

Projet DSSP

Jeyani George Clement
M2 BI Université Paris Cité

14 septembre 2022

Table des matières

1	Introduction	2
2	Matériel et méthodes	2
2.1	Matériel utilisé	2
2.2	HBPLUS	3
2.3	Scripts python	3
2.3.1	Script de parsing d'un fichier au format hb2	3
2.3.2	Script pour l'assignation des hélices	3
2.3.3	Script pour l'assignation des feuillets- β	4
3	Résultats	4
4	Conclusion et discussion	5
5	Bibliographie	5
6	Annexes	6
6.1	Sortie de DSSP pour la protéine 3ZY0	6
6.2	Difficultés rencontrées	6

1 Introduction

Au cours de ces dernières décennies, la compréhension de la relation existant entre la séquence primaire d'une protéine, constituée d'une succession d'acide aminé et de sa structure secondaire fût au centre de nombreuses recherches. Ainsi, dans l'article de Kabsch et al. [2], ils ont mis en place une succession de critères basés sur la reconnaissance de motifs issus de la formation de liaisons hydrogènes entre des résidus spécifiques d'une séquence protéique donnée. Au sein de leurs article, d'autres critères de sélections ont été mis en avant comme la prise en considération de l'énergie de ces liaisons mais également la géométrie de la protéine. Ces règles constituent une simplification de l'algorithme de DSSP (Define Secondary Structure of Proteins). Cet algorithme permet d'assigner les structures secondaires d'une protéine donnée en se basant sur les coordonnées x, y et z de chaque résidu composant cette protéine.

Dans le cadre du projet court, nous avons reçu un sujet portant sur l'implémentation des critères issus de l'article de Kabsch et al. [2] afin de pouvoir assigner à notre tour la structure secondaire associée à chaque résidu d'une séquence protéique. Pour arriver à cette assignation, nous nous concentrons uniquement sur la reconnaissance de motifs issus de liaison hydrogène. Chaque critère correspond à la détection d'un motif de liaison hydrogène qui est spécifique à une structure secondaire. L'algorithme DSSP permet de déterminer au total 8 types de structures secondaires au sein d'une protéine. Pour simplifier notre projet, nous avons réduit ce nombre afin de pouvoir détecter uniquement les hélices et les feuillets- β .

Il est important de noter que ces deux types de structures sont des structures complexes qui résultent de la succession de structures moins complexes. En outre, pour définir une hélice, nous devons tout d'abord définir une structure moins complexe qui est le *n-turn*. La succession de deux n-turns permettra ensuite de définir une hélice. De même, pour définir un feuillet- β nous devons d'abord définir un *bridge* qui sera soit parallèle soit antiparallèle. La succession d'un de ces deux types de bridge permettra de définir une autre structure qui est un *ladder* dont la succession permettra finalement de définir un feuillet.

Pour mener à bien ce projet, nous avons implémenté des scripts python afin de permettre de parser des fichiers au format .hb2 et d'assignation des hélices et des feuillets- β pour notre protéine d'intérêt.

2 Matériel et méthodes

2.1 Matériel utilisé

Dans le cadre de notre projet, nous devons assigner les structures secondaires présentes au sein d'une protéine donnée. Nous avons ainsi commencé par aller sur le site de la PDB, afin d'obtenir notre protéine d'intérêt au format .pdb. La protéine dont le code PDB est : 3ZY0 a été choisi. Cette protéine correspond à la protéine p63, une protéine tumorale présente chez l'Homme. Elle est constitué de quatre chaînes protéiques : A, B, C et D. Chaque chaîne contient 32 résidus. La structure que nous avons récupérée est issu d'une cristallographie par rayons X. Contrairement aux structures déterminer par NMR, celles issues de cristallographie ne contiennent pas d'hydrogène car c'est un atome qui est trop petit pour être détecté par les rayons X. Néanmoins, comme expliqué précédemment, nous nous basons uniquement sur la position des liaisons hydrogènes afin de déterminer une structure secondaire. Nous avons donc dû chercher un logiciel qui nous permettra de définir la position relative de ces liaisons hydrogène. Ainsi, nous avons utilisé le logiciel HBPLUS [1].

2.2 HBPLUS

Le logiciel HBPLUS (version 3.06) permet de calculer les liaisons hydrogène présentes au sein d'une protéine donnée. Pour ce faire, le logiciel récupère les coordonnées atomiques de chaque résidu et calcul la distance séparant ces derniers. Nous supposons que si cette distance est inférieure à 3.0 Å, alors ces deux résidus sont liés par une liaison hydrogène. Il créera ensuite un fichier au format .hb2 contenant un en-tête et un corps dont chaque ligne correspond à un couple de résidus (avec leurs noms de résidus (code 3 lettre), leurs position et la chaîne auxquelles elles sont associées)) impliqué dans une liaison hydrogène. C'est ce fichier que nous utiliserons pour effectuer nos assignations.

