Tiam1 sequences were tested in rather long MD simulations, in the apo form, using the same protein force field as in the CPD model (Amber force field) and an established explicit water model. These sequences were designed using Proteus, with Gly, Pro, and 11 peptide-binding positions held fixed but all others free to mutate. Six of the sequences remained stable over 200 ns simulation lengths and two were extended up to a microsecond, which represents a very stringent test of the designs. Sequence 6 was stable over the whole microsecond. The mean deviation of seq-6 from the starting, experimental wildtype structure was 1 Å, which is the same as the mean deviation in the MD simulation of the wildtype sequence itself. The deviation between the mean sequence 6 MD structure and the mean wildtype MD structure was also small, just 1.2 Å. Sequence 2 was also simulated and remained stable until just before the end of the microsecond simulation, at which point it underwent a larger fluctuation. The fluctuation regressed 100 ns later. An even longer simulation would be needed to determine if this fluctuation is harmless or signals the beginning of domain unfolding. Note that in the presence of a peptide ligand, we expect the structural stability of the designed domains would increase further. MD simulations of additional designed sequences would also be of interest. Direct experimental testing of the designed proteins remains to be done.

The CPD model was used for two applications. “Hydrophobic titration” of two PDZ domains illustrated a novel way to characterize protein designability. The cost or availability of hydrophobic side chain types was controlled by a bias energy term that was gradually varied. The mean overall hydrophobic “susceptibility”, the number of type of changes per kcal/mol and per position, differed by about 20% between Tiam1 and Cask. In Tiam1, 11 of the core positions remained hydrophobic even with the largest bias value favoring polar types, while 14 other positions changed to nonpolar types at the largest

nonpolar bias energy value. A comparison to other domain families would be of interest, and is left for future work.

Redesign of four specificity positions in Tiam1 allowed us to test the design model in a different way. It revealed some of the limitations of fixed backbone design, but also gave semi-quantitative agreement with available binding free energies. This agreement predicts new mutations that could be of interest for obtaining new specificity properties. They remain to be studied further and tested experimentally. Here, the apo and two holo states were studied, and designed separately. Information about binding affinities and binding specificity were then obtained by comparing the energies sampled in the different simulations. In the future, we would like to include binding affinity and/or specificity directly in the design calculations, as a property to be designed for or against within a single simulation. In addition, we should allow different backbone structures for the apo and each holo system. This could be done in the future, since recent hybrid Monte Carlo schemes^{35,81} can be used for multibackbone design, and can be extended to the problem of designing ligand binding specificity. We also note that since our energy function is physics-based, it has transferability to a range of molecule types, such as nonnatural amino acids (considered in an earlier protein–ligand design study³⁴). Such amino acids could be of interest for designing PDZ peptide ligands, to provide additional diversity and perhaps enhanced resistance to proteolysis.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: [10.1021/acs.jctc.6b01255](https://doi.org/10.1021/acs.jctc.6b01255).

Detailed description of the model cross validation and results obtained with a protein dielectric $\epsilon_p = 4$ ([PDF](#))

AUTHOR INFORMATION

Corresponding Author

*E-mail: thomas.simonson@polytechnique.fr.

ORCID

Nicolas Panel: [0000-0001-8782-0586](https://orcid.org/0000-0001-8782-0586)

Thomas Simonson: [0000-0002-5117-7338](https://orcid.org/0000-0002-5117-7338)

Author Contributions

[§]These authors contributed equally to the manuscript.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We are grateful for discussions with Michael Schnieders and Young Joo Sun (University of Iowa). Some of the calculations were done at the CINES supercomputer center of the French Ministry of Research. NAMD was developed by the Theoretical and Computational Biophysics Group in the Beckman Institute at the University of Illinois at Urbana.

REFERENCES

- (1) Harris, B. Z.; Lim, W. A. Mechanism and role of PDZ domains in signaling complex assembly. *J. Cell Sci.* **2001**, *114*, 3219–3231.
- (2) Hung, A. Y.; Sheng, M. PDZ Domains: Structural Modules for Protein Complex Assembly. *J. Biol. Chem.* **2002**, *277*, 5699–5702.
- (3) Tonikian, R.; Zhang, Y. N.; Sazinsky, S. L.; Currell, B.; Yeh, J. H.; Reva, B.; Held, H. A.; Appleton, B. A.; Evangelista, M.; Wu, Y.; Xin, X. F.; Chan, A. C.; Seshagiri, S.; Lasky, L. A.; Sander, C.; Boone, C.;

