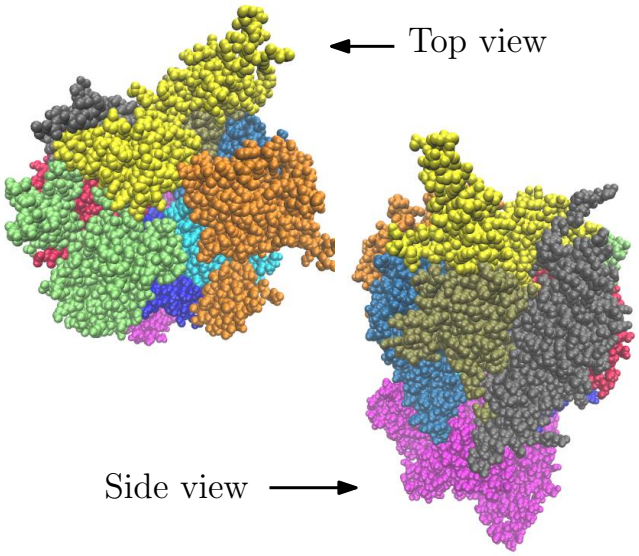
Projet SI4 - Algorithmes et Complexité.

Algorithmes pour l'inférence de connectivité

avec application en biologie structurale computationnelle



Johny Bond, Dorian Mazauric, Christophe Papazian, Stéphane Pérennes

2019 - 2020

Langages autorisées : Java, Python et C.

Date limite pour envoyer votre projet : 5 janvier 2020 à 23 heures (UTC+1).

Lien pour déposer le projet :
`https://www.dropbox.com/request/KPs8fjvAEII567gMhtmp`

Projet à réaliser en groupe de 4 (pas nécessairement d'un même groupe de TD).

Groupe à constituer au plus tard le mardi 3 décembre 2019 et inscrire dans le document
`https://drive.google.com/open?id=1hvJIl6Lf2c-rT_PEM9VBlIAoVXAeYURic2F4IX96-xw`

Question ? dorian.mazauric@inria.fr

Vous devez répondre pour chacune des questions (dans un fichier au format PDF obligatoire) et nous faire parvenir le code pour les questions 6 et 7.

L'objectif de ce projet est de développer des algorithmes efficaces pour déterminer les contacts entre les protéines d'un même assemblage moléculaire. Nous supposerons (tout au long du projet) que toutes les protéines d'un assemblage sont connues (données en entrée).

Par spectrométrie de masse native, il est possible d'obtenir de l'information concernant la structure d'un assemblage A. Plus précisément, nous avons la composition (en termes de protéines) de différents sous-complexes de A.

Un assemblage moléculaire A peut être représenté par un graphe $G = (V, E)$: chaque sommet $v \in V$ correspond à une protéine et il y a une arête entre deux sommets $u \in V$ et $v \in V$ si les deux protéines correspondantes à u et v sont en contact dans l'assemblage. L'objectif de ce projet est donc de déterminer l'ensemble E. Par spectrométrie de masse native, il est possible d'obtenir t sous-complexes $C_1, \ldots, C_t$ avec $C_i \subseteq V$ pour chaque $i \in \{1, \ldots, t\}$. Chaque sous-complexe forme une composante connexe de l'assemblage. En termes de graphe, chaque sous-complexe forme un sous-graphe connexe de G. Autrement dit, le sous-graphe $G[C_i]$ induit par les sommets de C_i est un sous-graphe connexe de G, pour chaque $i \in \{1, \ldots, t\}$.

Étant donné que chaque protéine est en contact avec un nombre limité d'autres protéines, nous proposons le problème suivant pour déterminer les contacts entre protéines.

Étant donné un entier $\Delta \geq 1$, un ensemble de sommets V et t sous-ensembles $C_1 \subseteq V, \ldots, C_t \subseteq V$, *le problème d'inférence de connectivité (IC)* consiste à trouver le plus petit ensemble d'arêtes E tel que
- pour chaque $i \in \{1, \ldots, t\}$, $G[C_i]$ est connexe,
- pour chaque sommet $v \in V$, le nombre d'arêtes de E incidentes à v est au plus Δ.

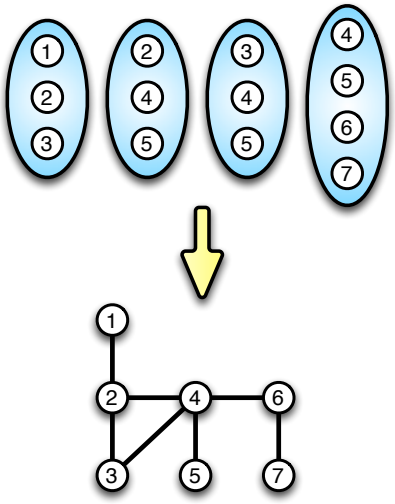
La deuxième contrainte signifie que le graphe $G = (V, E)$ induit par l'ensemble d'arêtes E a degré maximum Δ.

Nous formalisons ci-dessous la version décision du problème.

Nom : version décision du problème d'inférence de connectivité (IC)
Instance : un entier $\Delta \geq 1$, un entier $k \geq 1$, un ensemble de sommets V, t sous-ensembles $C_1 \subseteq V$, ..., $C_t \subseteq V$
Question : existe-t-il un ensemble d'arêtes E tel que $|E| \leq k$, $G[C_i]$ est connexe $\forall i \in \{1, \ldots, t\}$ et $G = (V, E)$ a degré maximum au plus Δ.