2.3 Scripts python

L'ensemble des scripts qui ont été rédigé utilise le langage de programmation python (version 3.8.12) ainsi que les modules suivants :

- module *re* : permet d'effectuer des opérations sur des expressions régulières.

2.3.1 Script de parsing d'un fichier au format hb2

Pour nos analyses nous avons écrit un script *hb2_parsing.py* nous permettant de parser ce fichier au format .hb2. Pour ce faire, deux fonctions ont été implémenté :

1. **select_res_interacting()** : fonction récupérant en entrée le nom du fichier à parser sous forme de chaîne de caractères. Il va ensuite parcourir ce fichier et stocker au sein d'une liste les lignes du fichier ne comportant que les couples de résidus impliqués dans une liaison hydrogène. Il retourne en sortie cette dite liste. Cette fonction permet une première étape de nettoyage du fichier.
2. **create_dico_hbonds()** : fonction prenant en entrée la liste issue de la fonction précédente. Il parcourt cette liste afin de récupérer les informations suivantes : le nom de chaque couple de résidus impliqués dans une liaison ainsi que leurs positions respectives (par le biais de *regex*). Ces informations seront stockées au sein d'un dictionnaire dans lequel chaque clé représente une chaîne de la protéine. Les informations sur ces couples de résidus seront ensuite stockés dans une liste de sous-liste dans chaque clé. La fonction ensuite ce dictionnaire.

L'utilisation de *regex* constitue une méthode de filtrage puissant et rapide pour généraliser des cas. Par exemple, le nom d'une chaîne protéique peut être A, a, A1 ou encore AA. Les *regex* nous permettent de prendre ses différentes conventions d'écriture en considération sans alourdir le code. De même, nous avons fait le choix de stocker cette information au sein d'un dictionnaire car cela permet de ranger des informations d'une manière spécifique. Dans le cas présent, les couples de résidus impliqués dans des liaisons hydrogènes sont rangés en fonction des chaînes présentes dans la protéine.

2.3.2 Script pour l'assignation des hélices

À présent que les informations nécessaires ont pu être extraites, nous pouvons commencer l'assignation des structures secondaires. Pour commencer, nous souhaitons assigner les hélices (pour simplifier nous n'avons pas fait de distinction entre les différents types d'hélices qui existent). Comme nous le spécifions dans l'introduction, une hélice est définie par la succession au minimum de deux *n-turns*. Pour trouver des *n-turns* nous nous basons sur le postulat suivant : "un *n-turn* est défini dans le cas où il existe une liaison hydrogène entre le CO d'un résidu *i* et le NH d'un résidu à la position *i+3*, 4 ou 5".

Ainsi, nous avons écrit le script suivant : *helix_assignment.py*, nous permettant cette assignation par le biais des deux fonctions suivantes :

1. **select_turn_res()** : cette fonction prend en entrée le dictionnaire généré par le parsing du fichier .hb2. De ce dictionnaire, la fonction va extraire les positions des résidus i effectuant des liaisons hydrogènes avec des résidus $i+3$, 4 ou 5. Ces positions de résidus seront stockées dans une liste qui sera stockée au sein d'un dictionnaire. La sortie de cette fonction sera donc un dictionnaire contenant pour chaque chaîne une liste de position de résidus.
2. **assign_helix()** : cette fonction prend en entrée le dictionnaire généré par la fonction précédente. Il va ensuite assigner pour chacun des résidus présents dans ce dictionnaire, la structure secondaire adéquate. Effectivement, dans le cas où deux n-turns successifs (ou plus) sont détectés, la lettre "H" sera assignée aux résidus concernés. De la même manière, s'il n'y a pas succession de n-turn alors la lettre "T" sera associée à ce résidu. La fonction retourne en sortie une liste contenant l'assignation des structures secondaires par résidu. S'il existe un gap entre les positions des résidus le caractère "-" sera assigné. Nous mettrons autant de "-" que de gap.

2.3.3 Script pour l'assignation des feuillets- β

La deuxième structure secondaire que nous voulons assigner correspond aux feuillets- β . Pour pouvoir assigner ces structures nous devons tout d'abord définir d'autres structures moins complexes. La première structure que nous devons définir correspond aux *bridges*. Ces derniers peuvent être parallèles ou antiparallèles et sont issus d'un motif de liaisons hydrogènes entre deux groupes de 3 résidus consécutifs (le groupe i et j) qui ne se chevauchent pas. Ces deux types de bridges suivent des lois spécifiques présent dans l'article. Dans le cas où des bridges ont été défini, nous considérons que la succession de deux bridges (parallèle ou antiparallèle) définissent une structure plus complexe qui est le *ladder*. Enfin, la succession de deux ladders permet d'affirmer que nous avons dans notre séquence un feuillet- β qui peut être parallèle ou antiparallèle.