- 1266 Bader, G. D.; Sidhu, S. S. A Specificity Map for the PDZ Domain
1267 Family. *PLoS Biol.* **2008**, *6*, 2043–2059.
(4) Gfeller, D.; Butty, F.; Wierzbicka, M.; Verschueren, E.; Vanhee,
1268 P.; Huang, H.; Ernst, A.; Darand, N.; Stagljar, I.; Serrano, L.; Sidhu, S.
1269 S.; Bader, G. D.; Kim, P. M. The multiple-specificity landscape of
1270 modular peptide recognition domains. *Mol. Syst. Biol.* **2011**, *7*, art484.
(5) Subbiah, V. K.; Kranjec, C.; Thomas, M.; Banks, L. PDZ
1271 domains: the building blocks regulating tumorigenesis. *Biochem. J.*
1272 **2011**, *439*, 195–205.
(6) Shepherd, T. R.; Fuentes, E. J. Structural and thermodynamic
1273 analysis of PDZ-ligand interactions. *Methods Enzymol.* **2011**, *488*, 81–
1277 100.
(7) Bacha, A.; Clausen, B. H.; Moller, M.; Vestergaard, B.; Chic, C.
1278 N.; Round, A.; Sørensen, P. L.; Nissen, K. B.; Kastrup, J. S.; Gajhede,
1279 M.; Jemth, P.; Kristensen, A. S.; Lundstrom, P.; Lambertsen, K. L.;
1280 Stromgaard, K. A high-affinity, dimeric inhibitor of PSD-95 bivalently
1281 interacts with PDZ1–2 and protects against ischemic brain damage.
1282 *Proc. Natl. Acad. Sci. U. S. A.* **2012**, *109*, 3317–3322.
(8) Roberts, K. E.; Cushing, P. R.; Boisguerin, P.; Madden, D. R.;
1283 Donald, B. R. Computational Design of a PDZ Domain Peptide
1284 Inhibitor that Rescues CFTR Activity. *PLoS Comput. Biol.* **2012**, *8*,
1285 e1002477.
(9) Zheng, F.; Jewell, H.; Fitzpatrick, J.; Zhang, J.; Mierke, D. F.;
1286 Grigoryan, G. Computational Design of Selective Peptides to
1287 Discriminate between Similar PDZ Domains in an Oncogenic
1288 Pathway. *J. Mol. Biol.* **2015**, *427*, 491–510.
(10) Lockless, W.; Ranganathan, R. Evolutionary Conserved
1289 Pathways of Energetic Connectivity in Protein Families. *Science*
1290 **1999**, *285*, 295–299.
(11) Kong, Y.; Karplus, M. Signaling pathways of PDZ2 domain: A
1291 molecular dynamics interaction correlation analysis. *Proteins: Struct.,
1292 Funct., Genet.* **2009**, *74*, 145–154.
(12) McLaughlin, R. N., Jr; Poelwijk, F. J.; Raman, A.; Gosal, W. S.;
1293 Ranganathan, R. The spatial architecture of protein function and
1294 adaptation. *Nature* **2012**, *491*, 138–142.
(13) Melero, C.; Ollikainen, N.; Harwood, I.; Karpiai, J.; Kortemme,
1295 T. Quantification of the transferability of a designed protein specificity
1296 switch reveals extensive epistasis in molecular recognition. *Proc. Natl.
1297 Acad. Sci. U. S. A.* **2014**, *111*, 15426–15431.
(14) Reina, J.; Lacroix, E.; Hobson, S. D.; Fernandez-Ballester, G.;
1298 Rybin, V.; Schwab, M. S.; Serrano, L.; Gonzalez, C. Computer-aided
1299 design of a PDZ domain to recognize new target sequences. *Nat.
1300 Struct. Biol.* **2002**, *9*, 621–627.
(15) Schmidt am Busch, M.; Sedano, A.; Simonson, T. Computational
1301 protein design: validation and possible relevance as a tool for
1302 homology searching and fold recognition. *PLoS One* **2010**, *5* (5),
1303 e10410.
(16) Smith, C. A.; Kortemme, T. Structure-Based Prediction of the
1304 Peptide Sequence Space Recognized by Natural and Synthetic PDZ
1305 Domains. *J. Mol. Biol.* **2010**, *402*, 460–474.
(17) Butterfoss, G. L.; Kuhlman, B. Computer-based design of novel
1306 protein structures. *Annu. Rev. Biophys. Biomol. Struct.* **2006**, *35*, 49–65.
(18) Lippow, S. M.; Tidor, B. Progress in computational Protein
1307 Design. *Curr. Opin. Biotechnol.* **2007**, *18*, 305–311.
(19) Dai, L.; Yang, Y.; Kim, H. R.; Zhou, Y. Improving computational
1308 protein design by using structure-derived sequence profile. *Proteins:
1309 Struct., Funct., Genet.* **2010**, *78*, 2338–2348.
(20) Feldmeier, K.; Hoecker, B. Computational protein design of
1310 ligand binding and catalysis. *Curr. Opin. Chem. Biol.* **2013**, *17*, 929–
1311 933.
(21) Tinberg, C. E.; Khare, S. D.; Dou, J.; Doyle, L.; Nelson, J. W.;
1312 Schena, A.; Jankowski, W.; Kalodimos, C. G.; Johnsson, K.; Stoddard,
1313 B. L.; Baker, D. Computational design of ligand-binding proteins with
1314 high affinity and selectivity. *Nature* **2013**, *501*, 212–218.
(22) Au, L.; Green, D. F. Direct Calculation of Protein Fitness
1315 Landscapes through Computational Protein Design. *Biophys. J.* **2016**,
1316 *110*, 75–84.
(23) Pokala, N.; Handel, T. Energy functions for protein design I: Efficient and accurate continuum electrostatics and solvation. *Protein Sci.* **2004**, *13*, 925–936.
(24) Samish, I.; MacDermaid, C. M.; Perez-Aguilar, J. M.; Saven, J. G. Theoretical and computational protein design. *Annu. Rev. Phys. Chem.* **2011**, *62*, 129–149.
(25) Li, L.; Francklyn, C.; Carter, C. W. Aminoacylating Urzymes Challenge the RNA World Hypothesis. *J. Biol. Chem.* **2013**, *288*, 26856–26863.
(26) Schmidt am Busch, M.; Lopes, A.; Mignon, D.; Simonson, T. Computational protein design: software implementation, parameter optimization, and performance of a simple model. *J. Comput. Chem.* **2008**, *29*, 1092–1102.
(27) Simonson, T. Protein:ligand recognition: simple models for electrostatic effects. *Curr. Pharm. Des.* **2013**, *19*, 4241–4256.
(28) Polydorides, S.; Michael, E.; Mignon, D.; Druart, K.; Archontis, G.; Simonson, T. In *Methods in Molecular Biology: Design and Creation of Protein Ligand Binding Proteins*; Stoddard, B., Ed.; Springer Verlag: New York, 2016; Vol. 1414; p 0000.
(29) Kuhlman, B.; Baker, D. Native protein sequences are close to optimal for their structures. *Proc. Natl. Acad. Sci. U. S. A.* **2000**, *97*, 10383–10388.
(30) Dantas, G.; Kuhlman, B.; Callender, D.; Wong, M.; Baker, D. A Large Test of Computational Protein Design: Folding and Stability of Nine Completely Redesigned Globular Proteins. *J. Mol. Biol.* **2003**, *332*, 449–460.
(31) Rohl, C. A.; Strauss, C. E. M.; S, M. K. M.; Baker, D. Protein structure prediction using Rosetta. *Methods Enzymol.* **2004**, *383*, 10383–103866–93.
(32) Lazaridis, T.; Karplus, M. Effective energy function for proteins in solution. *Proteins: Struct., Funct., Genet.* **1999**, *35*, 133–152.
(33) Mignon, D.; Simonson, T. Comparing three stochastic search algorithms for computational protein design: Monte Carlo, Replica Exchange Monte Carlo, and a multistart, steepest-descent heuristic. *J. Comput. Chem.* **2016**, *37*, 1781–1793.
(34) Druart, K.; Palmai, Z.; Omarjee, E.; Simonson, T. Protein:ligand binding free energies: a stringent test for computational protein design. *J. Comput. Chem.* **2016**, *37*, 404–415.
(35) Druart, K.; Bigot, J.; Audit, E.; Simonson, T. A hybrid Monte Carlo method for multibackbone protein design. *J. Chem. Theory Comput.* **2016**, *12*, 6035–6048.
(36) Gaillard, T.; Simonson, T. Full protein sequence redesign with an MMGBSA energy function. *J. Comput. Chem.* **2017**.
(37) Baker, D. Prediction and design of macromolecular structures and interactions. *Philos. Trans. R. Soc., B* **2006**, *361*, 459–463.
(38) Frenkel, D.; Smit, B. *Understanding molecular simulation*; Academic Press: New York, 1996; Chapter 3.
(39) Grimmett, G. R.; Stirzaker, D. R. *Probability and random processes*; Oxford University Press: Oxford, United Kingdom, 2001.
(40) Kleinman, C. L.; Rodrigue, N.; Bonnard, C.; Philippe, H.; Lartillot, N. A maximum likelihood framework for protein design. *BMC Bioinf.* **2006**, *7*, Art326.
(41) Fowler, R. H.; Guggenheim, E. A. *Statistical Thermodynamics*; Cambridge University Press: Cambridge, United Kingdom, 1939.
(42) Cornell, W.; Cieplak, P.; Bayly, C.; Gould, I.; Merz, K.; Ferguson, D.; Spellmeyer, D.; Fox, T.; Caldwell, J.; Kollman, P. A. Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *J. Am. Chem. Soc.* **1995**, *117*, 5179–5197.
(43) Aleksandrov, A.; Polydorides, S.; Archontis, G.; Simonson, T. Predicting the Acid/Base Behavior of Proteins: A Constant-pH Monte Carlo Approach with Generalized Born Solvent. *J. Phys. Chem. B* **2010**, *114*, 10634–10648.
(44) Hawkins, G. D.; Cramer, C.; Truhlar, D. Pairwise descreening of solute charges from a dielectric medium. *Chem. Phys. Lett.* **1995**, *246*, 122–129.
(45) Lopes, A.; Aleksandrov, A.; Bathelt, C.; Archontis, G.; Simonson, T. Computational sidechain placement and protein mutagenesis with

