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) [18], OSPREY (Open Source Protein REdesign for You) [47], Rosetta (le CPD n'est qu'une partie des fonctionnalités proposées par cette suite logicielle) [14] et Proteus [32, 48]. 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 1.30 page 30.

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 avec l'entropie et la température. Il s'agit de l'énergie libre de Gibbs G. De plus, on prend la différence entre états replié et déplié.

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

$$\Delta\Delta G = (G(S_B^P) - G(S_A^P)) - (G(S_B^D) - G(S_A^D)) \tag{2.1}$$

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

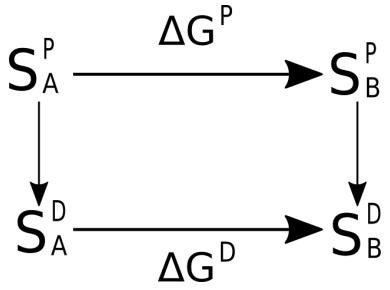

Figure 2.1 – Cycle thermodynamique qui définit la stabilité

2.2 Organisation générale

Proteus est constitué de deux parties :

- un ensemble de scripts X-PLOR, qui pilote le calcul de la matrice d'énergie. X-PLOR est un programme de simulation moléculaire [97], disponible en téléchargement sur le site de l'université de Yale.
- un programme écrit en C, que nous nommons « proteus » dans la suite de ce texte, qui explore l'espace des séquences-conformations

De plus, nous développons une version modifiée de X-PLOR, qui fournit plusieurs fonctionnalités spécifiques au CPD. 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 était obtenue avec Proteus par exemple de redesign de la tyrosyl-tARN synthétase [49], sur les calculs de pK_a [51] ou la création de nouveaux domaines PDZ [52].

Notre logiciel Proteus autorise le choix entre plusieurs types de fonctions d'énergie. Le type le plus simple « MMCASA » combine l'approche de la mécanique moléculaire pour

le traitement de la protéine et un modèle de solvant implicite de type CASA (voir 1.3.3 page 15). 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

Avoir une fonction d'énergie décomposable par paires a plusieurs avantages. En particulier, le calcul des énergies des paires permet de réduire le calcul de l'énergie d'une séquence-conformation pendant l'étape d'exploration 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 14, se définit comme :

$$E_{solv}^{surf} = \sum_i \sigma_{t_i} A_i \tag{2.2}$$

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.

Pour rendre ce terme décomposable, nous utilisons la méthode de Street et al. [53] dans laquelle la surface enfuie d'une chaîne est calculée à partir d'une somme sur les chaînes et les groupes du backbone 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 [1, 33].

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. Ce rayon n'est donc pas décomposable par paire. Pour qu'il le

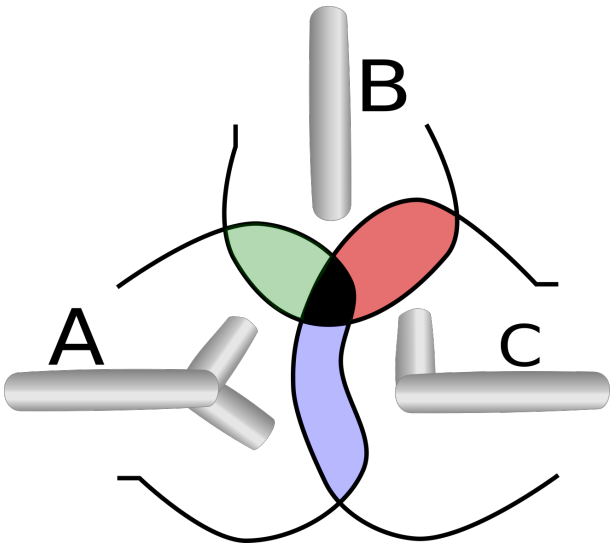

Figure 2.2 – **Une représentation de la surface accessible de trois résidus.** Les résidus A et B réduisent mutuellement leur surface exposée au solvant : c'est la zone verte. De même pour B, C : la zone rouge et A, C, la zone bleue. Un calcul par paires de résidus naïf surestime la surface accessible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

devienne, Proteus implémente l'approximation NEA qui est présentée à la section 1.3.5 page 18. Dans cette approche, le rayon de solvatation b_i de chaque groupe (backbone, chaîne latérale ou ligand) est calculé une fois pour toutes à partir de la structure native.

2.3.3 « Fluctuating Dielectric Boundary » (FDB)

Une nouvelle approximation du GB a récemment été introduite dans Proteus [51], toujours avec l'objectif, de transformer ce terme en en 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 [54], [55]. 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 commence par définir une énergie propre à chaque paire de résidus I, J par la somme suivante :

$$E_{IJ}^{self} \stackrel{def}{=} \sum_{i \in I, j \in J} E_{ij}^{self} \tag{2.3}$$

puis l'énergie propre d'un résidu I :

$$E_{I}^{self} \stackrel{def}{=} \sum_{J} E_{IJ}^{self} \tag{2.4}$$

alors le rayon de solvatation moyen B_I est défini par :

$$E_I^{self} \stackrel{def}{=} \tau \sum_{i \in I} \frac{q_i^2}{2B_I} \tag{2.5}$$

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

$$\left(\sum_{i \in I} q_i^2 \right) \frac{1}{B_I} = \sum_{i \in I} \frac{q_i^2}{b_i} \tag{2.6}$$

Donc, B_I est la moyenne harmonique des b_i, avec $i \in I$, pondéré 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 1.20 page 17, par :

$$g_{IJ} = \sum_{i \in I, j \in J} \tau q_i q_j \left(r_{ij}^2 + B_I B_J \exp[-r_{ij}^2 / 4 B_I B_J] \right)^{-1/2} \tag{2.7}$$

avec la somme pour $I = J$ excluant le cas $i \neq j$. On peut noter qu'aux distances fixes r_{ij}, et avec $B = B_I BJ$, $g_{IJ}(B)$ varie faiblement en fonction de B. Archontis et Simonson [56], proposent alors d'approximer cette fonction par :

$$g_{IJ}(B) \approx c_1^{IJ} + c_2^{IJ} B + c_3^{IJ} B^2 + c_4^{IJ} B^{-1/2} + c_5^{IJ} B^{-3/2} \tag{2.8}$$

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 donné. Comme les rayons de solvatation d'un résidu sont eux-mêmes des sommes sur les paires, le calcul de l'énergie GB à l'aide de l'approximation 2.8 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, ceci 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.

