Projet: Recherche de structure dans les protéines

October 24, 2022

1 Consignes (important)

Les projets seront faits en binôme. Vous rendrez votre projet sur l'Université Virtuelle, dans l'espace prévu à cet effet, sous la forme d'un fichier zip qui sera intitulé:

NOM1_prénom1_NOM2_prénom2.zip.

Ce fichier zip qui contiendra à la fois le code et un rapport détaillé. Le code devra être fait en langage python version 3, et devra être commenté. Le rapport sera noté sur 10 et le code sur 10. La date limite du rendu final du projet est fixée au 04/12/2022. Chaque jour de retard entrainera une diminution de la note du projet de 2 points.

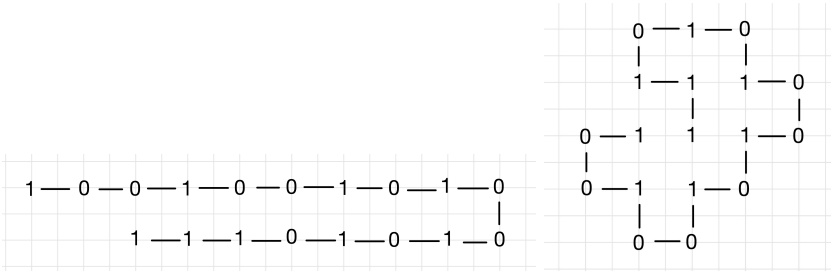
2 Définition du problème

Trouver la structure d'une protéine est un problème fondamentale en biologie. Dans ce projet, vous allez utiliser un modèle simple de protéines, qui groupe les acides aminés (AA) selon qu'ils sont hydrophobiques (n'aime pas l'eau) ou hydrophiles (aime l'eau). Les acides aminés hydrophobiques s'attire. Une protéine est alors simplement représentée par une séquence de 1 (pour hydrophobique) et de 0 (pour hydrophile). Le but est alors de trouver un agencement des 1 et 0 dans $\mathbb{N}^2$ qui maximise le nombre d'appariements (des 1 qui sont voisins dans l'espace choisi).

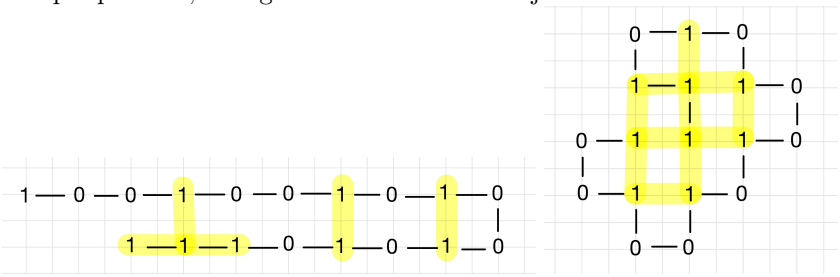
Avant de formaliser ce problème de manière mathématique, prenons un exemple. Considérons la séquece d'AA suivante

100100101001010111

Elle peut être plongée dans $\mathbb{N}^2$ de diverses manières. Par exemple:



Le *score* d'un plongement est donné par le nombre de paires de points voisins (horizontalement ou verticalement, mais pas en diagonal), qui sont étiquetés par 1. Par exemple, le score du plongement de gauche est 5 et celui de droite est 11. Graphiquement, il s'agit du nombre d'arêtes jaunes ci-dessous:



Nous pouvons maintenant donner une définition mathématique du problème. Un *mot* sur l'alphabet $\{0,1\}$ est une séquence d'éléments de $\{0,1\}$, par exemple, $p = 0100101$. Sa *longueur* $len(p)$ est le nombre de symboles qu'elle contient. L'ensemble des *positions* d'un mot p est défini comme l'ensemble $Pos(p) = \{1, 2, \ldots, len(p)\}$. Pour une position $i \in Pos(p)$, on note $p[i]$ le $i$ème symbole de p. Par exemple, pour $p = 0100101$, $p[1] = 0$ et $p[len(p)] = 1$.

