Threading moléculaire par double programmation dynamique

Anthony Jaquaniello

September 2019

1 Introduction

Le threading moléculaire est une technique de prédiction de structure tertiaire consistant à "enfiler" (to thread) une séquence d'acides aminés sur une structure protéique et à mesurer l'adéquation entre la séquence et la structure à l'aide d'un potentiel énergétique. Cette procédure répétée sur une banque de structure permet d'identifier celle correspondant au mieux à une même séquence, c'est donc une technique très utilisée en modélisation moléculaire. Le but de ce projet a été d'implémenter un programme de threading en se basant sur la double programmation dynamique, un algorithme utilisé par le programme THREADER. La programmation dynamique est un concept selon lequel un problème est décomposé en plusieurs sous problèmes, chacun étant résolu individuellement afin d'appporter une solution optimale au problème originel. Cette conception a été utilisé notamment dans les algorithmes de Needleman & Wunsch ou de Smith & Waterman pour l'alignement de séquences biologique. Avec de la programmation dynamique simple l'on cherche un alignement optimal entre deux séquences, qui sont des objets à une dimension, linéaires. Cependant, l'utilisation de la programmation dynamique simple pose problème lorsque l'on cherche à aligner des structures. En effet, une structure n'est pas forcément linéaire, mais définie par plusieurs paramètres (par exemple, deux acides aminés identiques n'ont pas forcément le même environnement physico chimique). On cherche l'alignement optimal d'une séquence à un "template". Pour pallier aux faiblesses de la programmation dynamique simple nous avons utilisés la programmation dynamique double. Elle consiste à construire une matrice (résidu x élément structural) dite de "haut niveau", et pour chaque couple (chaque case), construire une matrice dite "de bas niveau" où l'on réalise la première programmation dynamique. La fonction de score utilisée est un potentiel énergétique dépendant de la distance entre deux carbones alphas et dela nature des résidus: le potentiel statistique DOPE (Discrete Optimized Protein Energy). Le score minimal (optimal) de chaque matrice de bas niveau étant ensuite incorporé dans la matrice de haut niveau. Une fois remplie, elle servira à une seconde programmation dynamique, aboutissant à un score d'adéquation final (une énergie en kcal/mol) entre la structure et la séquence donnée en entrée.

2 Matériel et Méthodes

2.1 Python

Le script informatique a été réalisé en langage Python, sous sa version 3.7.4.

2.2 Packages

Les packages utilisés dans le script correspondent à des packages de base de Python, excepté numpy. Ce dernier a été utilisé sous sa version 1.16.5.

2.3 FASTA et PDB

Le code d'accession dans la PDB des fichiers utilisés pour le test du programme sont mentionnés dans la partie Résultat. Tous ont été extrait du site de la PDB : https://www.rcsb.org

2.4 Documentation

La documentation a été générée à l'aide de Doxygen 1.8.11. Elle est disponible à l'adresse:
https://github.com/AnthonyJaquaniello/projet_court

3 Résultats

Pour une même structure (colonne) nous avons regardé l'adéquation avec les séquences (lignes). Le témoin positif est en gras, il est sensé posséder le score le plus bas (la meilleure adéquation).

	1lry.pdb
1lry.fasta	**-1263.14**
1def.fasta	-1234.93
	-
3e8v.fasta	-1098.60
2xri.fasta	-1083.52
1vfl.fasta	-1175.67
1bs5.fasta	-1235.0

