Vendredi 12 Mars 2004

Démo 6 – Séquençage

Jean-Eudes Duchesne

1. Introduction

Avant d'être capable de rechercher ou d'aligner des séquences, il faut d'abord connaître ces séquences ! Ceci peut sembler trivial, mais transformer un génome en une séquence lisible par l'homme n'est pas un problème simple, surtout si l'on veut représenter fidèlement la réalité biologique. Alors le problème est le suivant : Étant donné une molécule d'ADN, comment la séquencer, ou déterminer sa séquence en nucléotides ?

La méthode « Shotgun » est une des méthodes expérimentales de référence qui résout ce problème. Cette méthode coupe la molécule d'ADN en fragments et ensuite séquence chaque morceau obtenu par électrophorèse sur gel. La résolution de l'électrophorèse est bonne seulement pour de petits fragments (25-400pb), c'est pourquoi il faut fragmenter les longues molécules d'ADN.

Le problème étudier en bioinformatique est donc de réassembler la macromolécule initiale à l'aide de tous les fragments séquencés. En général, les algos de reconstruction de la séquence globale se posent les questions suivantes :

- Quelles sont les régions en commun (overlap detection) ?
- Comment les fragements se placent-ils (fragment layout) ?
- Choix du consensus ?

2. Détection des chevauchements

S'il n'y avait pas d'erreurs, il serait facile de détecter les chevauchements communs entre deux séquences en temps O(n) (revient à trouver le plus long préfixe qui est aussi le plus long suffixe d'une autre séquence ; lisez sur les arbres de suffixes ou les boîtes Z dans Gusfield pour des idées !). Malheureusement, même avec la meilleure méthode expérimentale possible, il y a toujours possibilité que des erreurs se soient insérées pendant le séquençage des fragments. Comment solutionner ce problème ?

Solution : PD !

Problème ? Trouver le plus long préfixe d'une séqunce qui correspond au suffixe d'une autre séquence en permettant k erreurs ou un seuil. Ceci est un cas de programmation dynamique simple entre les séquences S et T où l'on initialise la table à 0 dans la première colonne. La valeur maximale de la dernière colonne représente la position où commence le chevauchement entre le suffixe de S et le préfixe de T.

3. Disposition des fragments

Le choix de la disposition des fragments est souvent effectué par un algorithme glouton :

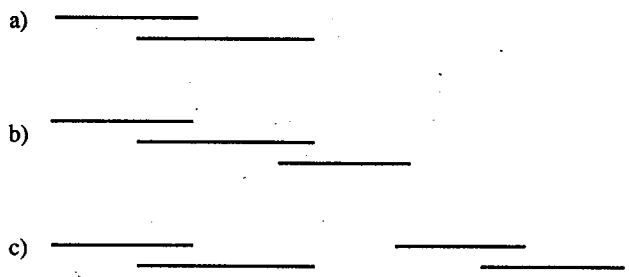


Figure 16.13: a. The first contig. b and c show the two different possibilities after the merge of the next best suffix-prefix pair.

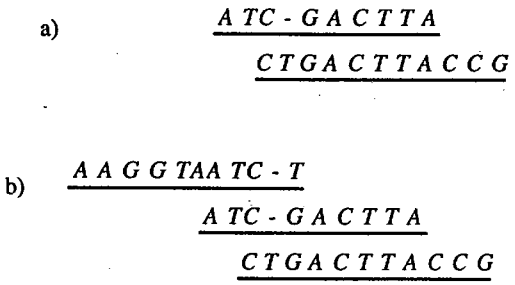
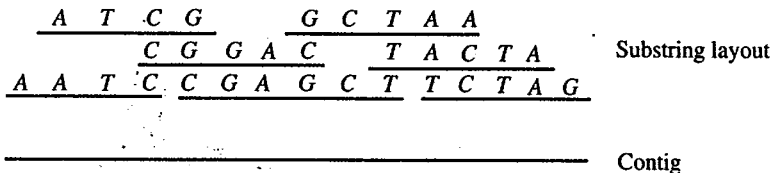


Figure 16.14: a. The first contig. The alignment in that pair of strings has one inserted space. The next best suffix-prefix match matches the suffix *ATCT* with the prefix *ATCG* as shown in b. That pairwise alignment does not contain a space, but a space is inserted into the third string when the string is added to the contig. That new space is inherited from the alignment of the first two strings.