- 1401 implicit solvent models. *Proteins: Struct., Funct., Genet.* **2007**, *67*, 853–
1402 867.
- 1403 (46) Lee, B.; Richards, F. The interpretation of protein structures:
1404 estimation of static accessibility. *J. Mol. Biol.* **1971**, *55*, 379–400.
- 1405 (47) Brünger, A. T. *X-PLOR version 3.1, A System for X-ray*
1406 *crystallography and NMR*; Yale University Press: New Haven, 1992.
- 1407 (48) Gaillard, T.; Simonson, T. Pairwise Decomposition of an
1408 MMGBSA Energy Function for Computational Protein Design. *J.*
1409 *Comput. Chem.* **2014**, *35*, 1371–1387.
- 1410 (49) Street, A. G.; Mayo, S. Pairwise calculation of protein solvent-
1411 accessible surface areas. *Folding Des.* **1998**, *3*, 253–258.
- 1412 (50) Schmidt am Busch, M.; Mignon, D.; Simonson, T. Computational
1413 protein design as a tool for fold recognition. *Proteins: Struct.,*
1414 *Funct., Genet.* **2009**, *77*, 139–158.
- 1415 (51) Eswar, N.; Marti-Renom, M. A.; Webb, B.; Madhusudhan, M. S.;
1416 Eramian, D.; Shen, M.; Pieper, U.; Sali, A. Comparative Protein
1417 Structure Modeling With MODELLER. *Curr. Prot. Bioinf.* **2006**, *Suppl.*
1418 *15*, S.6.1–S.6.30.
- 1419 (52) Tuffery, P.; Etchebest, C.; Hazout, S.; Lavery, R. A New
1420 Approach to the Rapid Determination of Protein Side Chain
1421 Conformations. *J. Biomol. Struct. Dyn.* **1991**, *8*, 1267–1289.
- 1422 (53) Krivov, G. G.; Shapalov, M. V.; Dunbrack, R. L. Improved
1423 prediction of protein side-chain conformations with SCWRL4.
1424 *Proteins: Struct., Funct., Genet.* **2009**, *77*, 778–795.
- 1425 (54) Gaillard, T.; Panel, N.; Simonson, T. Protein sidechain
1426 conformation predictions with an MMGBSA energy function. *Proteins:*
1427 *Struct., Funct., Genet.* **2016**, *84*, 803–819.
- 1428 (55) Kofke, D. A. On the acceptance probability of replica-exchange
1429 Monte Carlo trials. *J. Chem. Phys.* **2002**, *117*, 6911–6914.
- 1430 (56) Earl, D.; Deem, M. W. Parallel tempering: theory, applications,
1431 and new perspectives. *Phys. Chem. Chem. Phys.* **2005**, *7*, 3910–3916.
- 1432 (57) Gough, J.; Karplus, K.; Hughey, R.; Chothia, C. Assignment of
1433 homology to genome sequences using a library of hidden Markov
1434 models that represent all proteins of known structure. *J. Mol. Biol.*
1435 **2001**, *313*, 903–919.
- 1436 (58) Wilson, D.; Madera, M.; Vogel, C.; Chothia, C.; Gough, J. The
1437 SUPERFAMILY database in 2007: families and functions. *Nucleic*
1438 *Acids Res.* **2007**, *35*, D308–D313.
- 1439 (59) Andreeva, A.; Howorth, D.; Brenner, S. E.; Hubbard, J. J.;
1440 Chothia, C.; Murzin, A. G. SCOP database in 2004: refinements
1441 integrate structure and sequence family data. *Nucleic Acids Res.* **2004**,
1442 *32*, D226–229.
- 1443 (60) Durbin, R.; Eddy, S. R.; Krogh, A.; Mitchison, G. *Biological*
1444 *sequence analysis*; Cambridge University Press: Cambridge, United
1445 Kingdom, 2002.
- 1446 (61) Murphy, L. R.; Wallqvist, A.; Levy, R. M. Simplified amino acid
1447 alphabets for protein fold recognition and implications for folding.
1448 *Protein Eng., Des. Sel.* **2000**, *13*, 149–152.
- 1449 (62) Launay, G.; Mendez, R.; Wodak, S. J.; Simonson, T.
1450 Recognizing protein-protein interfaces with empirical potentials and
1451 reduced amino acid alphabets. *BMC Bioinf.* **2007**, *8*, 270–291.
- 1452 (63) Jo, S.; Kim, T.; Iyer, V. G.; Im, W. CHARMM-GUI: a web-
1453 based graphical user interface for CHARMM. *J. Comput. Chem.* **2008**,
1454 *29*, 1859–1865.
- 1455 (64) Nose, S. A unified formulation of the constant temperature
1456 molecular dynamics method. *J. Chem. Phys.* **1984**, *81*, 511–519.
- 1457 (65) Hoover, W. G. Canonical dynamics: equilibrium phase-space
1458 distributions. *Phys. Rev. A: At., Mol., Opt. Phys.* **1985**, *31*, 1695–1697.
- 1459 (66) Darden, T. In *Computational Biochemistry & Biophysics*; Becker,
1460 O., Mackerell, A., Jr., Roux, B., Watanabe, M., Eds.; Marcel Dekker:
1461 N.Y., 2001; Chapter 4.
- 1462 (67) Jorgensen, W.; Chandrasekhar, J.; Madura, J.; Impey, R.; Klein,
1463 M. Comparison of simple potential functions for simulating liquid
1464 water. *J. Chem. Phys.* **1983**, *79*, 926–935.
- 1465 (68) Brooks, B.; Brooks, C. L., III; Mackerell, A. D., Jr.; Nilsson, L.;
1466 Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch,
1467 S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.;
1468 Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.;
1469 Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer,
1470 York, D. M.; Karplus, M. CHARMM: The biomolecular simulation
1471 program. *J. Comput. Chem.* **2009**, *30*, 1545–1614.
- 1472 (69) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid,
1473 E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. Scalable
1474 molecular dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781–1475
1476 1802.
- 1477 (70) Liu, X.; Speckhard, D. C.; Shepherd, T. R.; Sun, Y. J.; Hengel, S.;
1478 Yu, L.; Fowler, C. A.; Gakhar, L.; Fuentes, E. J. Distinct roles for
1479 conformational dynamics in protein-ligand interactions. *Structure* **2016**,
1480 *24*, 2053–2066.
- 1481 (71) Madera, M.; Vogel, C.; Kummerfeld, S. K.; Chothia, C.; Gough,
1482 J. The SUPERFAMILY database in 2004: additions and improvements.
1483 *Nucleic Acids Res.* **2004**, *32*, D235–D239.
- 1484 (72) Jaramillo, A.; Wernisch, L.; Héry, S.; Wodak, S. Folding free
1485 energy function selects native-like protein sequences in the core but
1486 not on the surface. *Proc. Natl. Acad. Sci. U. S. A.* **2002**, *99*, 13554–1486
1487 13559.
- 1488 (73) Shepherd, T. R.; Hard, R. L.; Murray, A. M.; Pei, D.; Fuentes, E.;
1489 J. Distinct Ligand Specificity of the Tiam1 and Tiam2 PDZ Domains.
1490 *Biochemistry* **2011**, *50*, 1296–1308.
- 1491 (74) Polydorides, S.; Simonson, T. Monte Carlo simulations of
1492 proteins at constant pH with generalized Born solvent, flexible
1493 sidechains, and an effective dielectric boundary. *J. Comput. Chem.* **2013**,
1494 *34*, 2742–2756.
- 1495 (75) Kilambi, K.; Gray, J. J. Rapid calculation of protein pK_a values
1496 using Rosetta. *Biophys. J.* **2012**, *103*, 587–595.
- 1497 (76) Simonson, T.; Gaillard, T.; Mignon, D.; Schmidt am Busch, M.;
1498 Lopes, A.; Amara, N.; Polydorides, S.; Sedano, A.; Archontis, K. D. G.
1499 Computational protein design: the Proteus software and selected
1500 applications. *J. Comput. Chem.* **2013**, *34*, 2472–2484.
- 1501 (77) Mach, P.; Koehl, P. Capturing protein sequence-structure
1502 specificity using computational sequence design. *Proteins: Struct.,*
1503 *Funct., Genet.* **2013**, *81*, 1556–1570.
- 1504 (78) Simonson, T. What Is the Dielectric Constant of a Protein
1505 When Its Backbone Is Fixed? *J. Chem. Theory Comput.* **2013**, *9*, 4603–
1506 4608.
- 1507 (79) Archontis, G.; Simonson, T. A residue-pairwise Generalized
1508 Born scheme suitable for protein design calculations. *J. Phys. Chem. B*
1509 **2005**, *109*, 22667–22673.
- 1510 (80) Aguilar, B.; Shadrach, R.; Onufriev, A. V. Reducing the
1511 secondary structure bias in the generalized Born model via R6 effective
1512 radii. *J. Chem. Theory Comput.* **2011**, *6*, 3613–3630.
- 1513 (81) Ollikainen, N.; de Jong, R. M.; Kortemme, T. Coupling Protein
1514 Side-Chain and Backbone Flexibility Improves the Re-design of
1515 Protein-Ligand Specificity. *PLoS Comput. Biol.* **2015**, *1*, e1004335.

