M1 – BIM

Algorithmes sur les arbres et les graphes en bioinformatique

Algorithmes de reconstruction de longues séquences

Alessandra Carbone
Université Pierre et Marie Curie

Programme des semaines à venir:

Horaires - 10:30-12:30 cours lundi
13:30-17:45 TD/TME mercredi

Reconstruction des séquences d'ADN/génomes
Réarrangements des génomes
Phylogénie
Grands réseaux biologiques

Semaine du 9 novembre : cours et TD/TME seront assurés

Le premier exemple d'utilisation de la théorie des graphes en Biologie

Benzer 1950: un gène est linéaire, (où l'ADN n'est pas ramifié)

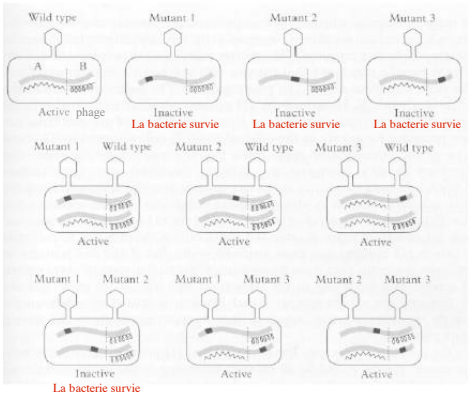
Les virus attaquent les bactéries

- Normalement les phages T4 tuent les bactéries
- D'autre part si T4 est muté (et que un gène important de T4 est supprimé), alors T4 peut devenir inactif et perdre son habilitée à tuer les bactéries
- Supposons que la bactérie soit infectée avec deux mutants et que tous les deux soient inactifs. Est-ce que la bactérie survie ?
- A **surprise**, une paire de virus inactifs peut tuer une bactérie même si tous les deux sont inactifs!
- Comment peut-on expliquer cela?

Idée de l'expérience de Benzer

- Idée: infecter la bactérie avec une paire de mutants du phage T4
- chaque mutant de T4 a un intervalle inconnu qui a été effacé de son génome
- Si deux intervalles chevauchent: la paire de T4 a perdu une partie du génome et est inactive – la bactérie survie
- Si les deux intervalles ne se chevauchent pas: la paire T4 a son génome entier et elle est active – la bactérie meurt

Complementation entre paires de mutants du phage T4

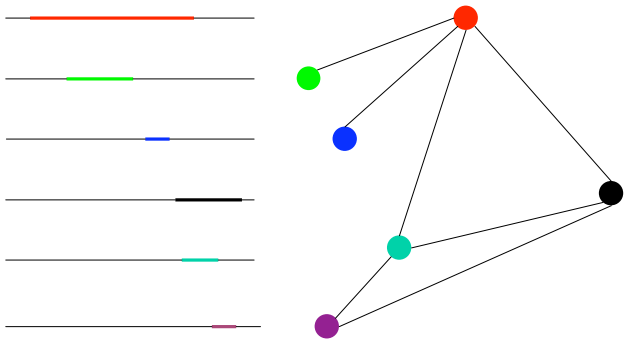


Idée soujacente : l'ADN est linéaire

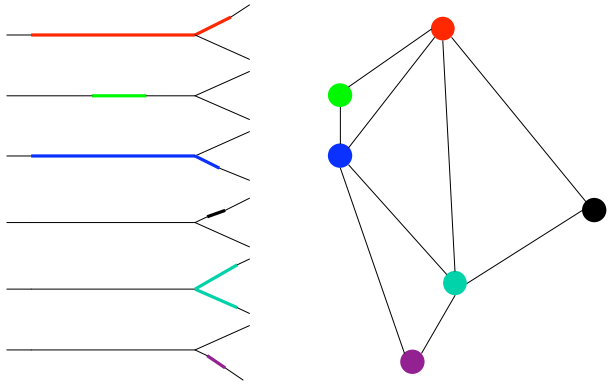
Construire un graphe d'intervalles: chaque mutant de T4 est un nœud et on place un arc entre paires de mutants quand la bactérie survie (i.e. quand les intervalles effacés de la paires de mutants chevauchent)

La structure des graphes d'intervalles révèle si l'ADN est linéaire ou ramifié.

