SORBONNE UNIVERSITE FACULTE DES SCIENCES
UFR D'INFORMATIQUE LICENCE 2

PROJET DE DÉVELOPPEMENT

Sujet 10 : Identification de motifs structuraux dans les protéines

Réalisé par :
Louiza AOUAOUCHE 3802084
Ines YAICI 3802715
Han ZHANG 3970759

Encadrées par : Mme Mathilde CARPENTIER

DM-IM 2019-2020

Table des matières

1 INTRODUCTION

Ces dernières décennies, le monde a connu une évolution explosive grâce à l'utilisation applicative des méthodes informatiques sur différents secteurs dont la biologie. Ainsi en découle le domaine de la bio-informatique qui utilise la rapidité et l'exactitude de la machine au profit de l'étude des phénomènes biologiques.Comprendre le fonctionnement des organismes c'est aussi comprendre ce qui les compose. Ainsi notre projet consiste à développer des algorithmes et les programmer avec la meilleure complexité possible afin d'identifier au mieux les similarités structurales dans des séquences de protéines.Pour cela, nous nous sommes basées sur l'algorithme de Karp-Miller-Rosenberg qui détecte les répétitions dans une structure de données. En premier lieu, nous avons élaboré un algorithme en une seule dimension qui permet d'étudier la composition de la structure protéique en retrouvant les répétitions de motifs dans un ou plusieurs génomes composé(s) d'acides aminés où chacun d'eux est identifié par une lettre. En second lieu, nous sommes passés en deux dimensions où nous retrouvons les similarités structurales d'une chaîne protéique selon les positions dans l'espace des acides aminés (identifiés par leurs Carbone α). Les algorithmes ainsi développés ont fait naître des programmes (codés en Python) sur lesquels nous avons effectué une étude expérimentale en terme de complexité et de temps d'exécution.

2 ALGORITHME

L'étude des similarités des structures protéiques nous a amené à considérer des séquences de protéines en 1 dimension (algorithme décrit en annexe) mais nous nous sommes principalement basées sur l'algorithme en 2 dimensions suivant (avec exemple d'exécution) : En dimension 2 on va principalement étudier la forme des protéines dans l'espace et trouver des similarités dans ces formes, alors que dans l'algorithme en dimension 1 on étudiait la constitution de ces chaînes d'acides aminées, on peut remarquer donc que ces deux algorithmes se complètent.

- *On extrait les informations qui nous intéressent sur la protéine à étudier à partir des lignes des fichiers PDB qui se présentent comme ceci :*

					x	y	z			
ATOM	3346	C	GLY	A 447	4.837	38.599	-13.394	1.00	67.40	C
ATOM	3350	C	SER	A 448	4.945	41.832	-13.138	1.00	63.68	C
ATOM	3356	C	LYS	A 449	6.722	41.522	-9.999	1.00	61.50	C
ATOM	3365	C	PRO	A 450	9.198	40.883	-8.121	1.00	62.64	C

- *On lit les positions (x,y,z) des carbones α de chaque acide aminé constituant la protéine.*
- *Puis à partir de ces positions on va calculer la distances entre chaque acide aminé et on obtiendra une matrice de distance.*

i /j	...	*5*	*6*	*7*	*8*
0		*1.75*	*2.2*	*2.00*	*1.64*
1		*3.84*	*1*	*2.18*	*2.02*
2		*4.5*	*2.6*	*2.98*	*1.99*
3		*2*	*2.3*	*2*	*2.75*

(Les indices représentent les numéros des carbones alpha comme parcourus dans le fichier PDB)

- *On va discrétiser cette matrice pour la manipuler plus facilement*

i /j	...	*5*	*6*	*7*	*8*
0		*1*	*2*	*2*	*1*
1		*3*	*1*	*2*	*2*
2		*4*	*2*	*2*	*1*
3		*2*	*2*	*2*	*2*

- *Cette matrice constitue notre premier vecteur V, qui sera une matrice carrée symétrique de dimension le nombre d'acides aminés.*
- *Cette matrice constitue notre premier vecteur V, qui sera une matrice carrée symétrique de dimension le nombre d'acides aminés.*