Malheureusement, pour le moment, nous n'avons pas réussi à assigner cette structure. Les fonctions que nous allons vous présenter sont les pistes que nous avons mais qui ne sont néanmoins pas fonctionnelles. Un script *sheet_assignment.py* a donc été écrit pour tenter de trouver des bridges au sein de notre protéine. Ce script contient pour le moment deux fonctions qui sont les suivantes :

1. **find_sequential_res** : cette fonction prend en entrée le dictionnaire généré par le parsing du fichier .hb2. Elle va récupérer les par groupe de 3, les résidus dont les positions sont successives. Par la suite elle stockera cela dans un dictionnaire dont les clés correspondent aux chaînes de la protéine. Chaque clé contiendra une liste de sous-liste contenant 3 éléments correspondant aux positions de ces résidus. Elle retournera ensuite ce dictionnaire.
2. **assign_antiparallel_bridge** : fonction dont l'implémentation n'est pas encore fonctionnelle mais dont le but est de définir un bridge antiparallèle au sein d'une séquence protéique.

3 Résultats

À partir de nos différents scripts, nous avons pu implémenter un outil permettant d'assigner uniquement des hélices. Nous avons testé nos scripts avec la protéine p63 dont le code PDB

est 3ZY0. Voici ci-dessous ce que nous obtenons pour chacune des chaînes composant cette protéine :



FIGURE 1 – (a) Visualisation de la protéine 3ZY0 représenté en *cartoon* et coloré en fonction des chaînes. (b) Visualisation de l’output de notre script.

Nous pouvons observer ici que notre script arrive à détecter sur chaque chaîne la présence d’hélices qui peut être visualisée sur l’image de la protéine. Notre assignation semble donc cohérente avec ce qui est montré ci-dessus.

4 Conclusion et discussion

En conclusion, au cours de ce projet nous avons réussi à implémenter un outil permettant l’assignation des hélices au sein d’une séquence protéique. Cette assignation a été basé sur la détection de motif de liaison hydrogène au sein d’un fichier au format .hb2. Néanmoins, par manque de temps et de compréhension du sujet, nous n’avons pas réussi à assigner les structures nous permettant d’obtenir des feuillets- β . Ces structures incluent les bridges mais aussi les ladders.

Avec un peu plus de temps et une meilleure compréhension de comment nous pourrions implémenter cela, nous aurions sans doute pu implémenter cette assignation. De plus, il serait aussi intéressant de rajouter à nos résultats un alignement de notre séquence d’acide aminé avec la séquence d’assignation des hélices pour savoir à quelle position ces hélices sont localisées. Il serait aussi intéressant de complexifier nos scripts afin de faire la distinction entre les différents types d’hélices qui existent. Enfin nous pourrions également comparer nos résultats avec ceux de DSSP pour avoir une idée de ce que nous aurions dû obtenir.

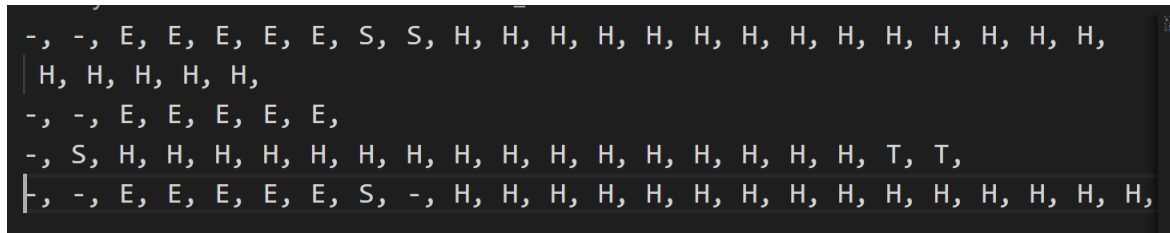
5 Bibliographie

Références

- [1] HBPLUS home page. Consulté le 10 septembre 2022. Visible à l’adresse [suivante](#).
- [2] Kabsch W, Sander C. Dictionary of protein secondary structure : pattern recognition of hydrogen-bonded and geometrical features. Biopolymers. 1983 ;22(12) :2577-2637. Consulté le 10 septembre 2022. Visible à l’adresse [suivante](#).

6 Annexes

6.1 Sortie de DSSP pour la protéine 3ZY0



```
- , - , E , E , E , E , E , S , S , H , H , H , H , H , H , H , H , H , H , H , H , H , H ,  
H , H , H , H , H ,  
- , - , E , E , E , E , E ,  
- , S , H , H , H , H , H , H , H , H , H , H , H , H , H , H , H , T , T ,  
- , - , E , E , E , E , E , S , - , H , H , H , H , H , H , H , H , H , H , H , H , H , H ,
```

FIGURE 2 – Sortie du logiciel DSSP avec la protéine 3ZY0

6.2 Difficultés rencontrées

Lors de ce projet, j'ai dû faire face à différentes difficultés :

1. Mauvaise compréhension de la définition d'un feuillet- β .
2. Perte de temps lors de la recherche du bon logiciel à utiliser pour calculer les liaisons hydrogène. Manque de temps pour tout implémenter.