Pour cette expérience, la valeur des deux derniers paramètres à été fixée à 50: nous avons donc aligné les 50 premiers résidus des séquences avec les 50 premiers carbones alphas de la structure. Pour plus de sensibilité, il conviendrait d'augmenter ces limites de tailles, cependant le temps de calcul s'en trouverait rallongé (cf table de correspondance ci dessous).
Nous observons que le score d'adéquation est le plus bas lorsque l'on "enfile" la séquence 1lry.fasta sur la structure 1lry.pdb, or ce score est une énergie en kcal.mol donc plus elle est basse, plus la stabilité est grande. Le témoin positif est donc validé. Le programme a su reconnaître quelle était la séquence qui correspondait le mieux à la structure testée. Cependant quelques difficultés ont été rencontrées (cf Discussion). Nous pouvons également observé la proximité de score obtenu entre 1def, 1lry, et 1bs5, toute les trois étant des peptides deformylase. 3e8v, 3l87, 2xri, et 1vf sont respectivement: transglutaminase, hydrolase, exoribonucléase, adenosine deaminase.

Taille	Temps de calcul
50x50	1min12
75x75	6min7
100x100	18min50

Le temps de calcul en fonction de la longueur de al séquence en entrée et de la structure en entrée, est donnée par la table ci dessus.

4 Discussion

Les résultats présentés dans la partie du même nom sont satisfaisant mais plusieurs limites existent. Tout d'abord des résultats surprenants ont parfois été observés lors de test avec d'autre structures que 1lry.pdb, le score le plus bas n'étant pas attribué au témoin positif. Après de nombreuses révisions du code, le problème n'a toujours pas été entiérement résolu. Plusieurs pistes avaient été envisagées mais sans succés, parmi celles-ci figuraient:

- Attribuer une énergie maximale lorsque la distance (en Angström) entre deux carbones alphas est nulle (sous entend qu'il s'agit d'une distance entre un même carbone alpha).
- Faire une moyenne des potentiels énergétiques DOPE entourant la valeur de distance (moyenne de la borne inférieure et supérieure).
- Sélectionner le minimum de chaque matrice bas niveau et de la matrice de haut niveau. En effet, le minimum ne s'avère pas toujours être la valeur inférieure droite de la matrice.

Une solution viable qui pourrait être envisagée serait de régler les valeurs des coût de gap lors de la programmation dynamique. En effet dans le programme ceux ci sont nuls. Or il serait possible d'attribuer une valeur de gap en fonction de la structure secondaire dans laquelle se trouve les résidus concernés, comme le suggère Jones[3]. De plus, un filtre des pdb pourrait être envisagé, il consisterait à éliminer les pdb de "mauvaise" qualité. En effet plusieurs pdb ont dû être retirés car le début ne correspondait pas au début de la séquence du fichier fasta.

5 Bibliographie

1. Jones, D.T., Taylor, W.R. & Thornton, J.M. (1992) A new approach to protein fold recognition. Nature. 358, 86-89.
2. Jones, D.T., Miller, R.T. & Thornton, J.M. (1995) Successful protein fold recognition by optimal sequence threading validated by rigorous blind testing. Proteins. 23, 387-397.
3. Jones, D.T. (1998) THREADER : Protein Sequence Threading by Double Dynamic Programming. (in) Computational Methods in Molecular Biology. Steven Salzberg, David Searls, and Simon Kasif, Eds. Elsevier Science. Chapter 13.

6 Remerciements

Je tiens à remercier Monsieur Gelly pour sa disponiblité, ses conseils, et ses explications.

7 Annexe

Dans cette annexe nous allons décomposer le code et évoquer les différentes difficultés rencontrées lors de la conception du projet.

7.1 Analyse du code

Le code peut se décomposer en quatres bloc:

1. Un bloc interface utilisateur.
2. Un bloc de construction de la matrice de distance.
3. Un bloc de construction des matrices bas niveaux.
4. Un bloc de construction de la matrice de haut niveau.

Le premier bloc ayant un intérêt limité en terme d'analyse, il ne sera pas traité.

Concernant le second bloc nous avons, pour chaque carbone alpha du fichier.pdb, extrait les informations pertinentes (coordonnées dans l'espace et nom du résidu) et les avons stockées dans des objets de la classe CarbonAlpha. Cette classe possède une méthode permettant de calculer chaque distance inter-carbone alpha et ainsi de remplir une matrice de distance euclidienne (matrice de contact).