Figure 5.23 – 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.

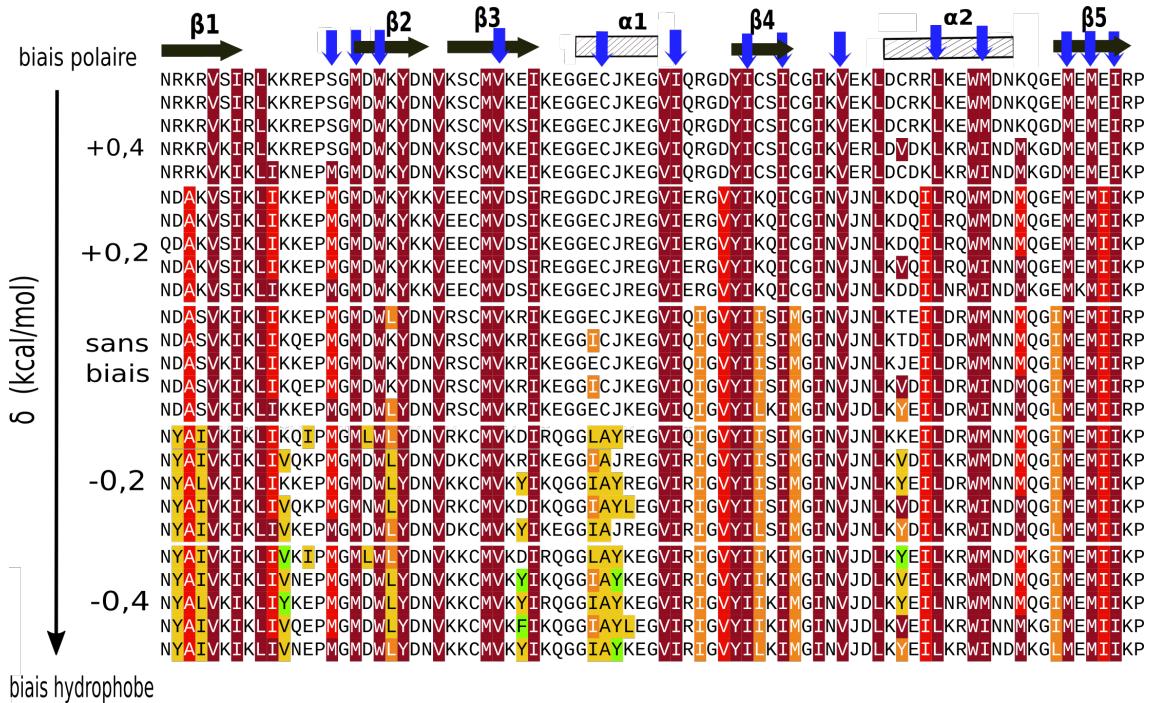


Figure 5.24 – Séquences Cask obtenues avec un delta des énergies de références à -0,4,-0,2,0,0,2 et 0,4 et la structure native. Les hydrophobes pour des deltas de -0,4,-0,2,0,0,2 et 0,4 sont représentés par un jeu de couleurs allant du rouge foncé au vert clair, en passant par le jaune.

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.

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 [57] (comparable au programme très populaire Scwrl4 [56]).

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 [89, 29], 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) [29]. 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 [115]. 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

Chapitre 5. Application aux domaines PDZ

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 [88, 29]. 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 [116, 79]. 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 9% plus haut chez Rosetta que pour la version FDB de notre modèle. Ce qui veut dire que Proteus modifie environ cinq positions en plus, en moyenne, par domaine PDZ. Pour finir, nous signalons que certaines séquences Proteus de Nicolas Panel se replient expérimentalement, voir [?].