1	*[(0.5) ,~~(0.8)~~, (1.6) ,~~(2.8)~~]*
2	*[(0.0) , (0.7), (1.7), ~~(1.8)~~, (2.6) ,(2.7), (3.5), (3,6), (3.7),~~(3.8)~~]*
3	*[(1.5)]*
4	*[(4.5)]*

- *On dépile le vecteur P et on lit à chaque tour sur le vecteur V une fois à droite une fois en bas de la case d'indice les coordonnées qu'on dépile, et là on aura à faire à des cas particuliers*
  - *S'il n'y a rien à droite de la case (si on est sur la bordure droite ou bien il y'a un -1 dans la case $(i+1, j)$) quand on regarde à droite, elle ne va plus nous intéresser et de même quand on regarde en bas.*

- *En suivant cela on arrive à créer le vecteur Q, il a principalement la même dimension que le vecteur P (on commence la lecture en regardant à droite)*

1	[[(2.7) , (0.7)] , ~~[(1.5)]~~]
2	[[(1.6) , (1.5)] , [(3.7) ,(3.6) ,(3.5) ,(2.6) , (1.7),(0.6)] , ~~[(2.5)]~~]
3	
4	

- *On va faire correspondre à chaque sous liste un numéro tel que :*
  - *Exemple : [(2.7), (0.7)] :1, [(1.6), (1.5)] : 3, [(3.7) ,(3.6),(3.5),(2.6) ,(1.7),(0.6)] :2 . . .*
  - *On ne prend pas en considération les sous listes à un seul élément.*
- *On obtient un nouveau vecteur V :*

i /j	...	5	6	7	8
0		3	2	2	-1
1		-1	3	2	-1
2		-1	2	1	-1
3		2	2	2	-1

- *On reconstitue un vecteur P (de dimension le nombre de sous listes dans le vecteur Q). Puis on recrée encore une fois un vecteur Q en éliminant les indices des cases sur le bord du bas, ou celles où il y'a un -1 en dessous.*

VP :

1	~~[(0.7) , (2.7)]~~
2	[(0.6), (1.7), (2.6) , ~~(3.5), (3.6) ,(3.7)~~]
3	[~~(0.5)~~ , (1.6)]

VQ :

1	[~~[(1.7)]~~]
2	[[(2.7) ,(0.7)] , ~~[(2.6)]~~]
3	[~~[(0.6)]~~ , ~~[(1.6)]~~]

- *On s'arrête ici pour ce petit exemple, il nous reste qu'une seule sous liste de couple [(2.7),(0.7)] , on a fait une opération à droite et une autre en bas, en revenant au tout premier vecteur V on va voir en ces positions des similitudes de formes de dimension 2*2 :*

i /j	...	5	6	7	8
0		1	2	2	1
1		3	1	2	2
2		4	2	2	1
3		2	2	2	2

- *Et c'est comme ceci qu'on va trouver des similitudes dans la forme d'une protéine.*

3 PROGRAMMATION/IMPLEMENTATION

Avec ce rapport nous avons fourni un dossier comportant différents fichiers qui représentent notre code, une documentation pour le code (où il y'a une description des fonctions implmentées) ainsi que des jeux de tests.

- *dans le fichier "dim1.py" se trouvent toutes les fonctions utiles à l'algorithme en dimension 1 .*
- *dans le fichier "dim2.py" se trouvent toutes les fonctions utiles à l'algorithme en dimension 2 .*
- *dans le fichier "main.py" on va trouver les fonctions de lecture de fichiers de type ".fasta" où on trouve la chaine de protéines (les successions d'acides aminés) qu'on va lire pour l'algorithme en dim 1 , ainsi que les fichier ".pdb" qui sont plus complexes qui contiennet tous les atomes , leurs positions ... qu'on va utiliser pour la dim2. Comme on pourrait le voir d'après son nom ce fichier contient la fonction main() et donc des instructions pour les tests.*
- *on a fourni un jeu de test qui va tester la dimension 1 ainsi que la 2, il suffit de se placer dans le terminal au niveau du dossier du code et de lancer la commande " bash ./Test.bash <PDB-ID> 1/0" (mettre 1 pour avoir l'affichage et 0 sinon).*
- *pour faciliter les choses on a mis dans le dossier ou il y'a le code des fichiers spécifiques aux protéines 4CR2 et 1BG8, pour lancer le jeu de tests entrez juste :*
  - *"bash ./Test.bash 4CR2 1"*
  - *"bash ./Test.bash 1BG8 1"*