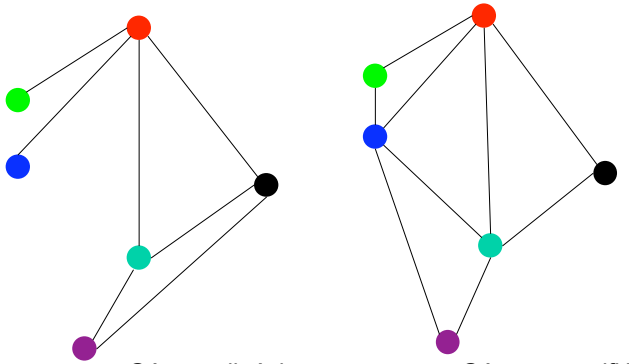
Graphe d'intervalles: gènes linéaires



Graphe d'intervalles: gènes ramifiés



Graphe d'intervalles: comparaison



Génome linéaire

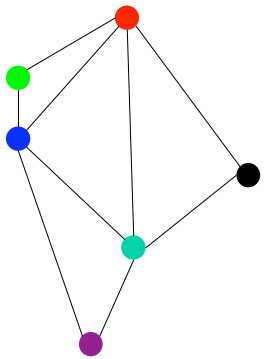
Génome ramifié

Il ne s'agit pas d'un graphe d'intervalles

G =(V,E) est un graphe d'intervalles

ssi

G est
1. « cordale » (cad, chaque cycle de >=4 arêtes contient une corde) et
2. le graphe complémentaire de G, $G^c = (V, E^c)$, est un graphe de comparaison (cad issu d'un ordre partiel (V,<):
   $(x,y) \in E^c$ ssi x<y ou y<x)



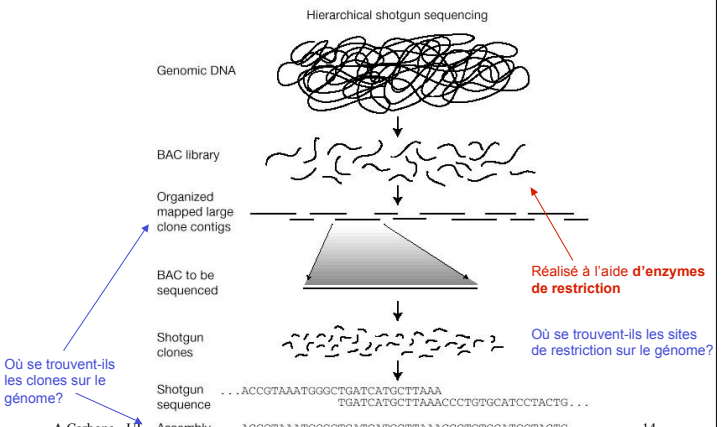
Génome ramifié

Trouver un ensemble d'intervalles qui représente un graphe d'intervalles peut être utilisé comme stratégie pour rassembler des sous-séquences contiguës d'ADN (Zhang et al 1994).

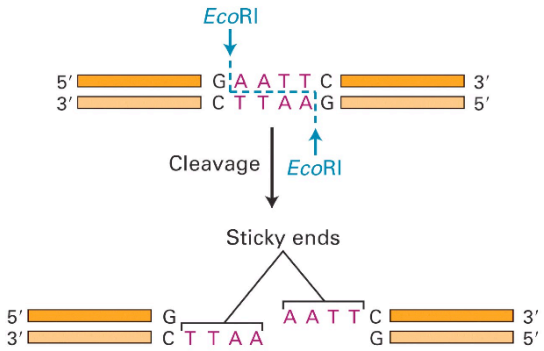
* Séquençage de l'ADN, 1977
* Premier génome entier,1996

- reconstruction de segments à partir de leur tailles
- reconstruction de segments à partir de leurs séquences
- reconstruction des positions des segments par rapport à des positions clé dans le génome