Remarque. Soit E une solution du problème. Le graphe $G = (V, E)$ n'est pas nécessairement connexe même si, d'un point de vue de l'application en biologie structurale, la contrainte de connexité du graphe semble naturelle. Mais, pour ce projet, nous n'avons pas cette contrainte.

Considérons l'exemple d'un assemblage A composé de 7 protéines. L'ensemble des sommets représentant les protéines est $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. La figure du haut représente quatre sous-complexes $C_1 = \{v_1, v_2, v_3\}$, $C_2 = \{v_2, v_4, v_5\}$, $C_3 = \{v_3, v_4, v_5\}$, $C_4 = \{v_4, v_5, v_6, v_7\}$. La figure du bas montre une solution optimale E composée de $|E| = 7$ arêtes pour le problème CI avec $\Delta = 4$. Observons que $G[C_i]$ est connexe pour tout $i \in \{1, \ldots, t\}$ (e.g. le sous-graphe induit par le sous-ensemble de sommets $\{v_4, v_5, v_6, v_7\}$ est un chemin de quatre sommets). De plus de chaque sommet part au plus quate arêtes.



Question 1. Considérons l'exemple de la figure.
Existe-t-il une solution pour le problème CI avec $\Delta = 3$? Si oui explicitez une solution optimale et prouvez son optimalité. Si non, prouvez-le.
Existe-t-il une solution pour le problème CI avec $\Delta = 2$? Si oui explicitez une solution optimale et prouvez son optimalité. Si non, prouvez-le.

Question 2. Pour cette question, nous considérons la version décision du problème IC. Nous supposons que $|C_i \cap C_j| \leq 1$ pour tout $i \in \{1, \ldots, t\}$ et pour tout $j \in \{1, \ldots, t\}$, $i \neq j$.

- Si $\Delta = |V|$, déterminez les valeurs de k pour lesquelles il y a une solution au problème IC (version décision).
- Si $k = |V|^2$, comment calculer les valeurs de Δ pour lesquelles il y a une solution au problème IC (version décision).

Question 3. Pour cette question, nous considérons la version minimisation du problème IC. Nous supposons encore que $|C_i \cap C_j| \leq 1$ pour tout $i \in \{1, \ldots, t\}$ et pour tout $j \in \{1, \ldots, t\}$, $i \neq j$.

- Quel est le nombre d'arbres couvrants différents d'un graphe complet à p sommets ?
- En déduire le nombre de solutions (optimales) différentes en fonction de la taille des C_i si $\Delta = |V|$.

Question 4. Pour cette question, nous considérons la version minimisation du problème. Nous supposons $\Delta = 2$ et $|C_i \cap C_j| \leq 1$ pour tout $i \in \{1, \ldots, t\}$ et pour tout $j \in \{1, \ldots, t\}$, $i \neq j$.

- Quelles sont les conditions d'existence d'au moins une solution ?
- Pour une instance admettant au moins une solution, expliciter le nombre de solutions différentes ?

Question 5. Les instances sont quelconques. Nous considérons le problème de décision. Décrire précisément deux algorithmes heuristiques polynomiaux et évaluer leurs complexités (les algorithmes ne donneront pas nécessairement une solution pour toutes les instances qui en admettent une). Vous donnerez ensuite quelques éléments qui nous permettront de comprendre pourquoi vos heuristiques devraient être efficaces.

Question 6. Le but est de construire des instances aléatoires pour le problème. Nous fixons uniquement le nombre de sommets à 100. Programmez une fonction qui, étant donné un entier $1 \leq p \leq 100$ et un entier $t \geq 1$, construit t sous-complexes $C_1 \subseteq V, \ldots, C_t \subseteq V$, avec $V = \{v_1, \ldots, v_{100}\}$ et $|C_i| = p$ pour tout $i \in \{1, \ldots, t\}$, tels que les p sommets de chaque C_i sont choisis de manière aléatoire et uniforme parmi les n sommets de V.

Question 7. Programmez vos deux heuristiques de la question 5.

Question 8. Le but de cette question est dévaluer l'efficacité de vos deux algorithmes heuristiques décrits dans la question 5 en utilisant la fonction qui génère des instances aléatoires de la question 6. Pour cette question le nombre de sommets / protéines est fixé à 100. Nous fixons $t = 20$ et $p = 10$. Vous remplirez trois tableaux de la manière suivante. Chaque case $f(k, \Delta)$ sera le pourcentage d'instances pour lesquelles votre algorithme a trouvé une solution. Vous préciserez le nombre de fois que l'algorithme a été exécuté (un nombre suffisamment grand pour avoir une valeur stable...). Vous ferez ensuite part de vos observations. Est-ce qu'un des deux algorithmes est significativement plus efficace pour certaines valeurs de k et de Δ ?

k Δ	10	20	30	40	50	60	70	80	90	100
2	f(2,10)	f(2,20)	f(2,30)	f(2,40)	f(2,50)	f(2,60)	f(2,70)	f(2,80)	f(2,90)	f(2,100)
3	f(3,10)	...	...	...	...	...	...	...	...	
4	f(4,10)	...								
5	f(5,10)	...								
6	f(6,10)	...								
7	f(7,10)	...				f(k,Δ)				
8	f(8,10)	...								
9	f(9,10)	...								
10	f(10,10)	...								

Question 9. Même question que la 8 avec
- $t = 20$ et $p = 15$.
- $t = 20$ et $p = 20$.
- $t = 30$ et $p = 10$.
- $t = 30$ et $p = 15$.
- $t = 30$ et $p = 20$.

Question 10 (bonus). Comment montreriez-vous que le problème est NP-complet en général ou pour des classes d'instances particulières ?