Bibliographie

- [1] Ponder J. and Richards F. M. (1987). «Tertiary templates for proteins : Use of packing criteria in the enumeration of allowed sequences for different structural classes». *J. Mol. Biol.*, volume 193:page 775–792.
cité pages 3, 4et 64
- [2] Su A. and Mayo S.L. (1997). «Coupling backbone flexibility and amino acid sequence selection in protein design». *Protein Science*, volume 9:page 1701–1707.
cité page 3
- [3] Finkelstein A. and Ptitsyn O. (1977). «Theory of protein molecule self-organization. i. thermodynamic parameters of local secondary structures in the unfolded protein chain». *Biopolymers*, volume 16:page 469–495.
cité page 4
- [4] Janin J., Wodak S., Levitt M. and Maigret B. (1978). «Conformation of amino acid sidechains in proteins». *Journal of Molecular Biology*, volume 125:page 357–386.
cité page 4
- [5] McGregor M., Islam S. and Sternberg M. (1987). «Analysis of the relationship between sidechain conformation and secondary structure in globular proteins». *J. Mol. Biol.*, volume 198:page 295–310.
cité page 4
- [6] Dunbrack R. and Karplus M. (1993). «Backbone-dependent rotamer library for proteins. application to side-chain prediction». *J. Mol. Biol.*, volume 230:page 543–574.
cité page 4
- [7] Dahiyat B. and Mayo S. (1997). «De novo protein design : fully automated sequence selection». *Science*, volume 278:page 82–87.
cité page 4
- [8] Desjarlais J. and Handel T. (1999). «Side-chain and backbone flexibility in protein core design». *J. Mol. Biol.*, volume 290:page 305–318.
cité page 4

Bibliographie

- [9] Druart K., Bigot J., Audit E. and Simonson T. (2016). «A hybrid monte carlo scheme for multibackbone protein design». *Journal of Chemical Theory and Computation*, volume 12:page 6035–6048.
cité pages 4 et 38
- [10] Dantas G., Corrent C., Reichow S., Havranek J., Eletr Z., Isern N., Kuhlman B., Varani G., Merritt E. and Baker D. (2007). «High-resolution structural and thermodynamic analysis of extreme stabilization of human procarboxypeptidase by computational protein design». *Journal of Molecular Biology*, volume 366:page 1209–1221.
cité page 4
- [11] Kuhlman B., Ireton G., Varani G., Stoddard B. and Baker D. (2003). «Design of a novel globular protein fold with atomic-level accuracy». *Science*, volume 302:page 1364–1368.
cité pages 4 et 31
- [12] Davis I., Arendall W., Richardson D. and Richardson J. (2006). «The backrub motion: how protein backbone shrugs when a sidechain dances». *Structure*, volume 14:page 265–274.
cité page 4
- [13] Georgiev, I., Lilien, R. H., Donald and B. R. (2008). «The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles». *J. Comput. Chem.* 2, volume 9:page 1527–1542.
cité pages 4 et 64
- [14] Smith C. and Kortemme T. (2008). «Backrub-like backbone simulation recapitulates natural protein conformational variability and improves mutant side-chain prediction». *Journal of Molecular Biology*, volume 380:page 742–756.
cité page 4
- [15] Dahiyat B. and Mayo S. (1996). «Protein design automation». *Protein Science*, volume 5:page 895–903.
cité pages 5 et 31
- [16] Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M. Jr., Ferguson, D. M. Spellmeyer, D. C., Fox, T., Caldwell, J. W., , Kollman and P. A. (1995). «A second generation force field for the simulation of proteins, nucleic acids and organic molecules». *J. Am. Chem. Soc.*, volume 117:pages 5179–5197.
cité pages 5 et 104
- [17] Brooks, B., Brooks III, C. L., Mackerell Jr., A. D., Nilsson, L., Petrella, R. J., Roux, B., Won, Y., Archontis, G., Bartels, C., Boresch, S., Caflisch, A., Caves, L., Cui, Q., Dinner, A. R., Feig, M., Fischer, S., Gao, J., Hodoscek, M., Im, W., Kuczera, K., Lazaridis, T., Ma, J., Ovchinnikov, V., Paci, E., Pastor, R. W., Post, C. B.,

- Pu, J. Z., Schaefer, M., Tidor, B., Venable, R. M., Woodcock, H. L., Wu, X., Yang, W., York, D. M., Karplus and M. (2009). «Charmm: The biomolecular simulation program». *J. Comp. Chem.*, volume 30:page 1545–1614.
cité page 5
- [18] Jorgensen W. and Rives J. T. (1988). «The opls force field for proteins. energy minimizations for crystals of cyclic peptides and crambin». *J. Am. Chem. Soc.*, volume 110:page 1657–1666.
cité page 6
- [19] Christen M., Hünenberger P., Bakowies D., Baron R. and Heinz T. Kastenholz M. Kräutler V. Oostenbrink C. Peter C. Trzesniak D. et van Gunsteren W. Bürgi R., Geerke D. (2005). «The gromos software for biomolecular simulation : Gromos05». *J. Comp. Chem.*, volume 16:page 1719–1751.
cité page 6
- [20] Zollars ES., Marshall SA., and Mayo SL (2006). «Simple electrostatic model improves designed protein sequences». *Protein Science*, volume 15:page 2014–2018.
cité page 8
- [21] Pokala N. and Handel T. (2005). «Energy functions for protein design : Adjustement with protein-protein complex affinities, models for the unfolded state, and negative design of solubility and specificity. j. mol». *Biol.*, volume 347:page 203–227.
cité page 8
- [22] Korkut A. and Hendrickson W. (2009). «A force field for virtual atom molecular mechanics of proteins». *Proc. Natl. Acad. Sci. USA*, volume 106:page 15667–72.
cité page 8
- [23] Wesson L. and Eisenberg D. (1992). «Atomic solvation parameters applied to molecular dynamics of proteins in solution». *Protein Science*, volume 1:page 227–235.
cité page 11
- [24] Rocchia W, Sridharan S, Nicholls A, Alexov E, Chiabrera A and Honig B. (2002). «Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: applications to the molecular systems and geometric objects». *J Comput Chem.*, volume 23(1):pages 128–37.
cité page 13
- [25] Baker NA, Sept D, Joseph S, Holst MJ and McCammon JA. (2001). «Electrostatics of nanosystems: application to microtubules and the ribosome». *Proc Natl Acad Sci USA*, volume 98(10):pages 10037–41.
cité page 13
- [26] Born M. (1920). «Volumen und hydratationswarme der ionen. z». *Phys.*, volume 1:page 45–48.
cité page 13

