Nicolas Silva

Université Paris Diderot (Paris 7)
Master 2 bio-informatique (m2bi)
Septembre 2019

Repliement protéique simplifié via un algorithme de Monte Carlo.

Basé sur l'article : *Thachuk C, Shmygelska A, Hoos HH. A replica exchange Monte Carlo algorithm for protein folding in the HP model. BMC Bioinformatics. 2007 Sep 17;8:342. PubMed PMID: 17875212; PubMed Central PMCID: PMC2071922.*

Introduction:

Les progrès technologiques ont permis ces dernières années d'obtenir rapidement et en masses des données de séquençages protéique. Cependant, le séquençage ne produit que la séquences d'acides aminés qui constitue la protéine d'intérêt, mais ne donne à priori aucune information sur sa structures secondaire, tertiaire et quaternaire. Un des objectifs lié à l'arrivage massif de données est de pourvoir prédire les repliements protéiques rapidement sans avoir à effectué des analyses en laboratoire comme des cristallographies au rayon X ou des imageries à résonance magnétiques. Pour ce faire, il est possible de créer des modèles de repliement aléatoire afin de tenter d'extrapoler au mieux un repliement protéique. Depuis plusieurs années déjà, différentes équipes ont tenté de créer des modèles capable de prédire des repliement de protéines. Parmi les différents travaux effectués, ceux de Thachuk et son équipe sont basé sur un algorithme de Monte Carlo.

Mon travail ici, a constitue en une inspiration de l'étude de Thachuk *et al.* (2007), afin d'implémenter un modèle simple de repliement protéique avec des algorithme de Monte Carlo.

Matériel et méthodes:

Tout les fichiers et documents sont disponible sur GitHub à cette adresse (https://github.com/nicolassilva/MonteCarloAlgorithm_Search). Sur le git, est également disponible un fichier .yml avec lequel la création d'un environnement conda identique à celui dans lequel le projet a été réalisé est possible.

L'algorithme créer ici s'effectuera avec un modèle de séquence "Polaire/Hydrophobe" (Dill 1985). Ce modèle classifie les acides aminés qui constituent une protéine en deux catégorie : les acides aminés polaires et les acides aminés hydrophobe. La séquence protéiques est donc modélisé par un enchainement d'acides aminés H et P. Pour ce modèle, une séquence aléatoire a été utilisée et est disponible dans le répertoire *data* du git. La séquence protéique est ensuite projetée en ligne droite sur une grille en 2D.

L'algorithme de Monte Carlo implémenté ici consiste en la prise aléatoire d'un avide aminé de la séquence et à lui faire effectuer un mouvement tout aussi aléatoire

parmi des mouvements définis. Il est considéré dans ce modèles des mouvements simple. En effet, chaque acides aminé peut effectué un déplacement d'une position, d'une case dans n'importe quelle direction à condition que la case d'arrivée soit vide.

Afin d'optimiser le repliement protéique, le modèle cherche à trouver le repliement 2D le plus énergétiquement stable, c'est-à-dire le modèle qui détient l'énergie la plus basse. Cette énergie de conformation correspond au nombre d'acides aminés hydrophobes voisins dans la grille 2D (Lau & Dill 1989). Chaque couple HH contribue de -1 à l'énergie. Si l'énergie suite au mouvement est inférieure ou égale à l'énergie initiale, alors la nouvelle formation est conservée et un nouveau déplacement aléatoire sur un acide aminé au hasard de la séquence est réitéré.

Pour l'exécution de cet algorithme simplifié de Monte Carlo, un seul paramètre doit être défini préalablement. Il s'agit du nombre de fois que l'algorithme de Monte Carlo tournera, noté *nstep*. La valeur de *nstep* est un entier positif à passer en second argument en ligne de commande dans un terminal UNIX après le nom du fichier contenant la séquence d'acides aminés.

Résultats:

L'algorithme implémenté réussi a effectué un repliement protéique environ un fois sur 4. Pour les tests effectués sur la chaîne protéique d'intérêt (chaîne de 9 acides aminés), *nstep* a été fixé à 500.

Il arrive que dans certain essais, l'algorithme n'arrive pas à effectuer un repliement de la chaîne protéique (Figure 1), et l'énergie initiale de la conformation est identique à l'énergie finale. Cependant dans certain cas, l'algorithme réussi a replié la protéine et la conformation finale a une énergie plus faible que la conformation initiale (Figure 2). Il est également arrivé que dans certain cas, l'algorithme effectue un repliement sans que la nouvelle conformation ne présente d'énergie plus faible.