Séquençage des génomes

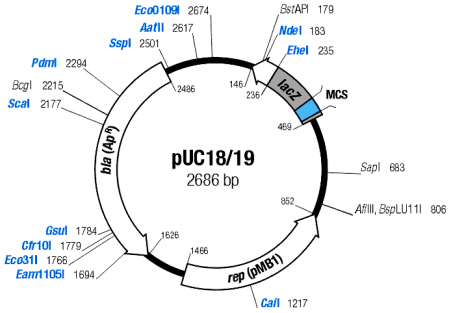


Ciseaux moléculaires : les enzymes de restriction



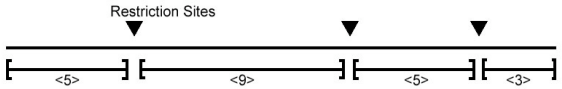
Cartes de restriction

- Une carte qui montre les positions des sites de restriction sur une séquence d'ADN.
- Si la séquence d'ADN est connue alors la construction de la carte de restriction est un exercice banale
- Au début de la biologie moléculaire, les séquences d'ADN étaient souvent inconnues. Donc, les biologistes ont du résoudre le problème de la construction des carte des sites de restriction sans connaître les séquences d'ADN



Digestion complète : Full Restriction Digest

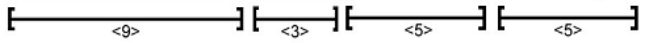
- Couper l'ADN à chaque site de restriction porte à recuperer plusieurs fragments de restriction:



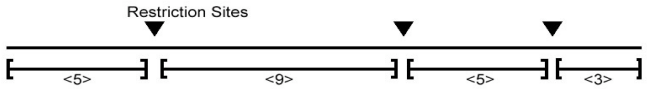
Est-il possible de reconstruire l'**ordre** des fragments à partir de leur taille {3,5,5,9} ?

Full Restriction Digest: solutions multiples

- Ordre alternative des fragments:

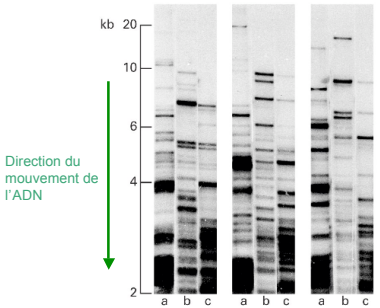


à la place de



Mesure expérimentale de la longueur des fragments de restriction

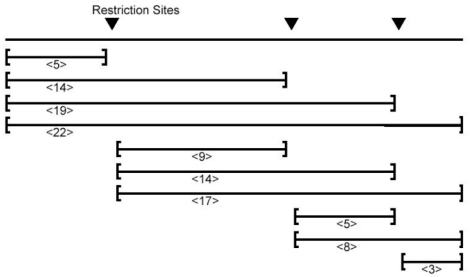
- Les enzymes de restriction coupent l'ADN dans des fragments de restriction.
- L'électrophorèse est un processus de séparation de l'ADN par taille et mesure la taille des fragments de restriction.
- Il peut séparer des fragments d'ADN à 1 seul nucléotide près, pour des fragments de taille variante jusqu'à 500 nucléotides.



Les fragments plus petits voyagent plus rapidement et vont plus loin

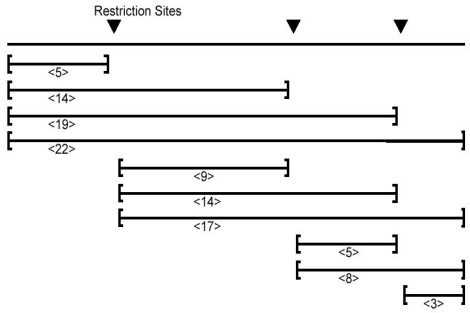
Digestion partielle (Partial Digest) : un exemple

- Partial Digest donne les 10 fragments de restriction suivant :



Multi-ensemble de fragments de restriction

On suppose que la multiplicité d'un fragment puisse être détectée, cad, le nombre de fragments de restrictions de la même longueur peut être déterminé (par exemple, en observant l'épaisseur des bandes dans le gel).



