Nom : Fourteau
Prénom : Ludovic
M2BIB & GitHub : https://github.com/Zygnematophyce/Protein_sequence_threading

# Projet court de bio-informatique : Conception d'un programme Python basé sur l'enfilage de séquences protéiques par double programmation dynamique

## 1. Introduction

La question de déterminer une structure tertiaire d'une protéine à partir de sa structure primaire (une suite d'acides aminés) est un enjeu majeur afin de modéliser la représentation structurale encore inconnue de la protéine. Plusieurs méthodes et algorithmes existent pour ce type de modélisation, mais il manque souvent la validation en comparant la solution structurale théorique avec des données expérimentales de description de la structure des protéines, obtenues par exemple avec les techniques d'analyse comme la RMN ou les Rayons-X et de leurs distributions dans les bases de données.

Le but de ce projet a été d'utiliser la méthode par enfilage de séquences protéiques (ou protein sequence threading en anglais) par double programmation dynamique d'après l'article de Jones (1998) avec comme finalité la modélisation de la structure théorique tertiaire d'une protéine.

## 2. Cadre théorique

La programmation dynamique considère qu'il existe une similitude entre l'optimisation de l'enfilage d'une séquence protéique sur une structure et le fait de trouver un alignement optimal entre 2 séquences protéiques. En d'autres termes, l'enfilage est simplement l'alignement des acides aminés sur une position de séquences dans l'espace.

Globalement, il s'agit d'utiliser deux matrices avec un algorithme de double programmation dynamique, qui sont nommées de la manière suivante : 1) matrice de bas-niveau (L -ow-matrix) et 2) matrice de haut-niveau (H – hight-matrix) ; ces matrices serviront à optimiser l'algorithme, sachant que la matrice de bas-niveau est dépendante de la matrice de haut-niveau.

Les étapes de la programmation et de l'algorithme sont les suivantes :

- Dans un premier temps, il s'agit d'effectuer l'ouverture dans les fichiers de la séquence d'intérêt. Tous les couples possibles d'acides aminés sont stockés dans une liste et sont associés à un potentiel statistique.

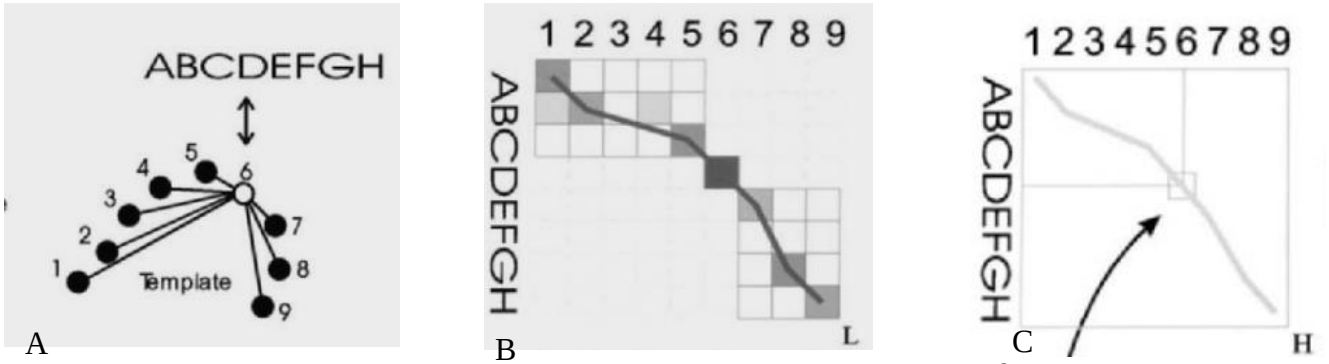
- Les distances pour tous les acides aminés entre eux est créées et stockées. La distance utilisée est la distance euclidienne en Angström.

- Ensuite pour un acide aminé et une position donnée dans la structure pour la matrice de haut-niveau, une matrice de bas-niveau est créée.

- Pour chaque couple acide aminé-structure de la matrice de bas-niveau un potentiel de statistique est associé grâce à la distance qui sépare les deux éléments. La distance entre deux éléments se fait par rapport aux Cα (carbone alpha des acides aminés). Il s'agit par exemple de la distance ASN CA en position 1 et la LEU CA en position 1 de la structure.

- Une fois la matrice de bas niveau complète, on additionne la somme des scores associés au trajet optimal. Une première programmation dynamique est réalisée avec le trajet optimal déterminé par l'algorithme Needleman-Wunsch sur la matrice de bas niveau. Le score final obtenu par la matrice de bas niveau est incorporé dans la matrice de haut-niveau.

- Enfin, en complétant au fur à mesure la matrice de haut niveau avec les scores finaux des matrices de bas niveaux une seconde programmation dynamique (algorithme NW) est effectuée pour connaître le trajet optimal d'une séquence par rapport à la structure de référence.