  les résultats et l'affichage suiveront (cela peut prendre jusqu'à 10min soyez patients).

4 ANALYSE ET STATISTIQUES

4.1 ETUDE THÉORIQUE

D'après Wikipédia : "Les protéines sont formées d'une ou plusieurs chaînes polypeptidiques, qui sont des biopolymères linéaires, pouvant être très longs, composés d'une vingtaine d'acides L-α-aminés différents."

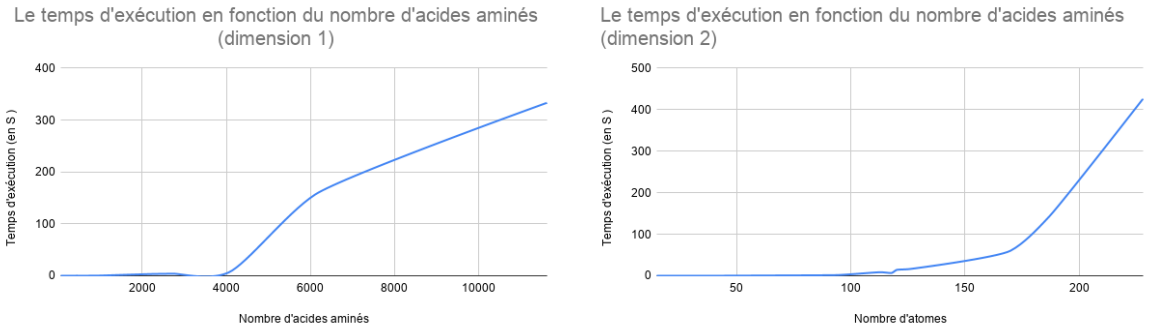
Pour la dimension 1, on considère 4 fonctions principales qui influencent la complexité dans comparerSequences : creerTabBase, creervp, creervq, creervv. On note N le nombre de chaînes polypeptidiques, et L la longueur de la chaîne la plus grande.

- *Pour creerTabBase, on parcourt la liste de séquences, il nous faut $O(N^2L^2)$.*
- *Pour creervp, on parcourt le vecteur v, il nous faut $O(NL)$.*
- *Pour creervq, on parcourt principalement le vp, il nous faut $O(NL)$.*
- *Parce qu'on va répéter cette procédure pour un nombre de tours qui ne dépassent pas la taille du motif. On otient que la complexité en temps est au maximum en $O(N^2L^2)$.*

Pour la dimension 2, on considère 4 fonctions principales : creer_vv_de_base, creervp_2d, creervq_2d, creervv_dim2. On note n le nombre d'atomes.