Multi-ensemble:
{3, 5, 5, 8, 9, 14, 14, 17, 19, 22}

Partial Digest : notation

X: L'ensemble de ***n*** entiers représentant la position de toutes les coupures dans la carte de restriction, incluant le début et la fin

n: Le nombre totale de coupures

Δ*X*: Le multi-ensemble des entiers représentants les longueurs de chacun des fragments produit par une digestion partielle

Exemple : représentation d'une instance de "Partial Digest"

X	0	2	4	7	10
0		2	4	7	10
2			2	5	8
4				3	6
7					3
10					

← position

← longueur des fragments

Représentation de Δ***X*** = {2, 2, 3, 3, 4, 5, 6, 7, 8, 10} comme un tableau 2-dimensionnel, avec les éléments de l'ensemble

X = {0, 2, 4, 7, 10}

sur lignes et colonnes. Les éléments (*i*, *j*) représentent $x_j - x_i$ pour $1 \leq i < j \leq n$.

Problème (Partial Digest Problem): étant donné toutes les distances entre points sur une droite, reconstruire les positions de ces points.
Entrée: le multi-ensemble de distances entre paires de points *L*, contenant *n(n-1)/2* entiers
Sortie: Un ensemble *X*, de *n* entiers, tel que Δ*X* = *L*

Il n'est pas toujours possible de reconstruire de façon unique un ensemble ***X*** basé seulement sur Δ***X***.

Par exemple:

1. les ensembles *X* = {0, 2, 5} et (*X* + 10) = {10, 12, 15} génèrent Δ***X***={2, 3, 5} comme ensemble de digestion partielle.
2. les ensembles *X* = {0,1,2,5,7,9,12} et *Y* = {0,1,5,7,8,10,12} génèrent: Δ*X* = Δ*Y* = {1, 1, 2, 2, 2, 3, 3, 4, 4, 5, 5, 5, 6, 7, 7, 7, 8, 9, 10, 11, 12}

Ensembles Homométriques

	0	1	2	5	7	9	12
0		1	2	5	7	9	12
1			1	4	6	8	11
2				3	5	7	10
5					2	4	7
7						2	5
9							3
12							

	0	1	5	7	8	10	12
0		1	5	7	8	10	12
1			4	6	7	9	11
5				2	3	5	7
7					1	3	5
8						2	4
10							2
12							

Les biologistes sont souvent intéressés à connaître tous les ensembles homométriques

Algorithmes "force brute"

Algorithmes de recherche exhaustive: ils examinent toutes les possibilités. Ils sont très rarement efficaces.

BruteForcePDP(*L*, *n*):

- Trouver le fragment de restriction de longueur maximale ***M*** dans L. ***M*** est la longueur de la séquence d'ADN.
- Pour chaque ensemble possible de positions ***X***={0, x_2, … ,x_{n-1}, *M*} (Les x_i sont des entiers arbitraires dans l'intervalle [0,M])
  calcule le Δ***X*** correspondant
- Si Δ***X*** est égale à l'ensemble de digestion issu des expériences ***L***, alors ***X*** est la carte de restriction correcte

BruteForcePDP a une complexité en temps de $O(M^{n-2})$ car il doit examiner tous les possibles ensembles de *n* positions (ils sont $\binom{M-1}{n-2}$). Une façon de l'améliorer est de délimiter les valeurs de x_i seulement aux valeurs qui apparaissent dans ***L***.

AnotherBruteForcePDP(*L*, *n*)

M ← élément maximum dans *L*
Pour chaque ensemble de *n* – 2 entiers $0 < x_2 < \dots x_{n-1} < M$ dans *L*
  $X \leftarrow \{ 0, x_2, \dots, x_{n-1}, M \}$
Calculer ΔX à partir de *X*
si $\Delta X = L$
    retourner *X*
sinon sortir "pas de solution"

Algorithme plus efficace, mais aussi lent.
Si ***L*** = {2, 998, 1000} (***n*** = 3, ***M*** = 1000), BruteForcePDP est extrêmement lent, et AnotherBruteForcePDP assez rapide. Beaucoup moins d'ensembles sont examinés, mais sa complexité en temps reste exponentielle : $O(n^{2n-4})$

Algorithme Branch and Bound pour PDP

1. Poser ***X*** = {0}
2. Eliminer l'élément le plus grand de ***L*** et l'insérer dans ***X***
3. Vérifier si l'élément se positionne bien à la droite ou à la gauche de la carte de restriction.
4. Si oui, rechercher toutes autres longueurs qui sont générées par cet élément et les enlever de ***L***
5. Revenir à l'étape 1 jusqu'à quand ***L*** est vide.