4. Choix du consensus

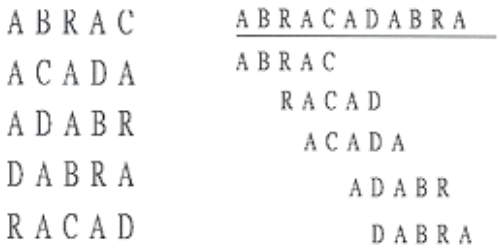
La façon la plus simple d'obtenir le consensus est de considérer le caractère le plus fréquent colonne par colonne de la disposition obtenue comme étant le caractère consensus pour la séquence globale. Évidemment, il est possible de considérer un modèle probabiliste plutôt qu'un caractère unique et il existe d'autres modèles, mais le choix du consensus n'est pas l'élément le plus constringent du problème.

5. Problèmes

- Le fait que l'électrophorèse ne fonctionne que pour de petits fragments signifie que s'il y a des problèmes de séquençage au niveau des fragments, les erreurs sont amplifiées sur la totatlité de la séquence.
- Les répétitions peuvent être mal interprétées. Il y a risque que celles-ci soient compressées en un seul élément dans la séquence globale. Ceci est surtout problématique quand les répétitions sont plus grandes que 400 pb, soit la résolution maximale de l'électrophorèse.

6. Super chaîne la plus courte

Soit un ensemble de k chaînes P = $\{s_1, s_2, s_3, \ldots, s_k\}$, une super chaîne de l'ensemble P est une chaîne qui contient tous les éléments de l'ensemble P en sous-chaînes.



Le problème qui minimise la taille de la super chaîne résout le problème d'assemblage intrinsèque à la méthode « shotgun ». Ainsi, il est possible de réduire le problème d'assemblage de la séquence à partir des ses fragments au problème de la super chaîne la plus courte. Par contre, ceci est un problème abstrait qui ne représente pas très bien la biologie. D'ailleurs, la réponse obtenue avec cet algorithme donne souvant une séquence plus courte que la séquence originale. Par contre, il est fort intéressant d'étudier ce problème puisqu'il n'est pas improbable que l'on en arrive un jour à un modèle biologiquement « correct ».

Malheureusement, en plus d'être biologiquement incorrect, le problème considéré est NP dûr ! Le meilleur algorithme connu produit une séquence au maximum 2.5 fois plus grande que la séquence optimale. L'algorithme que nous allons étudier assure une réponse au maximum 4 fois plus grande que la séquence optimale mais nous allons nous garder de prouver ou de démontrer ce fait. Il faut aussi spécifier que l'algorithme fait deux suppositions :

- Les brins d'ADN sont orientés
- Aucun élément de P n'est une sous-chaîne d'un autre élément de P.
- La source d'ADN est circulaire (plasmide ou BAC).

Soit deux chaînes, s et s', chevauche(s, s') est le chevauchement maximal entre le un suffix de s et un préfixe de s'. Soit préfix(s, s'), le préfixe de s restant après avoir supprimer la région définie par chevauche(s, s').

Si s = JAMBON et s' = BONNE, alors chevauche(s, s') = BON et préfix(s, s') = JAM.

Maintenant, supposons que dans la réponse optimale, les chaînes apparaissent de la gauche vers la droite dans l'ordre : $s_1s_2s_3\ldots s_n$. Le chevauchement entre deux séquences successives doit être maximal sinon la super chaîne n'est pas optimale. Ainsi, la réponse optimale (OPT) peut-être défénie comme suit :

$$OPT = |préfix(s_1, s_2)| + |préfix(s_2, s_3)| + \ldots + |prefix(s_{n-1},s_n)| + |prefix(s_n, s_1)| + |chevauche(s_n,s_1)|$$

Considérons le graphe de préfixes de ces séquences qui est un graphe complet dirigé avec poids G(V,E) avec V = {1, 2, …, n}, et poids(i, j) = $|préfix(s_i,s_j)|$. Notons que le chemin $i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_n \rightarrow i_1$ a un coût :

$$OPT = |préfix(s_1, s_2)| + |préfix(s_2, s_3)| + \ldots + |prefix(s_{n-1},s_n)| + |prefix(s_n, s_1)|$$

Donc, le chemin minimum du voyageur de commerce est une borne inférieure à notre solution optimale. Par contre, le problème du voyageur de commerce est trop difficile à calculer, ce qui fait que l'on se rabat sur un problème relié, soit la couverture de cycles minimale.