Etant donné un mot p, un *plongement* de p dans $\mathbb{N}^2$ est une fonction **injective** $P : Pos(p) \to \mathbb{N}^2$ telle que pour toute position $i \in Pos(p)$ telle que $i < len(p)$, on a que $P(i)$ et $P(i+1)$ sont voisins horizontalement ou verticalement[1]. Le *score* d'un plongement P de p, noté $score(p, P)$ est défini par $score(p, P) =$

$$|\{\{i, j\} \mid i, j \in Pos(p), i \neq j, P(i) \text{ et } P(j) \text{ sont voisins}, p[i] = p[j] = 1\}|$$

Autrement dit, c'est le nombre de paires de positions différentes i, j de p, étiquetées par 1 dans p, et qui se plongent vers des points voisins dans $\mathbb{N}^2$.

On définit aussi le *meilleur score* d'un mot p comme le score maximal sur tous les plongements possibles. Il est définit par la fonction suivante:

$$bestScore(p) = max\{score(p, P) \mid P \text{ est un plongement de } p \text{ dans } \mathbb{N}^2\}$$

Le but du projet et d'utiliser un solveur SAT pour calculer la fonction $bestScore$. Un squelette de code est fourni, à vous de le compléter.

[1] Formellement, deux points $(i, j), (k, l) \in \mathbb{N}^2$ sont voisins si $(|i-k|, |j-l|) \in \{(0, 1), (1, 0)\}$.

3 Questions

Il vous est demandé de compléter le squelette de code fourni sur l'UV, tout en répondant aux questions suivantes:

Question 1: existence d'un plongement de score supérieur à une borne donnée Expliquez comment construire, pour tout mot p et toute borne $b \in \mathbb{N}$, un ensemble de clauses $S_{p,b}$ qui est satisfaisable si et seulement si $bestScore(p) \geq b$. Pour cette question, nous supposons l'existence d'une fonction *card* qui prend en entrée un ensemble fini de variables X et un entier k, et qui retourne un ensemble de clauses $card(X, k)$, qui est satisfaisable si et seulement si il existe au moins k variables de X qui sont vraies. Implémentez dans le squelette de code fourni cette transformation (fonction `exist_sol(seq,bound)`). Vous pouvez utiliser la classe `pysat.card` pour y utiliser une méthode qui calcule la fonction $card(X, k)$.

Vous devez expliquer dans le rapport comment vous construisez cet ensemble de clauses

Question 2: calcul du meilleur score Implémenter une recherche dichotomique pour calculer le meilleur score d'un mot p donné en entrée: fonction `compute_max_score` du squelette de code. *Expliquez brièvement dans le rapport comment vous procédez.*

Question 3: calcul du meilleur score Même question que précédemment mais avec une méthode incrémentale: vous testez d'abord avec la borne 0, puis 1, puis 2, etc. L'implémentation sera faite également dans la fonction `compute_max_score`, mais lorsque l'option `-i` en ligne de commande est activée.

Question 4: implémentation des options Implémenter les différentes options expliquées dans le squelette du code. Elles sont expliquées dans le code du squelette.

4 Informations importantes

Il vous est demandé d'utiliser la librairie PySAT disponible ici:

`https://pysathq.github.io/`

Vous aurez également besoin de la librairie `func_timeout`:

`https://pythonfix.com/pkg/f/func-timeout/`

pour pouvoir tester votre code. En effet, une fonction de test avec des exemples et des réponses attendues est implémentée. Elle appelle les fonctions que vous avez du implémenter plus haut. Si un appel prend plus que 10s, alors l'exécution est arrêtée, et un code d'exception est retourné. Vous pouvez changer la valeur TIMEOUT pour vos expériences. Cette fonction de test sera utilisée lors de la correction du projet.