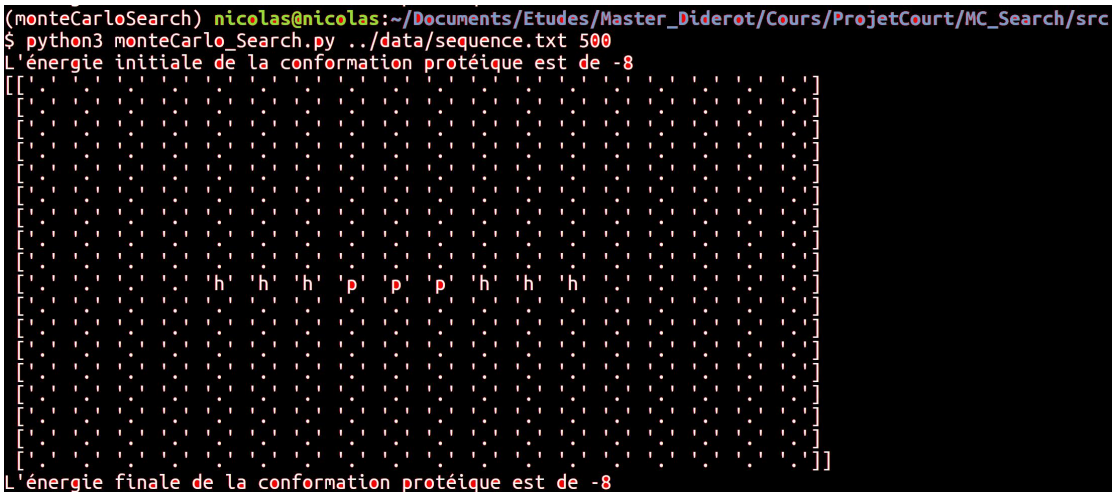

Figure 1: Repliement protéique via l'algorithme de Monte Carlo exécuté 500 fois. L'algorithme n'a pas pu effectuer de repliement (l'énergie finale est la même que l'énergie initiale).

Figure 2: Repliement protéique via l'algorithme de Monte Carlo exécuté 500 fois. L'algorithme a pu effectuer un repliement (l'énergie finale est plus basse que l'énergie initiale).

Discussion:

L'algorithme de Monte Carlo pour le repliement protéique implémenté ici permet d'effectué un léger repliement de la chaîne protéique testé en adoptant une conformation plus faible en énergie.

Cependant, le modèle présenté présente parfois des incapacité à effectuer le repliement d'une chaîne protéique. Cet échec peut être imputé à plusieurs phénomène comme notamment le fait que l'algorithme de Monte Carlo repose sur une choix au hasard d'acides aminés à déplacé et également sur une mouvement au hasard. Cependant, ici je ne pense pas que ces échecs sont dus au hasard. En effet, les mouvements implémenté sont beaucoup plus simple que dans l'étude de Thachuk et de son équipe. Ils ont modélisé plusieurs type de mouvements : les *VSHD* qui comprennent les *End moves*, les *Corner moves* et les *Crankshaft moves* ; et les *Pull moves*. Ici les mouvement sont les même pour chaque acides aminé et ne tiennent pas compte de la conformation de la chaîne protéique ce qui n'induit pas autant de flexibilité que dans l'étude de Thachuk *et al*. De plus, mon travail ne porte que sur un algorithme de Monte Carlo simple et ne prend pas en compte les échanges de répliques qui eux aussi permettent une meilleurs flexibilité de repliement.

Bibliographie:

1. Dill K.A. (1985). Theory for the folding and stability of globular proteins. *Biochemistry, 24(6)*, pp.1501-1509.
2. Lau K.F. & Dill K.A. (1989). A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules, 22(10)*, pp.3986-3997.

Annexes

Structures du programme:

Le code python implémenté pour l'algorithme de Recherche de Monte Carlo à été construit de la manière suivante:

- Une classe **File** avec une méthode *readFile* permettant de lire le fichier contenant la chaîne d'acides aminés.
- Une classe **Array** avec trois méthodes. La première *createArray* qui permet de créer une matrice de deux fois la taille de la séquence étudié ; la seconde *energy* qui permet de calculer l'énergie d'une conformation ; et la troisième *writeFile* qui permet d'écrire dans un fichier la matrice avec la conformation finale obtenue.
- La séquence est importé et une matrice est créée avec la séquence au centre de celle-ci. Les coordonnées des acides aminés dans la matrice sont alors stockés dans un dictionnaire, et l'énergie de la conformation est calculée.
- Commence ensuite l'algorithme de Monte Carlo. Un acide aminé est tiré au hasard dans la séquence, ainsi qu'un des 8 mouvements modélisés. Si la case d'arrivé du dit acide aminé est disponible (c'est-à-dire pas déjà occupée par un autre acide aminé) alors le déplacement est autorisé et les autres acides aminés suivant déplacés aussi.
- Une nouvelle matrice avec la nouvelle séquence est créée et l'énergie associée à cette nouvelle conformation est calculée. Si la nouvelle énergie est inférieure ou égale à l'énergie de l'ancienne conformation, alors la nouvelle conformation est conservée. Sinon le déplacement est annulé.
- L'algorithme de recherche de Monte Carlo recommence ensuite autant de fois que le *nstep*.
- A la fin du script, la matrice ainsi que l'énergie de la dernière conformation conservé est affichée. Et la matrice est sauvegarder dans un fichier *.txt*.

Exemple d'utilisation du programme:

Dans un premier temps il faut obtenir la chaine protéique d'intérêt sous forme d'enchainement de 'h' et de 'p'. Il doit s'agir d'un fichier .txt.
Il faut ensuite télécharger le fichier monteCarloSearch.yml de mon git et créé l'environnement conda associé. De cet environnement il faut alors exécuté le script python monteCarlo_Search.py en passant comme argument le nom du fichier et le nombre d'itération (*nstep*). Un fichier results.txt contenant la matrice avec la conformation finale est ensuite créé dans un dossier *results*.

Exemple : python3 monteCarlo_Search.py sequence.txt 500