- *Pour creer_vv_de_base, c'est une fonction qui parcourt la matrice de distance de taille $n \times n$.*
  D'abord, en parcourant la matrice la première fois, on fait agrandir le vecteur vv et on définit l'ensemble ens, ce qui nous fait un $O(n^2)$.
  Ensuite, on développe le dictionnaire à partir de l'ensemble,il nous faut au plus $O(n^2)$, et au moins $\theta(len(ens))$.
  En fin, on parcourt la matrice la deuxième fois. Et on remplace la distance dans vecteur V par la valeur associée selon le dictionnaire, ce qui nous faut $O(n^2)$.
  On peut conclure que la fonction creer_vv_de_base est en $(O(n^2))$.
- *Pour creervp_2d, on crée d'abord une liste vide, et après on parcourt le vecteur V qui est on le rappelle une matrice de dimension $n \times n$, ce qui nous donne $O(n^2)$.*
- *Pour creervq_2d, on parcourt le vecteur P et on justifie la position en bas ou a droite, ce qui nous fait dans le pire cas un $O(n^2)$, qui est actuellement le nombre de cases du vecteur P. A la fin, on ajoute ces positions dans le vecteur Q. La fonction creervq_2d est donc en $O(n^2)$.*
- *Parce qu'on va répéter cette procédure pour un nombre de tours qui est dans le pire des cas égal à la taille du motif. Donc la complexité du temps pour l'algorithme en dimension 2 est dans le pire des cas en $O(n^3)$ qui est aussi une complexité polynomiale.*

4.2 ETUDE EXPÉRIMENTALE



(a) Dimension 1

(b) Dimension 2

FIGURE 1 – Statistiques en comparaison

Nous prenons des exemples sur le site de RCSB (référencé dans la bibliographie) .
Pour la dimension 1, on a sélectionné des protéines ayant une longueur de séquences (qui est aussi le nombre des acides aminées) qui varie de 61 jusqu'à 11637 acides aminés, comme on le voit l'algorithme en dimension

1 nous permet d'analyser des protéines avec de très longues séquences et le graphe nous permet de voir que la compléxité reste polynomiale.

On voit que plus la longueur augmente plus le temps d'exécution augmente, et que plus le nombre des atomes augmente plus le temps d'exécution augmente aussi.

Pour la dimension 2, on a sélectionné des protéines ayant un nombre d'atomes(carbones alpha) qui va de 15 jusqu'à 228,pour l'algorithme en dimension 2 est moins performant niveau temps par rapport à l'algorithme en dimension une,à partir d'une séquence de 200 acides aminés (chaque acide aminé à un seul carbone alpha) l'attente commence à être de 5-6 min.

De plus, on voit en dimension 2, on prends plus de temps a chaque étape. Le temps d'exécution en dimension 2 augmente plus vite qu'en dimension 1,néanmoins la compléxité reste polynomiale.

On a essayé d'étudier de plus près cette question de temps d'éxécution, en modifiant quelques parties de notre algorithme (par exemple en utilisant la symétrie de la matrice de distance), mais on se retrouvait avec un temps d'éxécution plus élevé ou similaire à cause des contraintes du langage qui déplace la compléxité ailleurs.

On prend la protéine 4CR2 par exemple : la longueur de séquence de 4CR2 est 11637. Quand on applique notre algorithme en dimension 1 a 4CR2, il nous prend 333 secondes pour exécuter.

Cependant, en dimension 2, on prend la protéine 1BG8 ayant nombre d'atomes égal à seulement 228. Nous avons besoin de 426 secondes pour l'exécuter.

4.3 CONCLUSIONS ET COMPARAISONS

En conclusion, l'algorithme en dimension 1 est plus efficace qu'en dimension 2 quand on examine les protéines ayant un grand nombre de carbones alpha et des chaînes répétées.

En comparaison avec la compléxité de l'algorithme Karp-Miller-Rosenberg sité dans la section bibliographie ;

> On the other hand, the Karp–Miller–Rosenberg's algorithm (KMR) [5] is available to detect *a priori* unknown exactly repeated patterns in one given string, with time complexity $O(n \log_2 n)$, where n is the length of the input string and O the classical notation as described by Knuth [4]. Similarly, Martinez [6] describes a sorting algorithm, with a time complexity $O(n \log_2 n)$, to display *a priori* unknown identically repeated patterns in several molecular sequences.

On voit dans l'algorithme original la compléxité pour la dimension 1 est en $O(nlogn)$,où n est la taille de la chaîne de caractères, notre adaptation de cet algorithme à une compléxité de $O(n^2)$ qui n'est pas similaire mais pas très éloignée non plus.