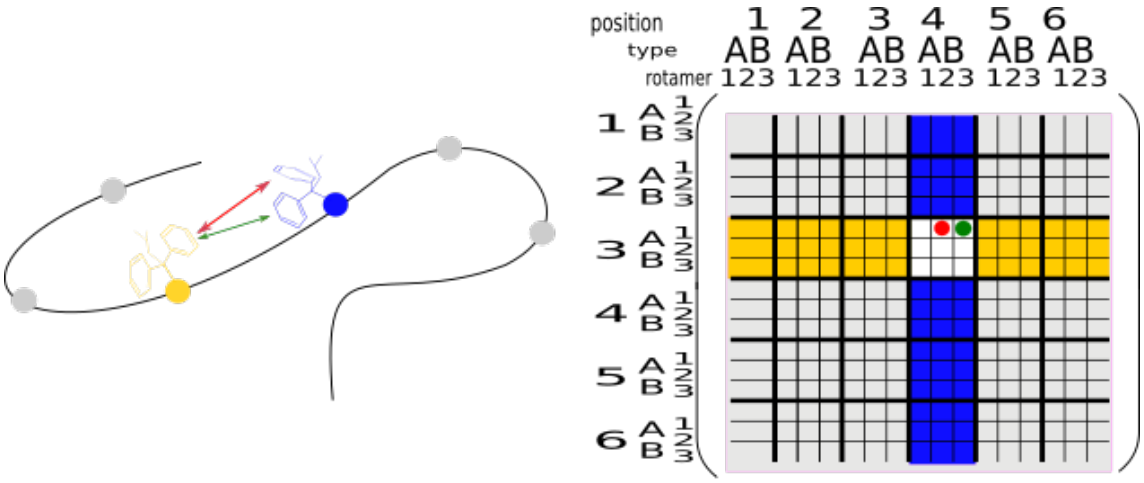

Figure 2.3 – **La matrice d'énergie.** Cet exemple montre un polypeptide de 6 résidus, chaque position possède 2 types d'acides aminés possibles et 3 rotamères possibles (2 pour le type A et 1 pour le type B). La matrice organise toutes les interactions de paires de chaînes latérales possibles. Les interactions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

2.4.1 Les énergies de l'état déplié

$\Delta\Delta G$ est une différence 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 9, telle que :

$$E_u(S) = \sum_{i}^{N} E(t_i) \tag{2.9}$$

avec $E(t_i)$ l'énergie du type d'acide aminé t_i. Ces énergies sont prises en entrée dans Proteus et donc leur détermination ne fait pas encore partir des fonctionnalités intégrées à notre logiciel. Il apparaît clairement dans la suite que la détermination 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 X-plor [97]. À partir d'un fichier PDB, une série de scripts X-plor commence par préparer le système. Il peut contenir un ligand. Le déroulement est le suivant :

1. L'utilisateur configure l'exécution. Il détermine :
   - un champ de force (Les champs AMBER ff99SB ou CHARMM toph19 sont tous les deux supportés.)
   - un traitement de l'électrostatique du solvant parmi SA GB/NEA GB/FDB
   - un jeu de coefficients σ_i pour la pondération de la surface accessible au solvant dans le terme hydrophobe
   - l'ensemble des résidus dans le backbone autorisés à muter
   - l'ensemble des chaînes latérales autorisées à changer de rotamère
2. Les atomes du squelette sont fixés une fois pour toutes.
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 possible, 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, seules les interactions de la paire entre les autres chaînes et le backbone sont prises en compte. Cette courte minimisation réduit l'impact de l'approximation de rotamères discrets. Les rotamères de chaînes latérales utilisés sont extraits d'une version légèrement étendue de la librairie de Tuffery et al. [57] qui possède 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 [33]. 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 en comparaison au programme spécialisé scwlr4 qui utilise une bibliothèque beaucoup plus grande [58, 59]. 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 obtenus permet d'obtenir deux fichiers d'énergies : un fichier d'énergies propres et un fichier d'énergies d'interactions. Le premier rassemble les énergies propres de chaque chaîne latérale possible et l'énergie de son interaction avec la partie fixe de la protéine, c'est-à-dire le backbone et les résidus gelés. Chaque ligne de ce fichier correspond à un rotamère possible de chaque chaîne latérale possible de chaque position non fixée. Les quatre premiers champs identifient la position du résidu, son type, son rotamère. Les autres champs contiennent les termes de l'énergie. Le nombre de ces termes variant selon le traitement du solvant, 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 le couple de types d'acide aminé considérés. Des lignes du second type suivent une ligne du premier type et donnent dans les deux premiers champs le couple de rotamères impliqué dans l'interaction, l'ensemble des termes énergétiques de l'interaction occupent les champs suivants, 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 à l'aide des énergies obtenues. 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

