Implémentation d'un l'algorithme de Monte Carlo pour le repliement de protéines HP

Baptiste Rousseau
M2-BI Université de Paris Cité

14 septembre 2022

1 Introduction

Les méthodes de repliement de protéines à partir de leur séquence s'appuient sur la minimisation d'une fonction d'énergie. Les protéines sont des molécules complexes et leur repliement l'est encore plus, c'est pourquoi nous supposons que des perturbations thermiques appliquée à une protéine l'amèneront vers un minimum local d'énergie. Les processus de repliement des protéines in vivo sont guidés, c'est pourquoi nous pouvons imaginer un paysage énergétique en entonnoir contenant un minimum global significatif. Beaucoup de facteurs agissent sur ce paysage énergétique mais nous n'en considérerons ici qu'un. Le modèle HP permet de réduire la complexité des protéines en considérant uniquement le cas de résidus hydrophobiques ou polaires. Ainsi l'objectif sera d'augmenter les interactions entre résidus hydrophobiques.

Afin de minimiser l'énergie l'approche la plus courante est l'utilisation d'un algorithme de Monte Carlo. Cet algorithme suit une descente de gradient mais peut, selon une probabilité, aller à contre-sens. Cela permet ainsi d'échapper aux minima locaux afin d'augmenter les chances du trouver le minimum global.

2 Matériels and Méthodes

Le modèle de séquence sur lequel nous nous appuierons est un modèle dit "Hydrophobe-Polaire" où les acides aminés sont classifiés en deux catégories, "hydrophobes" et "polaire". La séquence HP d'une protéine n'est alors constituée que d'un enchaînement de résidus H et P.

Nous utiliserons le classement ci-suit :

— hydrophobes : A I L M F V P G W C

— polaires : R N D E Q H K S T Y

Ces résidus seront placés sur une grille réduisant ainsi les coordonnées possibles. En plus de cela nous ne considérerons que le cas d'une grille en 2 dimensions. Chaque résidu peut avoir quatre voisins au maximum sur cette grille. L'énergie de conformation associée à cette grille correspond au nombre de contacts directs entre résidus hydrophobes étant voisins topologique, c'est à dire entre résidus voisins n'étant pas consécutifs dans la séquence. Chaque contact aura une contribution négative de -1 à l'énergie totale. Cette simplification repose sur le fait que les protéines solubles ont tendance à cacher leurs résidus hydrophobes de l'environnement (souvent polaire) en les regroupant dans un noyau hydrophobe.

L'algorithme de Monte Carlo s'applique sur une conformation initiale que l'on cherche à minimiser. Pour obtenir cette conformation initiale nous comparerons deux méthodes, un placement initial linéaire et un placement par marche aléatoire. L'algorithme consistera ensuite en une série d'itérations pendant lesquelles un mouvement pourra être appliqué sur un ou plusieurs résidus.

Il existe plusieurs type de mouvements, tout d'abord les mouvements VSHD et les mouvements pull. Il existe trois mouvements VSHD : end, corner et crankshaft. Ces mouvements sont simples mais ne permettent pas une

grande mobilité de la protéine. Les mouvements de type pull ont eux un plus grand impact sur la conformation globale.