ALGORITHME EXPONENTIEL mais efficace dans la pratique

Definons $\Delta(y, X)$ comme le multi-ensemble de toutes les distances entre le point ***y*** et tout autre point dans l'ensemble ***X***

$\Delta(y, X) = \{|y - x_1|, |y - x_2|, \dots, |y - x_n|\}$

pour $X = \{x_1, x_2, \dots, x_n\}$

```
PartialDigest(L):
  width ← Maximum element in L
  DELETE(width, L)
  X ← {0, width}
  PLACE(L, X)
```

```
PLACE(L, X) :
 if L is empty
       output X
       return
y ← maximum element in L
Delete(y,L)
if Δ(y, X ) ⊆ L
       Add y to X and remove lengths Δ(y, X) from L
       PLACE(L,X )
       Remove y from X and add lengths Δ(y, X) to L
if Δ(width-y, X ) ⊆ L
      Add width-y to X and remove lengths Δ(width-y, X) from L
      PLACE(L,X )
      Remove width-y from X and add lengths Δ(width-y, X ) to L
return
```


Exemple

$L = \{ 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 \}$
$X = \{ 0 \}$

Enlever 10 de ***L*** et l'inserer dans ***X***. On sait que cela doit etre la taille de la sequence d'ADN parce qu'il s'agit du plus grand fragment.

$L = \{ 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 \}$
$X = \{ 10 \}$

$L = \{ 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 \}$
$X = \{ 0, 10 \}$

Prendre 8 de ***L*** et définir ***y*** = 2 ou 8. On explore le cas ***y*** = 2:

$L = \{ 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 \}$
$X = \{ 0, 10 \}$

Les distances de *y=2* aux autres éléments de *X* sont
$\Delta(\boldsymbol{2}, \boldsymbol{X}) = \{8, 2\}$, donc on enlève {8, 2} de ***L*** et on ajout 2 à ***X***.

$L = \{ 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 \}$
$X = \{ 0, 2, 10 \}$

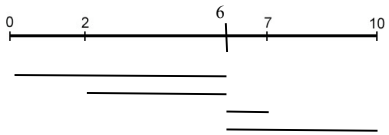
On enlève 7 de ***L*** et on définit $y = 7$ ou $y = 10 - 7 = 3$.
On explore ***y*** = 7 tout d'abord, et donc $\Delta(7, X) = \{|7 - 0|, |7 - 2|, |7 - 10|\} = \{7, 5, 3\}$.

$L = \{ 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 \}$
$X = \{ 0, 2, 10 \}$

Pour ***y*** = 7 : on enlève {7, 5 ,3} de ***L*** et on ajoute 7 à ***X***.

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 7, 10 }

On prend 6 de ***L*** et on definit ***y*** = 6. On s'aperçoit que Δ(*6*, *X*) = {6, 4, 1 ,4}, qui n'est pas un sous-ensemble de ***L***. Donc on explore pas cette branche.



L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 7, 10 }

On définit *y* = 4. Δ(*4*, *X*) = {4, 2, 3 ,6}, qui est un sous-ensemble de ***L*** et on explorera cette branche. On enlève {4, 2, 3 ,6} de ***L*** et l'on ajoute 4 a ***X***.

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 4, 7, 10 }

L est vide, et nous avons une solution, cad *X*.

Pour trouver d'autres solutions, on reviens en arrière.

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 7, 10 }

et encore :
L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 10 }

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 2, 10 }

Cette fois on explore y = 3. Δ(*3*, *X*) = {3, 1, 7}, qui n'est pas un sous-ensemble de *L*, et on n'explorera pas cette branche.