Bibliographie

- [27] Still W., Tempczyk A., Hawley R. and Hendrickson T. (**1990**). «Semianalytical treatment of solvataion for molecular mechanics and dynamics». *J Am Chem Soc*, volume 112:page 6127–29.
cité page 13
- [28] Polydorides S., Amara N., Aubard C., Plateau P., Simonson T. and Archontis G. (**2011**). «Computational protein design with a generalized born solvent model: application to asparaginyl-trna synthetase». *Proteins*, volume 79:page 3448–3468.
cité pages 15et 88
- [29] 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.
cité pages 31, 164, 165et 166
- [30] Gaillard T and Simonson T (**2014**). «Pairwise decomposition of an mmgbsa energy function for computational protein design». *J Comput Chem*, volume 35:page 1371–1387.
cité pages 15, 34, 38et 81
- [31] Isard C. Perry B. (**2012**). «Theory and practice in replica-exchange molecular dynamics simulation». *ProQuest, UMI Dissertation Publishing*.
cité page 16
- [32] Desmet J., Mayer M. D., Hazes B. and Lasters I. (**1992**). «The dead-end elimination theorem and its use in protein side-chain positioning». *Nature*, volume 356:page 539–542.
cité page 17
- [33] Goldstein R. (**1994**). «Ecient rotamer elimination applied to protein side-chains and related spin glasses». *Biophysical Journal*, volume 66:page 1335–1340.
cité page 18
- [34] Leach A. and Lemon A. (**1998**). «Exploring the conformational space of protein side chains using dead-end elimination and the α^* algorithm». *Proteins: Structure, Function, and Genetics*, volume 33:page 227–239.
cité page 18
- [35] Pierce N.A., Spriet J.A., Desmet J. and Mayo S.L. (**2000**). «Conformational splitting: a more powerful criterion for dead-end elimination». *J. Comp. Chem.*, volume 21:pages 999–1009.
cité page 18
- [36] Lilien R., Stevens B., Anderson A. and Donald B. (**2005**). «A novel ensemble-based scoring and search algorithm for protein redesign and its application to modify the substrate specificity of the gramicidin synthetase a phenylalanine adenylation enzyme». *Journal of Computational Biology*, volume 12:page 740–761.
cité page 18

- [37] Schiex T. (2000). «Arc consistency for soft constraints». *Principles and Practice of Constraint Programming*, volume 1894:page 411–424.
cité page 21
- [38] Allouche D., André I., Barbe S., Davies J., Givry S., Katsirelos G., O’Sullivan B., Prestwich S., Schiex T. and Traoré S. (2014). «Computational protein design as an optimization problem». *Artificial Intelligence*, volume 212:page 59–79.
cité pages 21, 64, 65et 70
- [39] Traoré, S., Allouche, D., André, I., De Givry, S., Katsirelos, G, Schiex, T, Barbe and S. (2013). «A new framework for computational protein design through cost function network optimization». *Bioinformatics*, volume 27 (19):pages 2129–2136.
cité pages 21, 64, 65et 70
- [40] Wernisch L., Hery S. and Wodak S. (2000). «Automatic protein design with all atom forceelds by exact and heuristic optimization». *Journal of Molecular Biology*, volume 301:page 713–736.
cité pages 21, 64, 65et 70
- [41] Goldberg D. E. and Holland J. H. (1988). «Genetic algorithms and machine learning». *Machine learning*, volume 3(2):pages 95–99.
cité page 23
- [42] Cercignani, C., Lampis and M. (1981). «On the h-theorem for polyatomic gases». *J Stat Phys*, volume 26:page 4.
cité page 25
- [43]
cité page 28
- [44] Gainza P., Roberts K., Georgiev I., Lilien R., Keedy D., Chen C., Reza F., Anderson A., Richardson D., Richardson J. and Donald B. (2013). «Osprey: Protein design with ensembles, exibility and provable algorithms». *Methods in Enzymology*, volume 523:page 87–107.
cité page 31
- [45] 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.
cité page 31
- [46] A. T. Brünger (1992). *X-PLOR version 3.1, A System for X-ray crystallography and NMR*. Yale University Press, New Haven.
cité pages 33, 37, 55, 71et 104
- [47] Simonson T., Ye-Lehmann S., Palmai Z., Amara N., Wydau-Dematteis S., Bigan E., Druart K., Moch C. and Plateau P. (2016). «Redesigning the stereospecificity of tyrosyltrna synthetase». *Proteins*, volume 84:page 240–253.
cité page 33

Bibliographie

- [48] 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 ?:page ??–??
cité pages 33, 35et 104
- [49] 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.
cité pages 33et 131
- [50] Street A.G. and Mayo S.L (1998). «Pairwise calculation of protein solvent-accessible surface areas». *Folding and Design*, volume 3:page 253–258.
cité pages 34et 71
- [51] Lopes A., Aleksandrov A., Bathelt C., Archontis G. and Simonson T. (2007). «Computational sidechain placement and protein mutagenesis with implicit solvent models». *Proteins*, volume 67:page 853–867.
cité pages 34, 71et 81
- [52] Hawkins G., Cramer C.J. and Truhlar D. (1995). «Pairwise solute screening of solute charges from a dielectric medium». *Chem. Phys. Lett.*, volume 246:page 122–129.
cité page 35
- [53] Schaefer M. and Karplus M (1996). «A comprehensive analytical treatment of continuum electrostatics». *J. Phys. Chem*, volume 100:page 1578–1599.
cité page 35
- [54] Archontis G and Simonson T (2005). «A residue-pairwise generalized born scheme suitable for protein design calculations». *J Phys Chem B*, volume 109:page 22667–22673.
cité page 36
- [55] Tuffery P., Etchebest C., Hazout S. and Lavery R (1991). «A new approach to the rapid determination of protein side chain conformations». *Journal of Biomolecular Structure Dynamics*, volume 8:page 1267–1289.
cité pages 38et 72
- [56] Krivov GG, Shapalov MV and Dunbrack RL (2009). «Improved prediction of protein side-chain conformations with scwrl4». *Proteins*, volume 77:page 778–795.
cité pages 38, 72et 164
- [57] Gaillard T, Panel N and Simonson T (2016). «Protein sidechain conformation predictions with an mmgbsa energy function». *Proteins*, volume 84:page 803–819.
cité pages 38, 72, 104et 164
- [58] Madera M, Vogel C, Kummerfeld SK, Chothia C and Gough J (2004). «The superfamily database in 2004: additions and improvements». *Nucl Acids Res*, volume 32:page D235–D239.
cité page 47

- [59] Andreeva, A., Howorth, D., Brenner, S. E., Hubbard, J. J., Chothia, C., , Murzin and A. G. (2004). «Scop database in 2004: refinements integrate structure and sequence family data». *Nucl. Acids Res.* 3, volume 2:page D226–229.
cité pages 47, 73et 126
- [60] Hughey R. and Krogh A. (1995). «Sam: Sequence alignment and modeling software system». cité page 48
- [61] <http://hmmer.org>.
cité page 48
- [62] Punta M., Coggill P., Eberhardt R., Mistry J., Tate J., Boursnell C., Pang N., Forslund K., Ceric G., Clements J., Heger A., Holm L., Sonnhammer E., Eddy S., Bateman A. and Finn R. (2012). «The pfam protein families database». *Nucleic Acids Research*, volume 40:page 290–301.
cité page 48
- [63] Finn, R.D., Bateman, A., Clements J., Coggill P., Eberhardt R.Y., Eddy S.R., Heger A., Hetherington K., Holm L., Mistry J., Sonnhammer E.L.L., Tate J. and Punta M. (2014). «The pfam protein families database». *Nucleic Acids Research*, volume Issue 42:pages D222–D230.
cité page 48
- [64] Henikoff S. and Henikoff J. (1992). «Amino acid substitution matrices from protein blocks.» *PNAS*, volume 89:pages 10915–10919.
cité page 49
- [65] Altschul S., Madden T., Schaffer A., Zhang J., Zhang Z., Miller W. and Lipman D. (1997). «Gapped balst and psi-blast : a new generation of protein database search programs». *Nucleic Acids Res*, volume 25:page 3389–3402.
cité page 49
- [66] Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K and Madden T.L. (2008). «Blast+: architecture and applications.» *BMC Bioinformatics*, volume 10:page 421.
cité page 49
- [67] Durbin R., Eddy S. R., Krogh A. and Mitchison G. (2002). *Biological sequence analysis*. Cambridge University Press, Cambridge, United Kingdom.
cité pages 49et 73
- [68] Launay G., Mendez R., Wodak S. J. and Simonson T. (2007). «Recognizing proteinprotein interfaces with empirical potentials and reduced amino acid alphabets». *BMC Bioinf.*, volume 8:page 270–291.
cité pages 50et 73