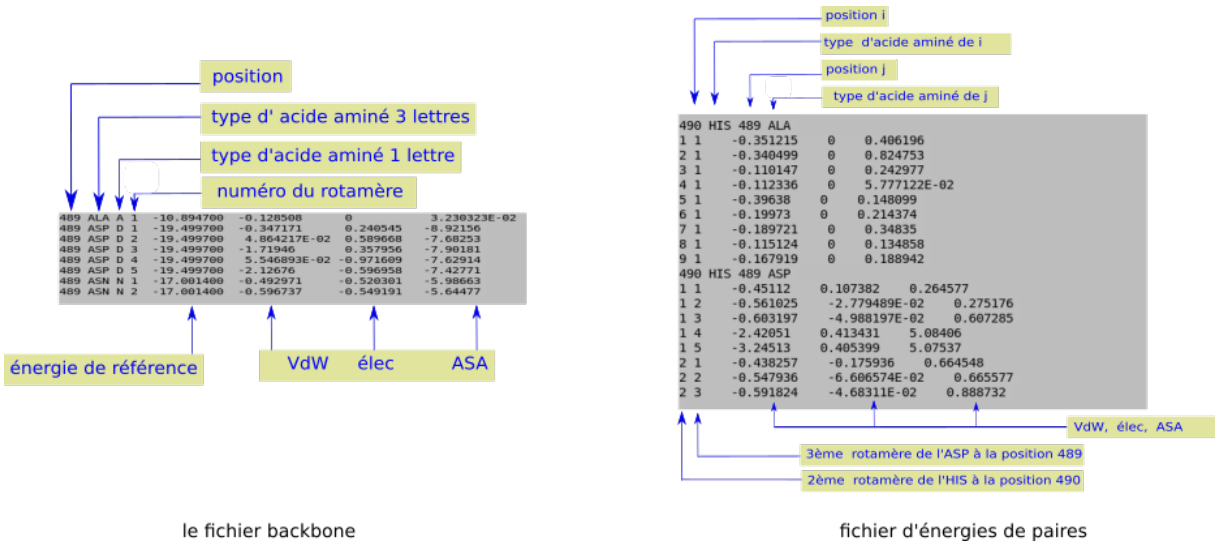

Figure 2.4 – les fichiers d'énergies en mode CASA

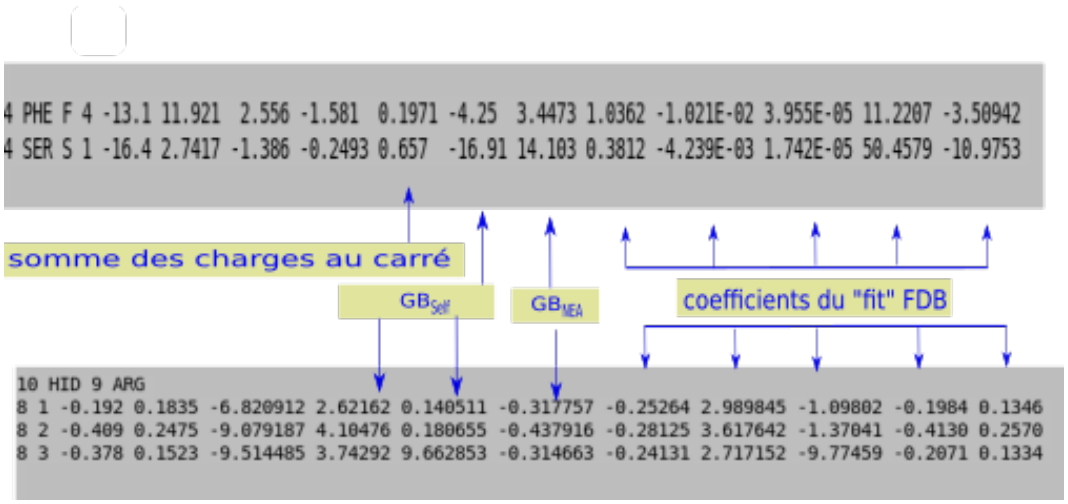

Figure 2.5 – les fichiers d'énergies en mode GB/NEA et FDB

près de quarante balises possibles, une partie est présentée table 2.1 page 45. 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 paramétrer le nombre de cycles et le nombre de passages sur la séquence, voir section 1.4.4 page 26. 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 1.36 page 32. Si elle est non nulle, l'exploration est de type REMC.

Pour contrôler les déplacements des marcheurs, il existe

- un ensemble de cinq balises pour définir les modifications à effectuer dans la séquence-conformation courante
- une balise qui définit un voisinage de la première position modifié dans lequel sera choisi une autre position pour une seconde modification

Il est possible de réduire l'espace des états que peut prendre une séquence-conformation. Les explications sont en 2.5.2. Il existe deux balises pour définir un système de groupes et déclarer une fonction de score. Elles permettent de définir une optimisation non plus uniquement du type de l'équation 2.1, mais répondant à un ensemble de problématiques nettement plus large. Nous expliquons ce système en 2.5.3 page 47.

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 aléatoire sur l'ensemble des positions du système. En revanche, le choix de la seconde position à modifier se fait par tirage aléatoire dans le voisinage de la première position. Le voisinage d'une position se définit de la façon suivante. 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 :

$$|E(r_{Pi},r_{pj})| > s_{vois}$$

avec s_{vois} 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 63.

Type	nom	Description
méthode d'exploration	Mode	détermine le mode d'exécution, les valeurs possibles sont 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(mode HEURISTIC)
fonction d'énergie	Optimization_Configuration	définition de la fonction d'énergie
	Group_definition	groupes d'énergies et d'énergies d'interactions,ce sont les éléments de base de la fonction d'énergie
restrictions de l'espace de séquence-rotamère	Space_Constraints	restreint les états pouvant être visités
paramètre du modèle	Temperature	attribue les températures aux marcheurs MC
Configuration Monte Carlo	Random_Generator	le générateur de nombre aléatoire de la « GNU Scientific Library »
	Rot_Proba Rot_Rot_Proba Mut_Proba Mut_Mut_Proba Mut_Rot_Proba	probabilité d'avoir un changement de rotamère à chaque pas probabilité d'avoir un double changement de rotamère à chaque pas ... ... (ancienne version de Proteus)
	Position_Weights Step_Definition_Proba	probabilité de tirage de chaque position, lors du premier choix probabilité de changer un rotamère ou un type d'aa
	Neighbor_Threshold	à chaque pas les changements se font dans le même voisinage énergétiques.Cette balise définit la taille des voisinages.
Input/Output	Fasta_File	le nom du fichier produit par le mode POSTPROCESS
	Seq_Output_File	le nom du fichier de séquences produit par le mode HEURISTIC ou MONTECARLO
	Energy_Output_File	le nom du fichier d'énergie produit par le mode HEURISTIC ou MONTECARLO