Pour solutionner ce nouveau problème, il faut construire une nouveau graphe G' = $(U \cup W, E')$ avec $U = \{u_1, u_2, u_3, \ldots, u_n\}$et $W = \{w_1, w_2, w_3, \ldots, w_n\}$tel que $(u_i, w_j) \in E'$ sii $(v_i, v_j) \in E$ et poids(u_i, w_j) = pois(v_i, v_j). Toute couverture de graphe en G correspond à une couverture parfaite en G' avec le même coût et inversement ! Ainsi, trouver la couverture de graphe minimale de G revient à trouver la couverture minimale en G'.

Si c est un cycle dans le graphe de préfixe, il est possible de définir deux nouvelles valeurs :

- $\alpha(c) = préfix(s_1, s_2) \circ préfix(s_2, s_3) \circ \ldots \circ préfix(s_{l-1},s_l) \circ préfix(s_l, s_1)$
- $\sigma(c) = \alpha(c) \circ s_1$

Ainsi, nous observons que :

- $|\alpha(c)|$ = poids(c) et $|\sigma(c)|$ = poids(c) + $|s_1|$
- $\sigma(c)$ est une super chaîne de $\{s_1, s_2, s_3, \ldots, s_l\}$.
- Toutes les chaînes de P sont sous-chaîne de : $\alpha(c_1) \circ \alpha(c_2) \circ \alpha(c_3) \circ \ldots = (\alpha(c))^{\infty}$
- $\sigma(c)$ a été construite en ouvrant un cycle c à la chaîne arbitraire s_1 qui est appelée la chaîne représentative de c.

Approximation du problème de la super chaîne la plus courte :

i. Construire le graphe de préfixe à partir de P
ii. Trouver une couverture de cycle minimale $C = \{c_1, c_2, c_3, \ldots, c_m\}$ du graphe de préfixe.
iii. Retourner $\sigma(c_1) \circ \sigma(c_2) \circ \ldots \circ \sigma(c_m)$.

7. Phred

Phred est un algorithme appelé « base caller », c'est à dire, un algorithme qui détermine si une base azoté est observée à un moment précis dans l'électrophorèse sur gel.

i. Prédicrion des pics idéaux.

   Utilise les séries de Fourrier. Observe les traces obtenues de l'électrophorèse pour établir un model global de pics (pics virtuels).

ii. Identification des pics observés.

   Si $2 * v(i) \geq v(i-1) + v(i+1)$ alors il y a un pic à la position v(i).

iii. Match des pics observés avec les pics idéeaux.

   a. Match faciles. Correspondances exactes entre les pics observés et les pics idéaux.
   b. Programmation dynamique pour aligner les pics qui ont été ignorés précédemment. C'est un algorithme qui maximise le score (aire sous la courde) sur trois opérations : split, shift-right, shift-left. Split, divise un pic en deux, mais ne peut pas créer un pic plus petit qu'un seuil donné. Shift-right est plus pénalisé que shift-left puisque cette option est moins probable (on veut traiter le cas où une molécule s'hybride sur elle-même et se déplace plus vite dans le gel).
   c. Raffinement avec le reste des données. Algorithme glouton qui attribue un pic au meilleur candidat disponible.

iv. Match des pics restants (uncalled). Un pic restant est considéré seulement s'il satisfait toutes les conditions suivantes :

   a. Doit être le signal le plus fort des quatre traces (A, C, G, T).
   b. Doit être au dessus d'un seuil donné.
   c. Ne doit pas avoir été préalablement divisé.
   d. Doit être borné par des pics préalablement assignés.
   e. Ajouter le pic favorise l'espacement entre les pics (améliore la solution globale).