Bibliographie

- [69] Dantas, G., Kuhlman, B., Callender, D., Wong, M., Baker and D. (2003). «A large test of computational protein design: Folding and stability of nine completely redesigned globular proteins». *J. Mol. Biol.* 33, volume 2:page 449–460.
cité pages 64et 130
- [70] Lippow SM and Tidor B (2007). «Progress in computational protein design». *Curr Opin Biotech*, volume 18:page 305–311.
cité pages
- [71] Saven and J. G. (2011). «Computational protein design: engineering molecular diversity, nonnatural enzymes, nonbiological cofactor complexes, and membrane proteins». *Curr. Opin. Chem. Biol.*, volume 15:page 452–457.
cité pages
- [72] Feldmeier K and Hoecker B (2013). «Computational protein design of ligand binding and catalysis». *Curr Opin Chem Biol*, volume 17:page 929–933.
cité pages
- [73] Tinberg CE, Khare SD, Dou J, Doyle L, Nelson JW, Schena A and et al (2013). «Computational design of ligand-binding proteins with high affinity and selectivity». *Nature*, volume 501:page 212–218.
cité page 64
- [74] Pokala N and Handel TM (2004). «Energy functions for protein design i: Efficient and accurate continuum electrostatics and solvation». *Prot Sci*, volume 13:page 925–936.
cité page 64
- [75] Samish I, MacDermaid CM, Perez-Aguilar JM and Saven JG (2011). «Theoretical and computational protein design». *Ann Rev Phys Chem*, volume 62:page 129—149.
cité page 64
- [76] Li Z., Yang Y., Zhan J., Dai L. and Zhou Y. (2013). «Energy functions in de novo protein design: Current challenges and future prospects». *Ann Rev Biochem*, volume 42:page 315–335.
cité page 64
- [77] Koehl P. and Levitt M. (2002). «Protein topology and stability define the space of allowed sequences». *Proc. Natl. Acad. Sci. USA* 9, volume 9:page 1280–1285.
cité page 64
- [78] Larson S. M., England J. E., Desjarlais J. R. and Pande V. S. (2002). «Thoroughly sampling sequence space: Large-scale protein design of structural ensembles». *Prot. Sci.* 1, volume 1:page 2804–2813.
cité pages
- [79] Schmidt am Busch M, Mignon D and Simonson T (2009). «Computational protein design as a tool for fold recognition». *Proteins*, volume 77:page 139–158.
cité pages 64, 65, 72, 74, 85, 88, 125et 166

- [80] Polydorides S and Simonson T. (2013). «Monte carlo simulations of proteins at constant ph with generalized born solvent, flexible sidechains, and an effective dielectric boundary». *J Comput Chem*, volume 34:page 2742–2756.
cité page 64
- [81] Kilambi K. and Gray J.J. (2012). «Rapid calculation of protein pka values using rosetta». *Biophys J*, volume 103:page 587–595.
cité page 64
- [82] Looger L. and Hellinga H. (2001). «Generalized dead-end elimination algorithms make largescale protein side-chain structure prediction tractable: Implications for protein design and structural genomics». *Journal of Molecular Biology*, volume 307:page 429–445.
cité page 64
- [83] Zou J. and Saven J. G. (2003). «Using self-consistent fields to bias monte carlo methods with applications to designing and sampling protein sequences». *J. Chem. Phys.* 11, volume 8:page 3843–3854.
cité page 64
- [84] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A. and Teller E. (1953). «Equation of state calculations by fast computing machines». *The Journal of Chemical Physics*, volume 21:page 1087–1092.
cité pages 65, 67et 68
- [85] Sugita, Y., Okamoto and Y. (1999). «Replica-exchange molecular dynamics method for protein folding». *Chem. Phys. Lett.* 31, volume 4:page 141–151.
cité page 65
- [86] Kofke DA (2002). «On the acceptance probability of replica-exchange monte carlo trials». *J Chem Phys*, volume 117:page 6911–6914.
cité page 69
- [87] Earl D. and Deem M. W. (2005). «Parallel tempering: theory, applications, and new perspectives». *Phys. Chem. Chem. Phys.*, volume 7:page 3910–3916.
cité pages 65et 69
- [88] Schmidt Am Busch M., Lopes A., Mignon D. and Simonson T. (2008). «Computational protein design: software implementation, parameter optimization, and performance of a simple model». *Journal of Computational Chemistry*, volume 29:page 1092–1102.
cité pages 65, 71, 81, 125et 166
- [89] Schmidt am Busch M., Lopes A., Amara N., Bathelt C. and Simonson T. (2008). «Testing the coulomb/accessible surface area solvent model for protein stability, ligand binding, and protein design». *BMC Bioinformatics*, volume 9:page 148–163.
cité pages 66, 70, 71, 103et 164

Bibliographie

- [90] Simonson T (2013). «Protein:ligand recognition: simple models for electrostatic effects». *Curr Pharma Design*, volume 19:page 4241–4256.
cité pages 65, 66, 71, 103et 125
- [91] Dahiyat B., Gordon D. and Mayo S. (1997). «Automated design of the surface positions of protein helices». *Protein Science*, volume 6:page 1333–1337.
cité page 65
- [92]
cité pages 66, 73et 126
- [93] Wilson D, Madera M, Vogel C, Chothia C and Gough J (2007). «The superfamily database in 2007: families and functions». *Nucl Acids Res*, volume 35:page D308—D313.
cité pages 66, 73et 126
- [94] Lee B. and Richards F. (1971). «The interpretation of protein structures : estimation of static accessibility. j. mol». *Biol.*, volume 55:page 379–400.
cité pages 71et 104
- [95] Mandell D., Coutsias E. and Kortemme T. (2009). «Sub-angstrom accuracy in protein loop reconstruction by robotics-inspired conformational sampling». *Nature Methods*, volume 6:page 551–552.
cité pages
- [96] Harris BZ and Lim WA (2001). «Mechanism and role of pdz domains in signaling complex assembly». *J Cell Sci*, volume 114:page 3219–3231.
cité page 99
- [97] Hung AY and Sheng M (2002). «Pdz domains: Structural modules for protein complex assembly». *J Biol Chem*, volume 277:page 5699–5702.
cité pages
- [98] Tonikian R, Zhang YN, Sazinsky SL, Currell B, Yeh JH, Reva B and et al (2008). «A specificity map for the pdz domain family». *PLoS Biology*, volume 6:page 2043–2059.
cité pages
- [99] Gfeller D, Butty F, Wierzbicka M, Verschueren E, Vanhee P, Huang H and et al (2011). «The multiplespecificity landscape of modular peptide recognition domains». *Molec Syst Biol*, volume 7:page 484.
cité pages
- [100] Subbaiah VK, Kranjec C, Thomas M and Ban L (2011). «Structural and thermodynamic analysis of pdz-ligand interactions». *Biochem J*, volume 439:page 195–205.
cité page 99