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

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 a la possibilité de restreindre cet espace en utilisant la balise `<Space_Constraints>` 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 deux types ASN et ARG sont seuls possibles et que pour ARG seuls les rotamères 1 , 8 et 12 (selon l'indexation fournie dans le fichier « backbone ») sont autorisés. 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 dépend d'une différence d'énergie libre de protonation. Le problème de la reconnaissance d'un ligand par une protéine peut également être traité. Cela peut se faire par un critère d'affinité via l'énergie libre d'interaction ou par un critère de spécificité. Le critère d'affinité se base sur le cycle thermodynamique 2.6 qui permet d'écrire l'équation :

$$\Delta\Delta G = (G(P : L_2) - G(P : L_1)) - (G(L_2) - G(L_1)) \tag{2.10}$$

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

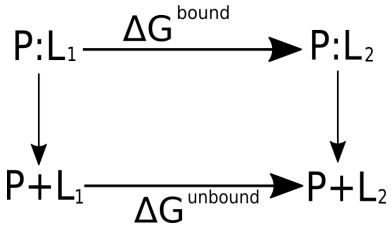

Figure 2.6 – Cycle thermodynamique qui définit l'affinité.

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 2.10 en contributions intramoléculaire et intermoléculaire. De plus, il autorise une pondération des différents termes et permet de dupliquer des parties du système. Cela permet également des

optimisations simultanées de deux conformations d'un sous-système à séquences identiques, mais non fixées. Nous détaillons maintenant la façon dont ce dispositif fonctionne.

Définition de groupes

L'utilisateur a la possibilité de définir des groupes au sein du système biologique étudié. Un groupe est associé à une partie du système. Il se définit en donnant la liste des positions qu'il contient dans la balise `<Group_Definition>` du fichier de configuration. Par exemple :

```
<Group_Definition>
grp1 13
grp2 4 5
grp3 6-8
</Group_Definition>
```

Il est possible de définir plusieurs groupes associés à un même ensemble de positions :

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

Dans l'exemple précédent, le groupe grp4 représente une seconde conformation-séquence des résidus compris entre les numéros 6 et 8 du fichier backbone. Ces deux groupes possèdent tout deux leur propre espace d'états. L'utilisateur peut les définir différemment l'un de l'autre :

```
<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 garantie que les groupes 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, il est possible d'exprimer simplement l'énergie d'un groupe et l'interaction entre deux groupes. On a :

$$E(grp_i) = \sum_{a \in grp_i} E_a + \sum_{a \in grp_i} \sum_{b \in grp_i, a<b} E_{a,b} \tag{2.11}$$

et

$$E(grp_i,grp_j) = \sum_{a \in grp_i} \sum_{b \in grp_j} E_{a,b} \tag{2.12}$$

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 à la position a et $E_{a,b}$ l'énergie d'interaction entre les rotamères aux positions a et b. Il est possible de visualiser les énergies de groupes sur la matrice d'énergies, figure 2.7. Les contributions à l'énergie d'un groupe se situent dans un carré sur la diagonale de la matrice et les contributions à l'énergie d'interaction entre deux groupes se situent dans un rectangle hors de la diagonale.

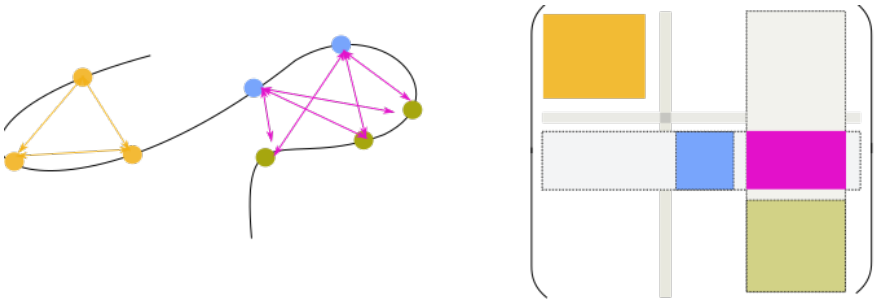

Figure 2.7 – **Les contributions aux énergies de groupes et d'interaction entre groupes dans la matrice d'énergie.** L'énergie des groupes orange, bleu et vert est une somme de termes situés des carrés de même couleur dans la matrice d'énergie. L'interaction entre le groupe bleu et le groupe vert est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

Déclaration de la fonction de score

Il est alors possible 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 petite 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 alors comme une combinaison linéaire des éléments de cette matrice. Pour cela, il fait utiliser la balise `<Optimization_Configuration>` :

```
<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~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. Il est également possible d'utiliser `t(...)` pour indiquer une fonction de seuil :

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

ici A est meilleure que B si $E(A) < 125{,}8$. C'est utile dans le cas d'une utilisation 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 sont forment de matrice. Il n'y a pas d'interaction entre grp3 et grp4 parce qu'ils sont associés à une même partie du système.

2.6 Les fichiers de sortie

Dans le mode HEURISTIC (algorithme MSD), proteus produit en sortie deux fichiers. Un fichier de conformations-séquences dans lequel chaque ligne donne 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 26. Le second fichier est un fichier d'énergies qui donne le résultat de 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 (algorithme 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. Les fichiers d'énergies contiennent la température dans l'entête (la première ligne du fichier).