Concernant le troisième bloc (le principal), nous avons construit une matrice de stockage (prime_matrix) où chaque ligne correspond à un acide aminé de la séquence et chaque colonne à un carbone alpha de la structure. Itérativement, nous fixons un couple, puis construisons une matrice de bas niveau, siège de la première programmation dynamique. Conceptuellement cela revient à chercher l'alignement optimal (minimum énergétique) entre la structure et la séquence, sachant qu'un acide aminé donné de la séquence est fixé/aligné à un carbone alpha donné de la structure.
Soit k la ligne et l la colonne, chaque case de la matrice de bas niveau est remplie comme suit: La valeur dans chaque case est le minimum entre la valeur inscrite dans la case de gauche (k,l-1), celle du haut(k-1,l), et celle supérieure gauche (k-1,l-1). A ce choix minimal est additionné la valeur de potentiel DOPE entre le couple actuel (k,l) et celui fixé (i,j), cette valeur est obtenue à l'aide de la distance entre les deux couples et du nom des résidus impliqué (table dans le fichier dope.par.txt). Le processus continue jusqu'à aboutir à une valeur de potentiel optimale dans la case inférieure droite. Chaque matrice de bas niveau est ensuite stocké dans une case de la matrice de stockage, case étant celle du couple fixé au départ (case (i,j)). Une matrice bas niveau peut être représentée comme suit:

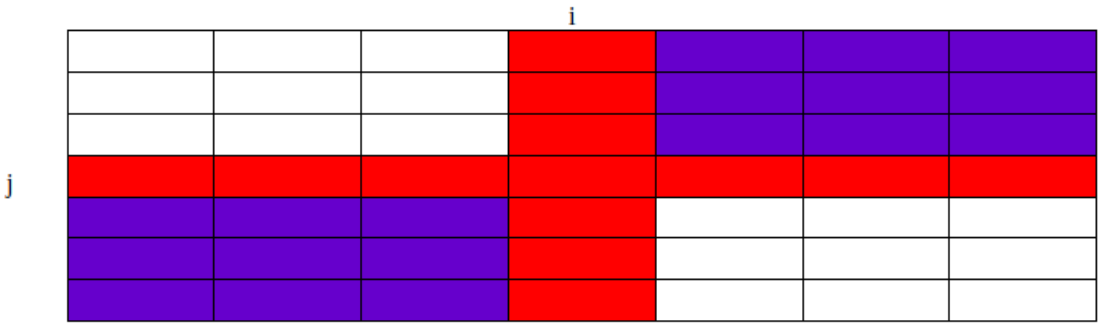

Figure 1: Exemple de matrice de bas niveau (7x7)

Pour un couple acide aminé/carbone alpha (i,j) fixé on a la matrice de bas niveau ci dessus, avec k lignes et l colonnes. Lorsque les valeurs de k ou de l sont telles qu'on se trouve dans une des cases en violet, une pénalité (constante positive) est apportée afin de "forcer" les choix à se diriger vers la case inférieure droite, ce sont des zones interdites. Dans le programme cette constante de "forcing" a une valeur de 50.

Enfin le troisième bloc correspond à la construction des matrices de haut niveau. Dans chacune des case on récupère la valeur minimale de la matrice de base niveau stockée dans la case correspondante dans la matrice de stockage. Une seconde programmation dynamique a lieu, dont le principe est quasi-identique à la première. La valeur d'adéquation renvoyée est celle contenue dans la case inférieure droite de la matrice de haut niveau: c'est le score d'adéquation final entre la séquence et la structure, en kcal/mol.

7.2 Difficultés rencontrées

La première difficulté du projet a été de définir les relations qu'entretiennent les diffèrentes matrices entre elles, ainsi que le concept biologique derrière chacune d'elle. A cela s'ajoute la difficulté pratique due à l'absence de pratique de la programmation dynamique avant.