A B C

Figure 1 : 3 schémas a, b, c représentant la suite algorithmique de l'enfilage de séquences protéiques par double programmation dynamique.

A) Le premier schéma A, nous montre la sélection d'un acide aminé dans la séquence pour une position donnée qui est déterminée par la matrice de haut niveau. Par exemple, ici il est choisi de prendre l'acide aminé D pour la position 6. Cette sélection va aboutir à la création de la matrice de bas niveau (schéma B) en prenant cette position est en comparent la fonction de score de l'énergie statistique entre tous les autres acides aminés de la structure.

B) Le second schéma B est donc une matrice de bas niveau qui possèdent pour chaque position de la matrice une valeur associée au potentiel statistique. Le chemin optimal nous permet d'obtenir la valeur de score à incorporer dans la matrice de haut niveau (schéma C)

C) Le schéma C, nous montre le positionnement d'une valeur de score obtenue par la matrice de bas niveau dans la matrice de haut niveau. Le chemin optimal nous donne un score final qui est le score d'adéquation.

# 3. Matériels et méthodes

- Matériels

Les fichiers suivants ont été utilisés :

- Le fichier « dope.par » qui contient l'ensemble des valeurs des potentiels statistiques avec pour chaque couple d'acides aminés 30 valeurs espacées par une distance de 0.25 Angströms.

- Les fichiers .fasta et .pdb qui ont le code de référence «  2XRI » et qui ont été téléchargés sur le site de « protein data bank ». Ils servent de support pour la séquence primaire de la protéine d'intérêt et aussi de référence au niveau de la structure tertiaire.

Enfin, un environnement « conda » nommé « conda_threader » a été créé pour faciliter la reproductibilité des données. L'utilisation de conda permet de virtualiser un environnement et de télécharger toutes les dépendances de certaines librairies de python comme le package numpy et pandas ou encore Python 3.7 associés à ce projet.

- Méthodes

La séquence à suivre est la suivante :

- Ouvrir un terminal sous un environnement Linux, se positionner dans la racine du dossier et se déplacer dans le fichier source appelé « src » avec la commande :

```
$ cd src/
```

- puis activer l'environnement conda avec la commande suivante :

```
$ conda env create -f conda_threader.yml
$ conda activate conda_threader
```

- lancer ensuite le programme avec l'interpréteur Python et les paramètres suivants :

```
$ python3 main_threader.py ../data/pdb/2xri.pdb ../data/dope/dope.par
```

# 4. Résultats

Différents résultats sont générés par le programme python. Les premiers résultats sont l'affichage des matrices de bas niveau pour chaque couple d'acides aminés associés à une position. La matrices de haut niveau devrait ainsi afficher la suite des scores optimaux obtenues. Le meilleur score obtenu aussi nommé score d'adéquation de la matrice de haut niveau est une indication pour savoir si la séquence possède une similitude avec la structure. Plus ce score d'adéquation est basse plus elle est proche de la structure. En effet, car plus l'énergie associée est basse plus c'est stable.

## 5. Discussion

Au niveau du temps de réalisation, l'algorithme de double programmation est un processus très long. Pour les paires d'acides aminés possibles une complexité globale de l'algorithme en $O(N^4)$. Le problème devient important lorsqu'il faut comparer une longue séquence (plus de > 500 résidus) avec une structure de taille similaire il faudrait environ 12h de calcul. Les fonctions de gap ne sont pas prises en compte dans l'alignement des séquences. De plus une seule structure est prise en compte dans le programme, néanmoins il est important d'avoir un ensemble de structures (data set) pour optimiser l'alignement d'une séquence qui n'a pas de structure tertiaire déterminée.

## 6. Annexes

L'ensemble du programme n'a pas été réussi. En effet, la matrice de bas niveau comporte des bugs. Ces erreurs viennent du fait que j'ai réalisé des dataframes pour une matrice de bas niveau et que lorsque je rajoute un élément avec une colonne possédant le même nom alors il ajoutera cette élément sur toutes les colonnes avec ce même nom pour une position donnée. Ayant repéré et compris cette erreur trop tardivement, je n'ai pas été en mesure de changer la conception du remplissage des données. On pourrait penser autrement en utilisant les matrices avec la librairie numpy. Enfin, l'architecture du programme est la suivante :

```
├── conda_threader.yml
├── data
│   ├── 2019-09-10
│   │   ├── 2xri.fasta
│   │   ├── 2xri.fasta.txt
│   │   └── 2xri.pdb
│   └── 2019-13-10
│       └── dope.par
├── doc
├── README.md
├── results
└── src
    └── main_threader.py
```

## Références


- Jones, D. (1998) Chapter 13 THREADER: protein sequence threading by double dynamic programming. pp. 285-311.

- Fichier dope : http://www.dsimb.inserm.fr/~gelly/data/dope.par