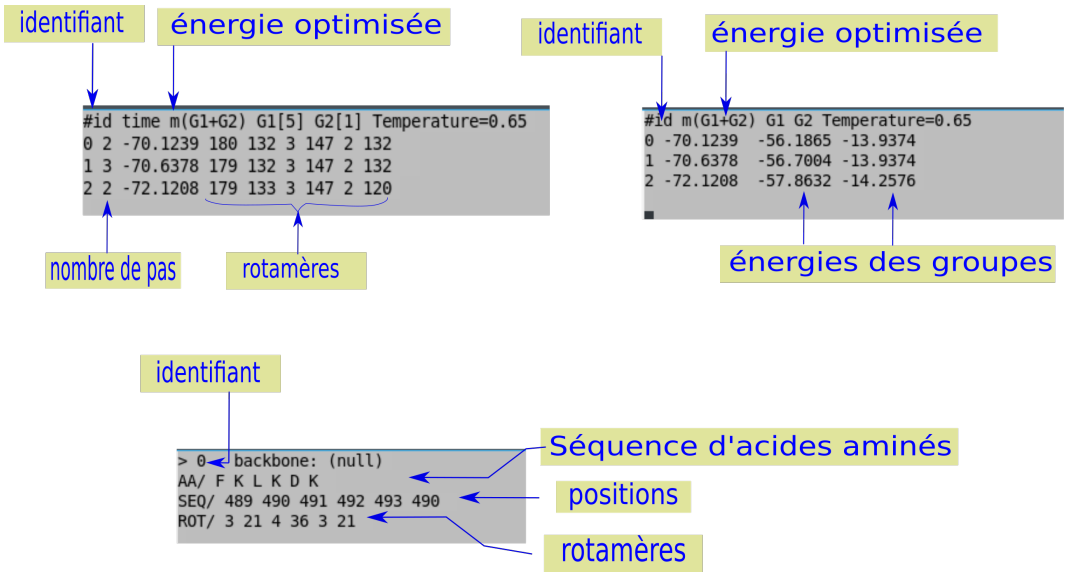

Figure 2.9 – **Les fichiers en sortie de proteus** en haut pour le mode MONTECARLO, en bas pour le mode POSTPROCESS

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 donnée 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 [60] est un ensemble composé :

- d'une base de données de modèles de Markov cachés, où chaque modèle représente une structure 3D d'un domaine de la classification SCOP [61].

- d'une série de scripts qui annotent les séquences données en entrée. Ici, nous utilisons uniquement l'association de chaque séquence au modèle 3D SCOP le plus vraisemblable.

Nous travaillons avec la base de données à la version 1.75, et en conjonction, nous utilisons SAM (version 3.5) [62] et HMMER (version 3.0) [63] deux collections d'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. C'est-à-dire :

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

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

2.7.3 Taux d'identité par position

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

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

avec m le nombre de séquences de A_S. Ce taux d'identité donne une mesure de la ressemblance entre un alignement et une séquence. Cela nous permet de comparer nos séquences calculées à la séquence native. Mais cela n'est pas notre seul objectif. Nous voulons également les comparer à l'ensemble des séquences naturelles de la famille de la native.

2.7.4 Alignements Pfam

La base de données Pfam (Protein families database) [64, 65] regroupe les domaines protéiques connus en familles. Chaque famille est représentée par des alignements multiples de séquences et des profils de 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) [66] qui sont construites à partir de blocs d'alignement très conservés (plus de 40% et 62% d'identités respectivement). Les fréquences des mutations y sont calculées. Le score BLOSUM d'une substitution est alors le logarithme 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 simplement 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 entre eux 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 [67, 68]. 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 à la liste de nos 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 [69], à partir de la formule :

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

avec $f_j(i)$ la fréquence du type de résidu j à la position i. Au lieu de distinguer les 20 types d'acides aminés, nous utilisons six classes de résidus, correspondant aux groupes suivants : {L,V,I,M,C}, {F,Y,W}, {G}, {A,S,T,P}, {E,D,N,Q} et {K,R,H}. Cette classification a été obtenue par un clustering de matrice BLOSUM62 et une analyse des énergies de contact entre résidus dans les protéines [70]. 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 de l'ensemble 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 dans les séquences analysées.

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 page suivante. Le fichier backbone contient, d'une part la description de tous les rotamères possibles à chaque position du système biologique étudié 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 page 47. 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 à son tour 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 utilisée 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.