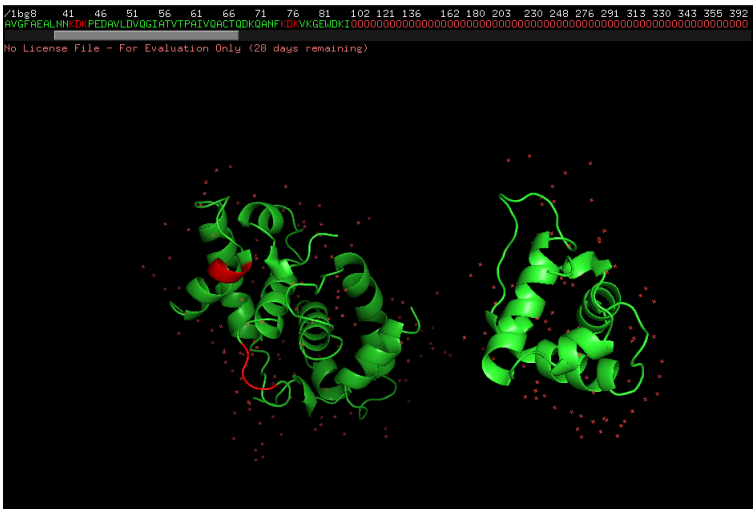
5 ILLUSTRATION DES RESULTATS

Ici nous allons décrire d'une certaine manière l'approche qu'on a pris par rapport à notre code et nos résultats.

- *l'éxécution de la dimension 1 pour la protéine 1BG8 :*

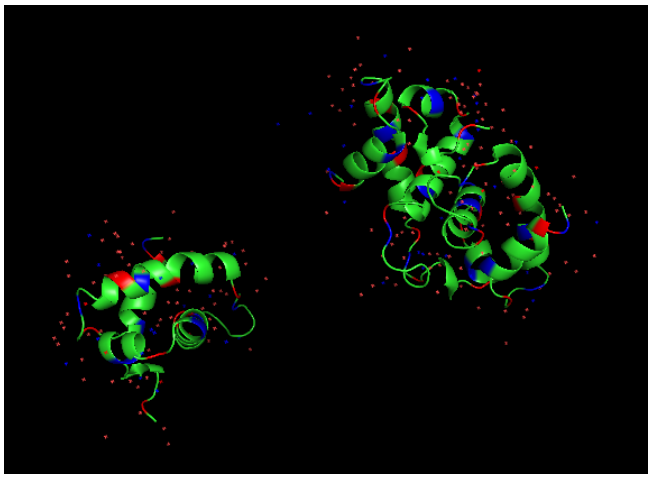
```
TAILLE DU MOTIF: 3
----------------------

MOTIF->POSITION(S)
----------------------
KDK -> [41, 74]
```

- *le résultat observé avec Pymol (qui est un logiciel d'observation de structures chimiques en 3D) :*



on voit donc sur la protéine en rouge les successions de taille 3 de motifs similaires (ils ne sont pas de la même forme, mais c'est les mêmes acides aminés).

- *pour la dimension 2,le programme nous donne directement l'instruction pymol à éxecuter pour identifier les endroits geométriquement similaires :*



tels que les motifs coloriés en rouge correspondent géometriquement à un motif colorié en bleu.

6 CONCLUSION

Ce projet à été une application de nos connaissances en informatique dans un domaine (biologie) ou on n'est pas forcément à l'aise, et on se rend compte qu'en tant que future informaticiens on va être amené à travailler dans des domaines variés d'où l'importance d'apprendre à s'adapter.

on a essayé de développer un algorithme en vérifiant à chaque fois les complexités pour éviter qu'elles s'éloignent de l'algorithme de base, ce qui n'était pas une tâche facile, mettre un test dans une fonction ou dans une autre peut changer beaucoup de choses, on a compris qu'avant de se lancer dans quelque chose d'aussi complexe, il fallait vraiment étudier toutes les possibilités pour avoir quelque chose d'efficace et de rapide.