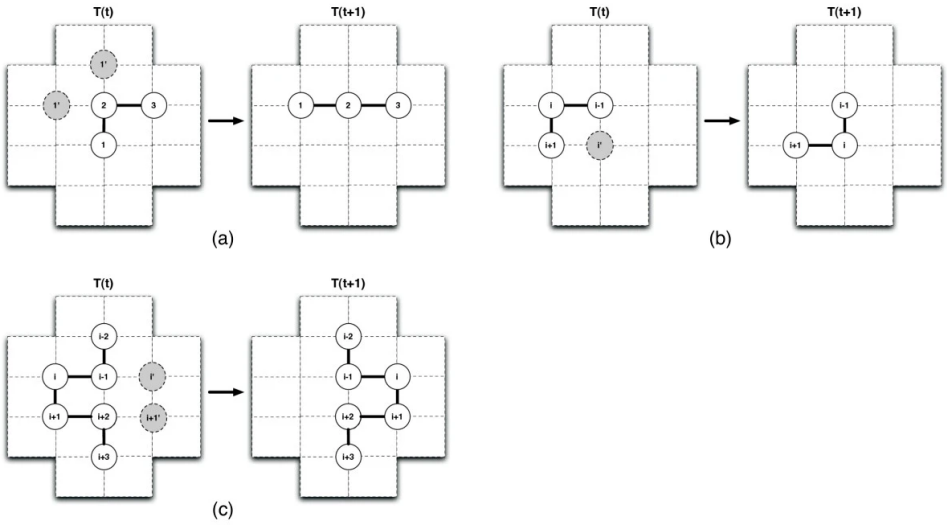

FIGURE 1 – Mouvements VSHD. En 1a, deux positions sont possibles pour le résidu 1, indiquées par les cercles gris. Si une position ou plusieurs positions s'avèrent être libres, le résidu est déplacé au hasard. En 1b, il n'y a qu'une seule nouvelle position potentielle pour un déplacement de type corner. La position doit se trouver sur le plan formé par i - 1, i et i + 1. Enfin, 1c montre le cas d'un mouvement de crankshaft. Ici deux résidus font une rotation autour d'un axe.

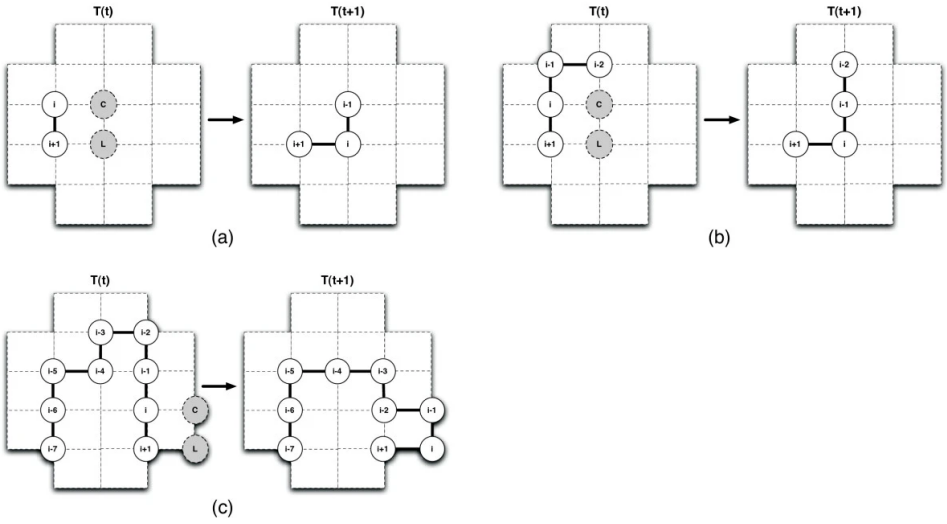

FIGURE 2 – Mouvements pull. En 2a, le cas le plus simple où la position C est occupée par le résidu i - 1 est montré. Ce mouvement est équivalent à un mouvement de coin dans le jeu de mouvements VSHD. En 2b, le résidu i est déplacé vers L et i - 1 vers C. La chaîne est dans une conformation valide et le déplacement est terminé. En 2c, les résidus i jusqu'à i - 3 doivent être tirés jusqu'à ce qu'une conformation valide soit trouvée.

Chaque mouvement peut apporter des modifications à l'énergie totale de la grille, ces modifications vont déterminer sont acceptation. La probabilité d'appliquer un changement est alors définie par la probabilité suivante :

$$Pr[c \to c'] := \begin{cases} 1 & si\ \Delta E \leq 0, \\ e^{\frac{-\Delta E}{T \cdot K_b}} & sinon. \end{cases}$$

où $\Delta E := E(c') - E(c)$ est la différence d'énergie entres les conformations c' et c, T est la température et K_b est la constante de Boltzmann.

Ainsi, dans un cas peu défavorable, la différence d'énergie sera proche de 0 mais positive. L'exponentielle sera alors plus proche de 1 et il y a alors des chances d'accepter néanmoins le mouvement. A l'inverse, avec une différence d'énergie très positive (très en défaveur de la nouvelle conformation) alors l'exponentielle serait plus proche de 0 et le changement moins probable. C'est ce mécanisme qui permet à l'algorithme de Monte Carlo de sortir de minima locaux.

Nous implémenterons aussi un algorithme de Monte Carlo avec échange de réplicas. Ici nous lancerons χ simulations à travers χ réplicas. Chaque réplica a une température qui lui est associé, déterminant alors sa liberté de mouvement. Il est possible d'échanger deux réplica et ainsi d'échanger leur températures :

$$\begin{aligned} Pr[c \to c'] &:= Pr[l(c_i) \leftrightarrow l(c_j)] \\ &:= \begin{cases} 1 & si\ \Delta \leq 0, \\ e^{-\Delta} & otherwise. \end{cases} \end{aligned}$$

où Δ est le produit de la différence d'énergie et de la différence de température inverse :

$$\Delta := (\beta_j - \beta_i) \times (E(c_i) - E(c_j))$$

où $\beta_i = \frac{1}{T_i \cdot K_b}$ est l'inverse de la température du réplica.