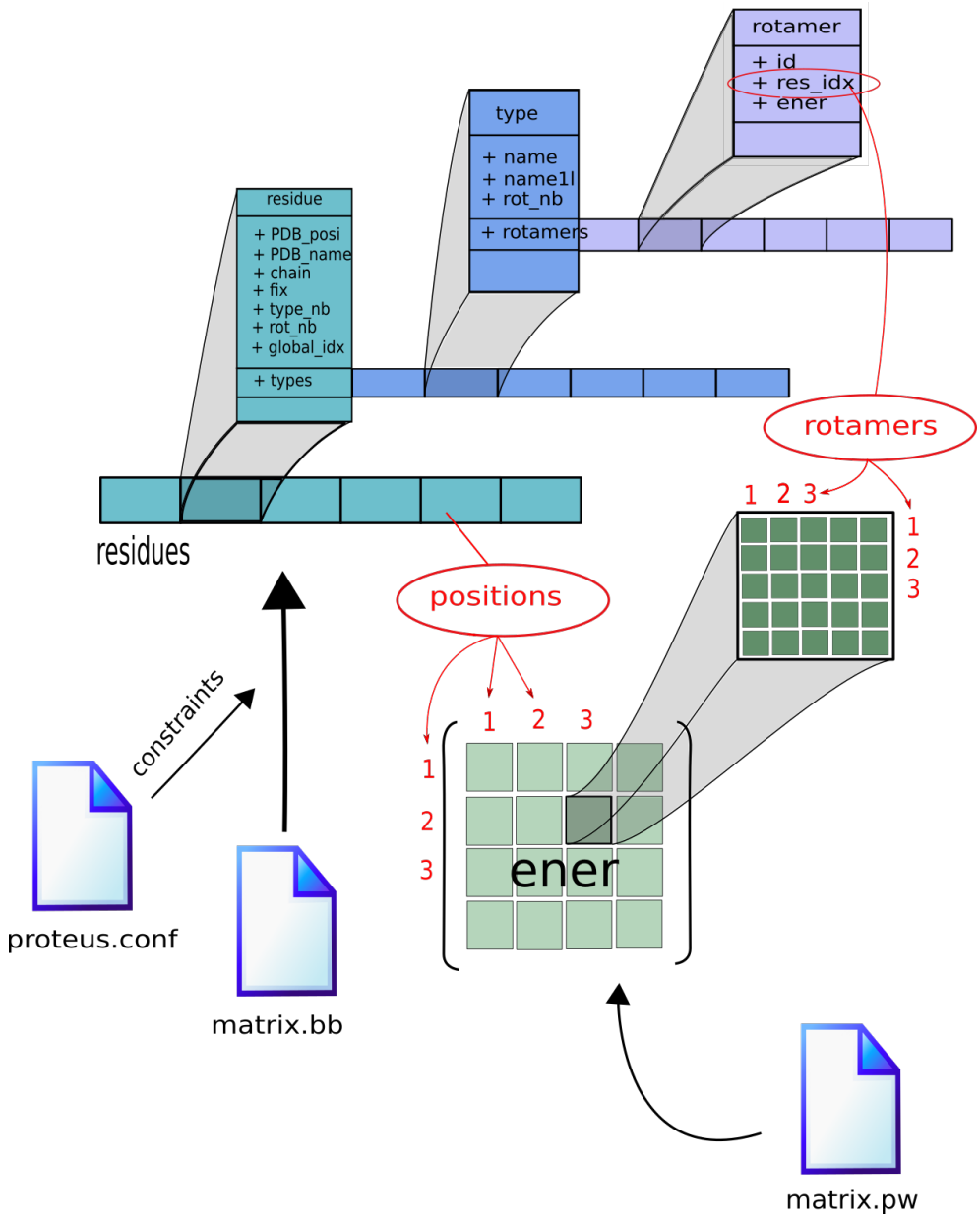

Figure 2.10 – Les principales structures « physiques » dans proteus

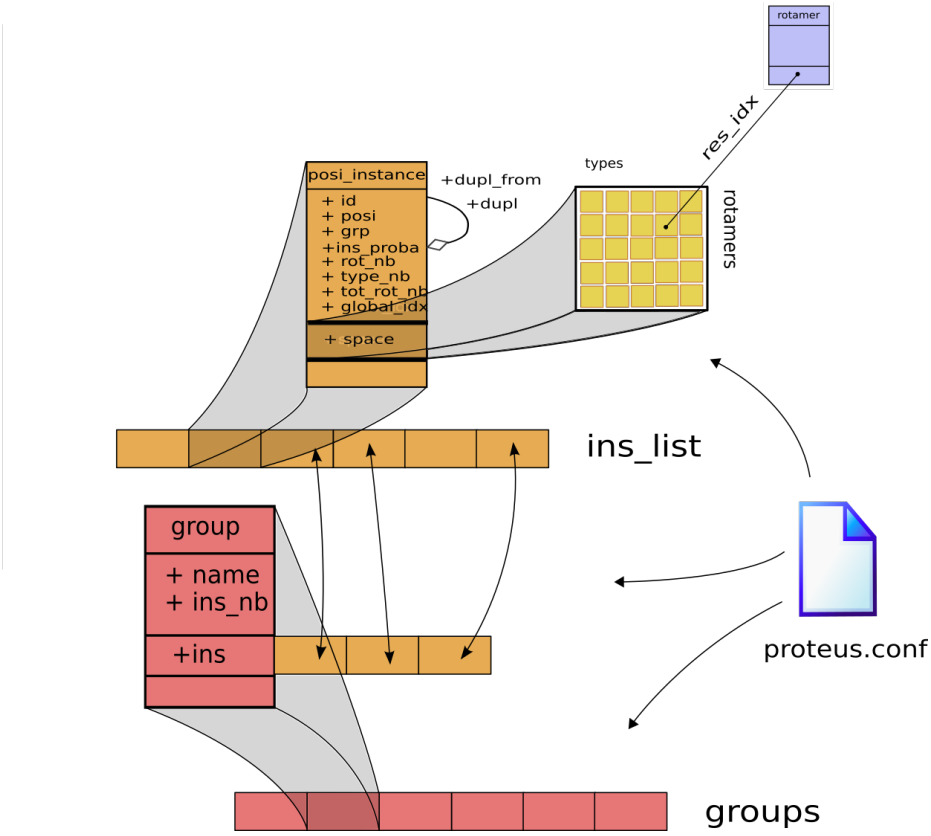

Figure 2.11 – Les principales structures « logiques » dans proteus

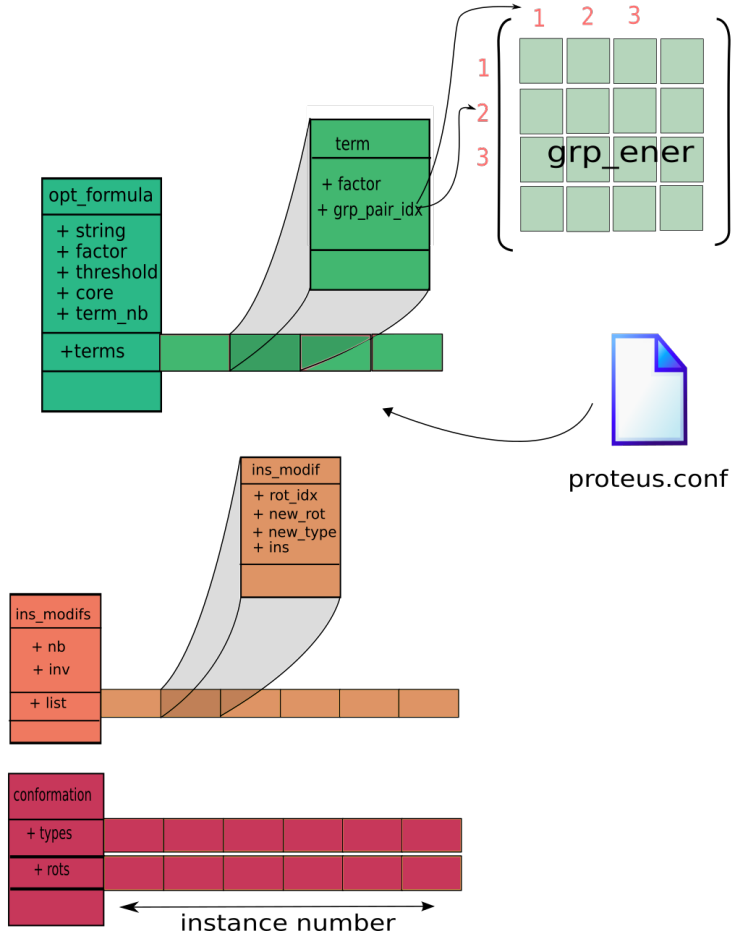

Figure 2.12 – Les principales structures « dynamiques » dans proteus

Chapitre 3

Développements

Ce chapitre expose les modifications faites dans le logiciel Proteus et dans X-plor pendant la préparation de cette thèse. X-plor est un logiciel conçu pour la biologie structurale développé à l'origine par Axel T. Brünger à l'université de Yale [97]. Il propose un langage de script permettant d'exploiter ses fonctionnalités. Pour adapter X-plor 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. X-plor 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'X-plor 