On peut dire qu'on a apris beaucoup de choses sur les protéines et les acides aminés,on a ainsi utilisé le logiciel Pymol pour s'approfondir graphiquement et voir nos résultats réellement ; on s'est bien approprié le sujet et ça à été très instructif et intéressant.

Pour finir nous remerçions de façon générale toute les personnes qui nous ont aidé à réaliser ce projet et particulièrement notre encadrante Mme Mathilde CARPENTIER qui nous a guidé,conseillé et encadré depuis le début malgré les circonstances de travail assez délicates.

7 ANNEXE

7.1 ALGORITHME DIMENSION 1

Pour illustrer l'algorithme 1D, considérons la séquence suivante : ***ACFIJACF***
Ici, nous recherchons des similarités dans une seule séquence mais "algorithme est valable pour plusieurs séquences différentes.

- *On identifie l'alphabet associé aux acides aminés et on affecte à chaque lettre un indice :*

- *Initialisation du Vecteur V de taille égale à la taille de la séquence. Chaque indice i du Vecteur V correspond à la i-ème lettre de la séquence*

ALPHABET :

A	*C*	*F*	*I*	*J*
0	*2*	*5*	*8*	*9*

i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Séquence	*A*	*C*	*F*	*I*	*J*	*A*	*C*	*F*
Vecteur V	*0*	*2*	*5*	*8*	*9*	*0*	*2*	*5*

- *Tant qu'on n'atteint pas la taille voulue (maximale) ou que la chaîne n'est pas totalement parcourue on effectue les instructions suivantes :*

  – *Construction du Vecteur P :un tableau de piles où dans chaque pile d'indice p_i on stocke les indices i des lettres de la séquence tel que Vecteur $V[i]$=p_i*

VP :

5	*6*	*7*		
0	*1*	*2*	*3*	*4*
0	*2*	*5*	*8*	*9*

  – *Construction du Vecteur Q : un tableau de piles. On dépile le Vecteur P, supposons que nous ayons dépilé l'élément i.On va alors à la position $i+k$ (k étant un "pas") dans le Vecteur V (si c'est possible) et on y prend la valeur notant la x. Ainsi on empile dans la pile Vecteur Q [x] la valeur Vecteur $V[i+k]$. Lorsqu'on passe de la pile p_i à la pile p_i+1 dans le Vecteur P, on empile un séparateur (/) sur toutes les piles du Vecteur Q.Pour k=1, on obtient le VQ suivant ainsi que les nouveaux groupes :*

VQ :

0	/	/	/	/	/	*4*	/
2	/ *5*	*0*	/	/	/	/	/
5	/	/ *6*	*1*	/	/	/	/
8	/	/	/	*2*	/	/	/
9	/	/	/	/	*3*	/	/

GROUPES :

0	*1*	*2*	*3*	*4*

  – *Ainsi on forme à nouveau le Vecteur V :*

i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Séquence	*A*	*C*	*F*	*I*	*J*	*A*	*C*	*F*
Vecteur V	*1*	*2*	*4*	*3*	*0*	*1*	*2*	-

  – *A l'issue de ce premier tour on connaît deux motifs de taille 2 aux positions $(0,5)->AC$ et $(1,6)->CF$*

- *Avec ce Vecteur V on refait les instructions précédentes et on obtient à la fin une répétition du motif 'ACF' aux positions $(0,4)$:* ***ACFIJACF***

8 BIBLIOGRAPHIE

- *Ce premier lien nous a servi à trouver des fichiers pdb et des structures de protéines et à récupérer les fichiers pour les étudier.*
  https://www.rcsb.org/

- *pour nos algorithmes on s'est principalement inspiré de l'algorithme KMR à partir de la thèse de notre encadrante (Mathilde Carpentier) et d'un atricle :*

  " An Algorithm for Finding a Common Structure Shared bu a Family of Strings "

  écrit par : ANNE M. LANDRAUD, JEAN-FRANCOIS AVRIL et PHILIPPE CHRETIENNE.

- *Nous avons aussi utilisé Wikipédia, pour quelques questions qu'on s'est posées sur les protéines et acides aminés.*