L'implémentation de cet algorithme a été réalisée en python (3.10). Cela a permis une implémentation orientée objet à travers quatre classes principales, `Residue`, `Protein`, `Lattice` et `Movement`. Les deux algorithmes que sont celui de Monte Carlo et celui du Replica Exchange Monte Carlo ont été implémenté à travers deux fonctions. La compléxité et lenteur de ce programme vient en grande partie de la représentation des conformation, passant par des matrices, devant être copiées récursivement à chaque calcul de mouvement.

3 Résultats

La performance du programme limitant le nombre d'itération qu'il est possible de réaliser le repliements sont loin de ceux attendus et les énergies obtenues sont au maximum moitié moindre que celles attendues.

ID	E^*	MC	REMC
S1	-9	-4	-4
S2	-9	-4	-3
S3	-8	-3	-1
S4	-14	-3	-1

Table 1 – Résultat des simulations sur les protéines de benchmark

Toutes les simulations ci-dessus ont été obtenues avec les paramètres suivants :

— MC : n-steps=5000, temperature=200

— REMC : n-replica=5, max-steps=500, local-steps=100, temperature-min=160, temperature-max=220

Les limites de cet algorithme sont notamment dues à la position initiale ne laissant que peu de liberté de mouvement. Nous pouvons suspecter des erreur dans l'implémentation des mouvements pull, qui permettrait pourtant une bien meilleur mobilité.

4 Discussion

La réalisation de ce projet a soulevé plusieurs difficultés. Premièrement les simulations deviennent extrêmement lentes à mesure que le nombre d'itération augmente. Python ne semble alors pas adapté pour ce genre

d'implémentation à l'instar de langages compilés tels que le C++, utilisé par les auteurs de l'article.

Le peu de temps dont nous disposions ne nous a pas permis de développer de bonnes pratiques de programmation. Notamment l'implémentation systématiques de tests (à travers pytest), permettant la création d'exemples reproductibles. Mais aussi le calcul du temps de calcul par simulation ou bien mettre en place des simulation non aléatoires.

Enfin une autre difficulté fût la correction d'erreurs de l'article. En effet il ne fait pas mention de l'utilisation de la constante de Boltzmann. Cette constante est cependant utilisée dans le programme lié à l'article, même si la valeur utilisée correspond à la dimension en $kcal/molK$, dimension étrange à l'instar de la dimension classique de cette constante qu'est le J/K.