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 X-plor. Ils représentent respectivement la position

dans une chaîne polypeptidique, le type d'acide aminé, le segment ou molécule à laquelle appartient le résidu. Un « modèle » est un jeu de coordonnées d'une resclass. Une resclass peut avoir plusieurs modèles.

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.

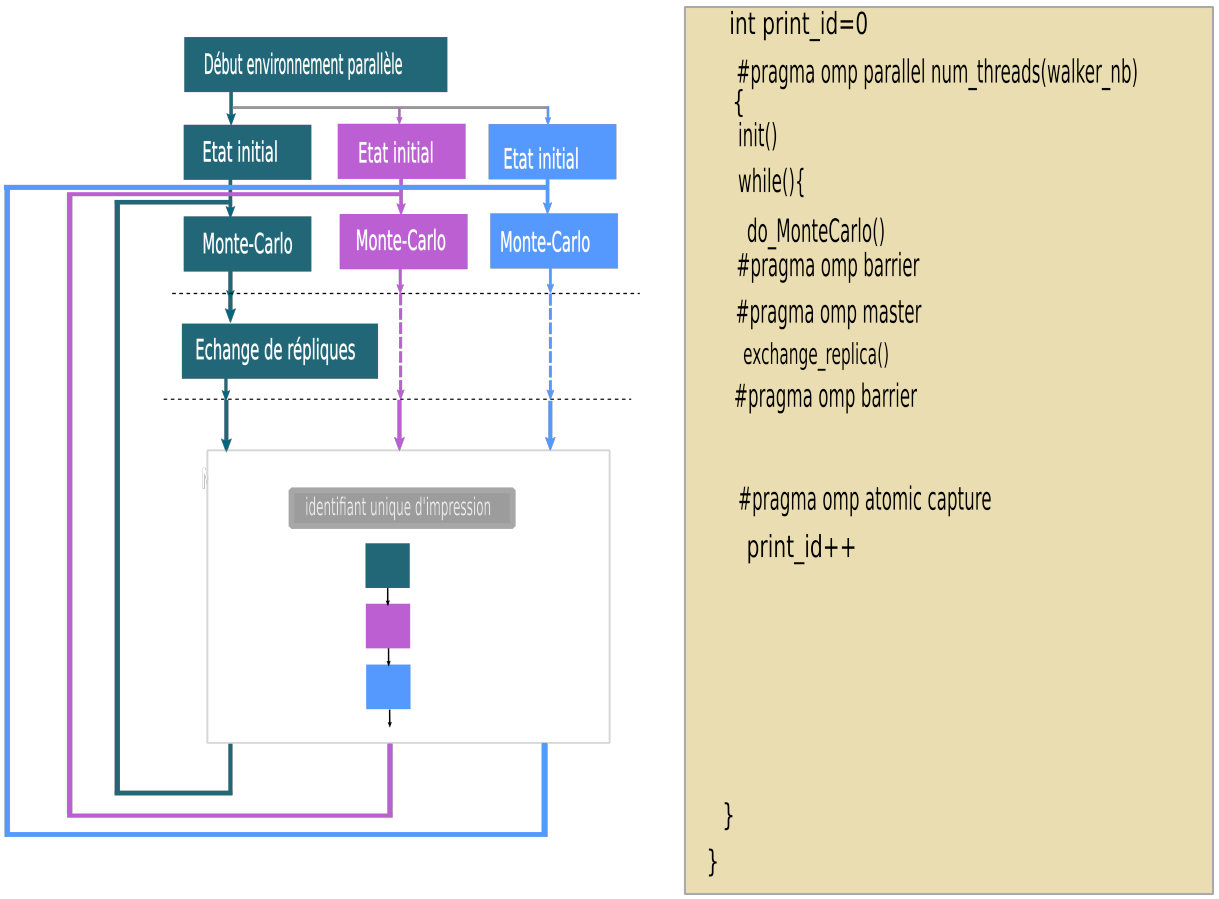

Figure 3.2 – **La correspondance entre les directives OpenMP à droite et le comportement des fils d'exécution à gauche sur un REMC simplifié.** Sont schématisés, la création d'une région parallèle, deux synchronisations (les lignes pointillées à gauche), une région exécutée uniquement par un fil maître,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