- [101] Roberts KE, Cushing PR, Boisguerin P, Madden DR and Donald BR (**2012**). «Computational design of a pdz domain peptide inhibitor that rescues cftr activity». *PLoS Comp Bio*, volume 8:page e1002477.
cité page 99
- [102] Zheng F, Jewell H, Fitzpatrick J, Zhang J, Mierke DF and Grigoryan G (**2015**). «Computational design of selective peptides to discriminate between similar pdz domains in an oncogenic pathway». *J Mol Biol*, volume 427:page 491–510.
cité page 99
- [103] Kong Y and Karplus M (**2009**). «Signaling pathways of pdz2 domain: A molecular dynamics interaction correlation analysis». *Proteins*, volume 74:page 145–154.
cité page 99
- [104] McLaughlin Jr RN, Poelwijk FJ, Raman A, Gosal WS and Ranganathan R (**2012**). «The spatial architecture of protein function and adaptation». *Nature*, volume 458:page 859–864.
cité pages
- [105] Melero C, Ollikainen N, Harwood I, Karpiak J and Kortemme T (**2014**). «Quantification of the transferability of a designed protein specificity switch reveals extensive epistasis in molecular recognition». *Proc Natl Acad Sci USA*, volume 111:page 15426–15431.
cité page 99
- [106] Reina J, Lacroix E, Hobson SD, Fernandez-Ballester G, Rybin V, Schwab MS and et al (**2002**). «Computer-aided design of a pdz domain to recognize new target sequences». *Nat Struct Mol Biol*, volume 9:page 621–627.
cité page 99
- [107] Smith CA and Kortemme T (**2010**). «Structure-based prediction of the peptide sequence space recognized by natural and synthetic pdz domains». *J Mol Biol*, volume 402:page 460–474.
cité page 99
- [108] Baker D (**2006**). «Prediction and design of macromolecular structures and interactions». *Phil Trans R Soc Lond*, volume 361:page 459–463.
cité pages 99et 125
- [109] Kleinman CL, Rodrigue N, Bonnard C, Philippe H and Lartillot N (**2006**). «A maximum likelihood framework for protein design». *BMC Bioinf*, volume 7:page Art. 326.
cité pages 101et 102
- [110] 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.
cité pages 63et 103

Bibliographie

- [111] Druart K, Palmai Z, Omarjee E and Simonson T (**2016**). «Protein:ligand binding free energies: a stringent test for computational protein design». *J Comput Chem*, volume 37:page 404–415.
cité page 103
- [112] Aleksandrov A, Polydorides S, Archontis G and Simonson T (**2010**). «Predicting the acid/base behavior of proteins: A constant-ph monte carlo approach with generalized born solvent». *J Phys Chem B*, volume 114:page 10634–10648.
cité page 104
- [113] (**2006**). «Comparative protein structure modeling with MODELLER». *Curr. Prot. Bioinf.*, volume Suppl. 15:pages 5.6.1–5.6.30.
cité page 125
- [114] Jaramillo A, Wernisch L, Hery S and Wodak S (**2002**). «Folding free energy function selects nativelike protein sequences in the core but not on the surface». *Proc Natl Acad Sci USA*, volume 99:page 13554–13559.
cité page 130
- [115] Druart K., Le Guennec M., Palmai Z. and Simonson T. (**2017**). «Probing the stereospecificity of tyrosyl- and glutaminyl-tRNA synthetase with molecular dynamics simulations». *Journal of Molecular Graphics and Modelling*, volume 71:page 192–199.
cité page 165
- [116] Schmidt am Busch M., Sedano A. and T. Simonson (**2010**). «Computational protein design: validation and possible relevance as a tool for homology searching and fold recognition». *PLoS One*, volume 5(5):page e10410.
cité page 166

Résumé

Titre de la thèse

Le CPD ou « Computational protein design » est la recherche par modélisation moléculaire des séquences d'acides aminés compatibles avec une structure protéique ciblée. L'objectif est de concevoir une fonction nouvelle et/ou d'ajouter un nouveau comportement. Le CPD est en développement au sein de notre laboratoire depuis quelques années, avec le logiciel Proteus qui a plusieurs succès à son actif. Au cours de cette Thèse, Nous avons enrichi Proteus sur plusieurs points, avec notamment l'ajout d'une nouvelle méthode d'exploration de l'espace d'état de type Monte Carlo avec échange de réplique. Une série de comparaisons entre les algorithmes de Proteus ont été effectuées. Ces comparaisons portent sur trois familles de protéines (SH2, SH3 et PDZ) avec dans chaque famille deux ou trois représentants dont la taille varie entre 57 et 109 acides aminés. Le jeu de données constitué a également été utilisé pour optimiser certain paramètre de MCER : distribution des températures, nombre de répliques, fréquences des échanges, etc. Les résultats montrent que le MCER avec un protocole de huit marcheurs et des températures comprises entre 3 et 0,2 est systématiquement meilleur en terme d'énergie ou équivalent que tous les autres protocoles testés (le MCER huit marcheurs avec des températures plus écartées, le MCER avec quatre marcheurs, le Monte-Carlo, l'optimisation itérative sur chaque position).

Toujours pour évaluer proteus, nous avons utilisé le programme toulbar2 d'une équipe de l'Université de Toulouse (UMR792) qui propose un algorithme de recherche de type Dead End Elimination (DEE). Cet algorithme trouve le minimum global en éliminant successivement les configurations ne pouvant pas appartenir à ce minimum. Cette méthode fonctionne bien si l'espace de recherche n'est pas trop grand. Nous avons alors fixé plusieurs acides aminés de nos protéines afin de diminuer la taille de l'espace de recherche. Si on laisse libre entre dix et vingt acides aminés selon les protéines alors toulbar2 donne un résultat en moins de 24 heures de calculs sans demander plus de 8 Go de mémoire vive. Des comparaisons sont en cours pour des espaces allant de zéro à vingt acides aminés libres. Les premiers résultats montrent que proteus trouve systématiquement le minimum global.

Mots-clés : modélisation moléculaire, conception de protéine par ordinateur, Proteus, Monte Carlo, domaine PDZ

Abstract

Thesis title

XXX

Keywords: molecular modeling, computational protein design, Proteus, Monte Carlo PDZ domain