Appendices

Annexe A Structure du code

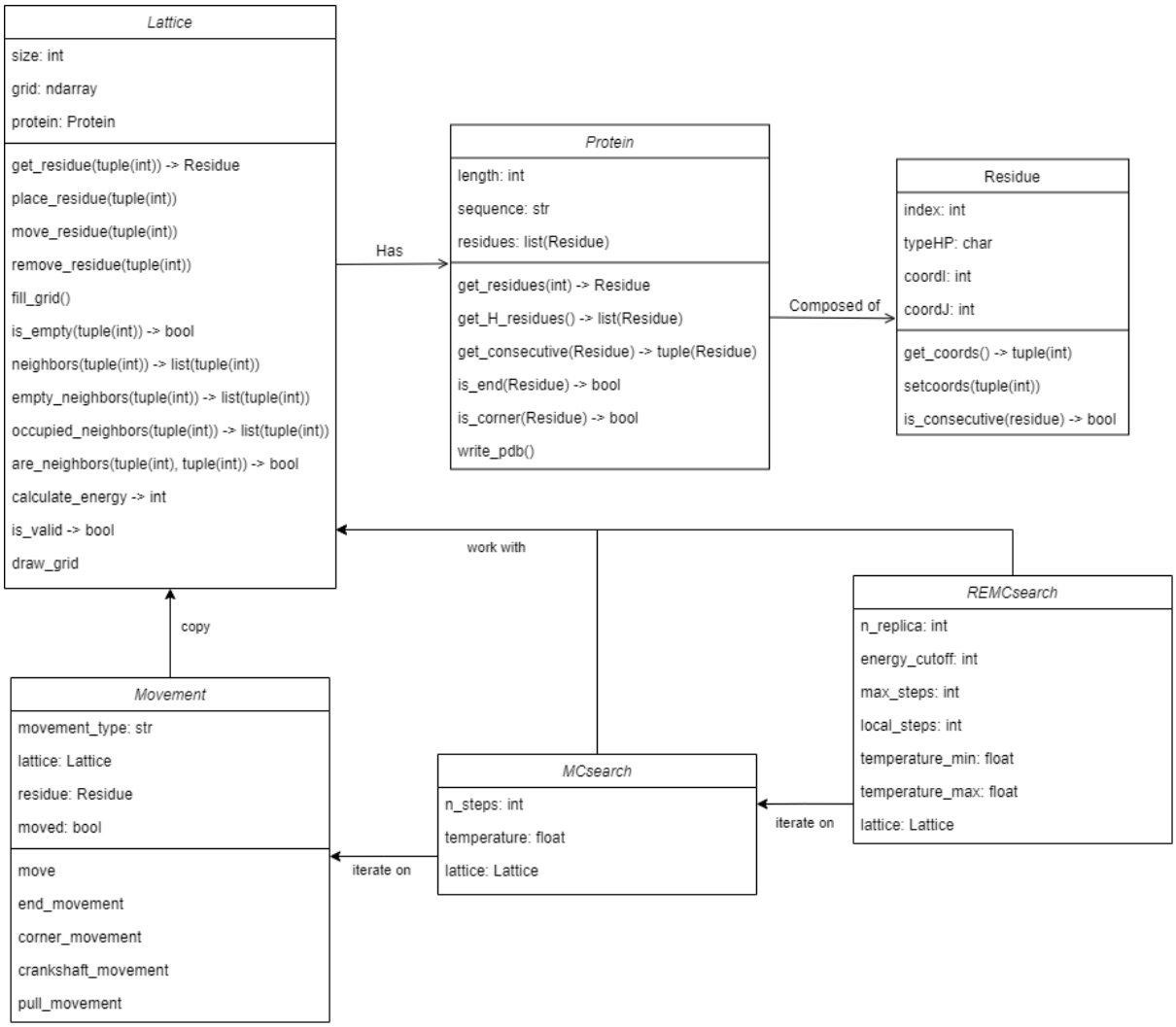

FIGURE 3 – Structure des classes et fonctions principales

Annexe B Sorties des runs de benchmarking

```
Running test case: HPHPPHHPHPPHPHHPPHPH with optimal energy: -9
Running MC with parameters: n-steps=5000, temperature=200
Final lattice with energy of -4
+-----------+
|           |
|        PP |
| PH     HH |
| HHHPHHPHP |
| PP  PP  H |
|           |
+-----------+
Running REMC with parameters: n-replica=5, energy-cutoff=-9, max-steps=1000, local-steps=100, temper
Final lattice with energy of -4
```

```
+--------------+
|              |
| PH        PP |
| HHHPHPPHPHHH |
| PP        HP |
|              |
+--------------+

Running test case: HHPPHPPHPPHPPHPPHPPHPPHH with optimal energy: -9
Running MC with parameters: n-steps=5000, temperature=200
Final lattice with energy of -4
+------------------+
|                  |
|             PPHP |
| HHPPHPPHHPPHHHHP |
|        PP  PP    |
|                  |
+------------------+
Running REMC with parameters: n-replica=5, energy-cutoff=-9, max-steps=500, local-steps=100, tempera
Final lattice with energy of -3
+--------------------+
|                    |
| HPP           PP   |
| HHHPPHPPHPPHPPHHH  |
| PP                 |
|                    |
+--------------------+

Running test case: PPHPPHHPPPPHHPPPPHHPPPPHH with optimal energy: -8
Running MC with parameters: n-steps=5000, temperature=200
Final lattice with energy of -3
+---------+
|         |
|  PP     |
| PPP PH  |
| PHH PHP |
| HHHPPPP |
| PP   P  |
|     HH  |
|         |
+---------+
Running REMC with parameters: n-replica=5, energy-cutoff=-8, max-steps=500, local-steps=100, tempera
Final lattice with energy of -1
+----------------------+
|                      |
| PP               PHH |
| HHHPPPPHHPPPPHHPPP   |
| P                    |
| P                    |
|                      |
+----------------------+
```

The source code for this paper is openly available in a github repository, you can access it here.

This document was written in LaTeX and generated by overleaf, you can find the source code here.