3.2.2 REMC dans proteus

Comme nous l'avons vu à la section 1.4.7 page 32, l'algorithme REMC considère plusieurs simulations indépendantes d'un même système. Cela fait de lui un algorithme bien adapté à la programmation parallèle. Deux points demandent une attention particulière : l'échange de répliques et la création de l'identifiant unique d'impression, voir 2.6. Le schéma général de REMC dans proteus est présenté à la figure 3.2. Une région parallèle est initiée par la directive « pragma omp parallel » : les différents marcheurs sont créés. Chacun réalise une trajectoire de type MC jusqu'à un nombre de pas multiple de la période d'échange. A priori l'avancée des marcheurs n'est pas simultanée. Donc une directive « pragma omp barrier » est placée avant le test d'échange, elle garantit que chacun a effectué le bon nombre de pas. Une fois tous les marcheurs bloqués par cette directive, ils sont libérés. La directive « pragma omp master » dédie l'exécution du test et de l'échange de répliques au

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

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 le cycle thermodynamique en 2.1 page 36. 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 différents états possibles des chaînes latérales à l'état déplié. Plusieurs points de vue sont possibles. On peut considérer qu'il existe autant de rotamères à l'état déplié qu'à état replié et qu'ils ont tous la même énergie. Cette possibilité a l'avantage de rendre la probabilité de mutation symétrique, c'est-à-dire que le passage du type t vers t' est aussi probable que le passage de t' vers t. Mais elle a l'inconvénient d'être peu réaliste parce que les rotamères sont les positionnements préférentiels à l'état replié et non à l'état déplié. Nous avons fait évoluer proteus vers un autre point de vue, en considérant qu'une chaîne latérale à l'état déplié n'a qu'un seul rotamère dominant. Cela induit une dissymétrie dans la probabilité de sélection qui est corrigée par une probabilité d'acceptation du type de l'équation 1.4.6 page 31. Cette nouvelle probabilité d'acceptation est décrite en détail en 4.6 page 71.

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

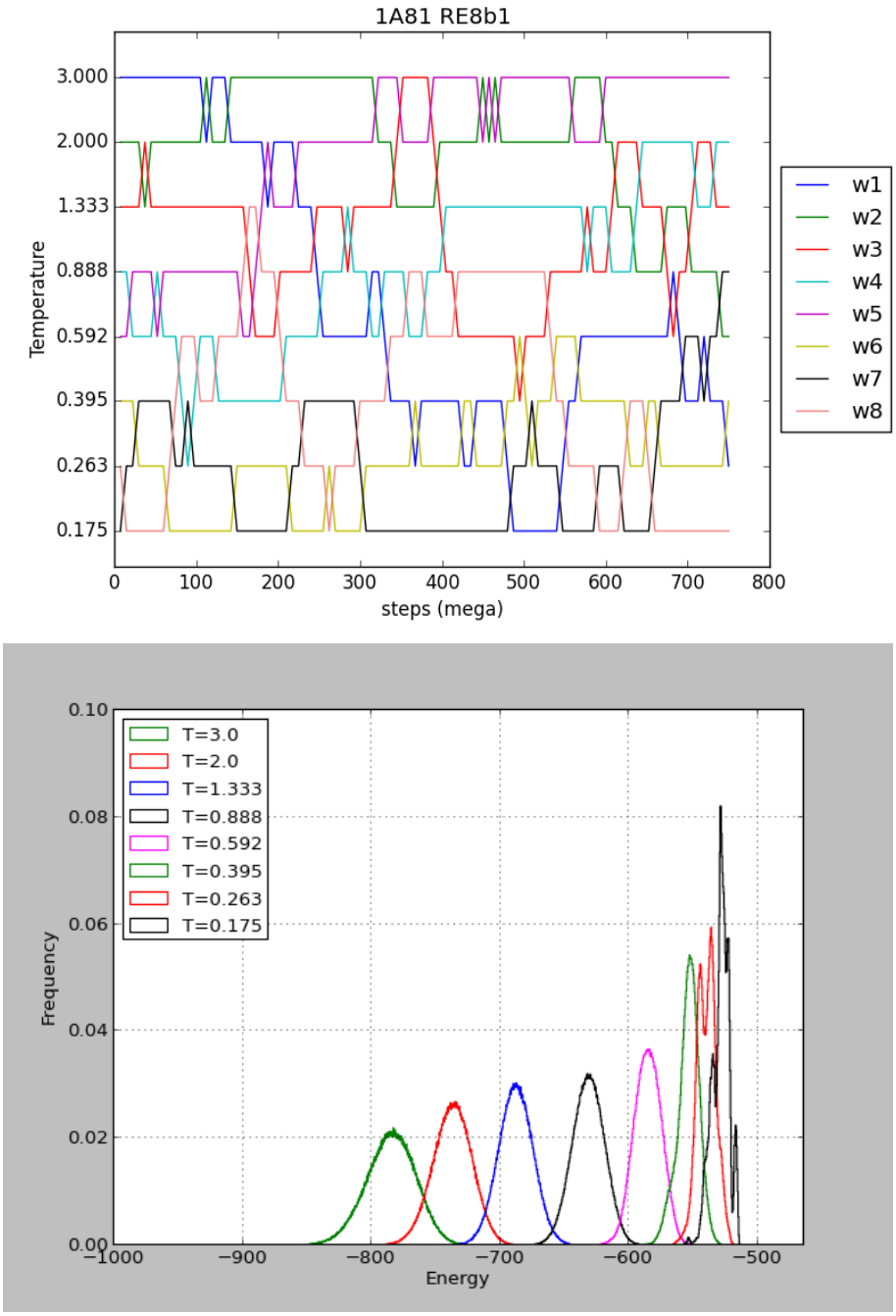

Figure 3.3 – **Comportement de proteus pour un REMC à huit marcheurs.** En haut, la trajectoire des marcheurs dans l'ensemble des huit températures. En bas, la distribution des énergies par marcheurs. On peut imaginer la distribution cumulée des marcheurs et remarquer que son graphe serait proche de 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 page 44 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, voir 2.5.2 pour limiter le chargement des énergies à celles nécessaires à l'exploration.