On reviens en arrière jusqu'à la racine. On n'a pas d'autres solutions. (Le y=8 étant symétrique de y=2)

L = { 2, 2, 3, 3, 4, 5, 6, 7, 8, 10 }
X = { 0, 10 }

Analyse de l'algorithme PartialDigest

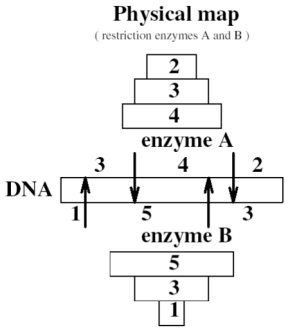
- Exponentiel au pire des cas, mais très performant en moyenne
- Informellement, soit *T(**n**)* le temps pris par PartialDigest pour localiser *n* coupures
  - Cas 1: pas de branchement: $T(n) < T(n-1) + O(n)$
    où O(n) est le temps utilisé pour les opérations sur X et L.
    - Quadratique
  - Cas 2: branchement: $T(n) < 2T(n-1) + O(n)$
    - Exponentiel

Il y a des exemples pathologiques qui forcent l'algorithme à explorer les alternatives gauche et droite quasiment à toutes les étapes. Il existe un algorithme polynomial (Nivat et al., 2002)

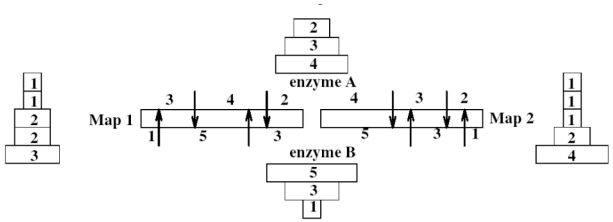
Double Digest Mapping

- Double Digest est une autre méthode expérimentale employée dans la reconstruction des cartes de restriction: elle utilise **deux** enzymes de restriction et **trois** digestions pleines:
  - Une avec seulement le premier enzyme
  - Une avec seulement le deuxième enzyme
  - Une avec les deux enzymes
- D'un point de vue calculatoire, le problème de Digestion Double est plus complexe que celui de Digestion Partielle.

Double Digest: Example



Double Digest: exemple



Sans information sur Δ***X,*** i.e. distances générées par ***A+B***, il est **impossible** de résoudre le problème de digestion double (comme démontré dans le diagramme).

Double Digest : le problème

Entrée: ΔA – longueurs des fragments d'après la digestion de l'enzyme ***A***.

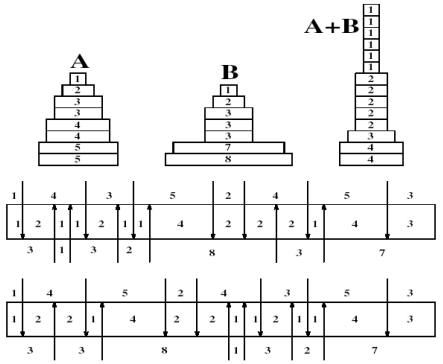
ΔB – longueurs des fragments d'après la digestion de l'enzyme ***B***.

ΔX – longueurs des fragments d'après la digestion des *deux* enzymes ***A*** et ***B***.

Sortie: ***A*** – localisation des coupures dans la carte de restriction de l'enzyme ***A***.

B – localisation des coupures dans la carte de restriction de l'enzyme ***B***.

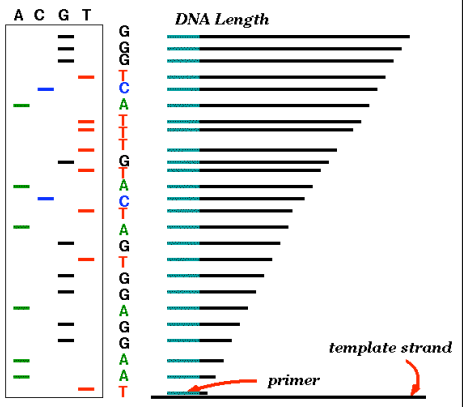
Double Digest: solutions multiples



Reconstruction des fragments à partir de leur séquences

Lecture de petites séquences, "reads" (méthode de Sanger)

1. Commencer par des primer (site de restriction)
2. Faire croître la chaîne d'ADN
3. Inclure ddNTPs
4. Arrête la réaction sur tous les points possibles
5. Séparer les produits par la longueur, en utilisant la gel électrophorèse.



Assemblage de fragments lus

- **Défis computationnel:** assembler les petits fragments individuels (reads) dans une seule séquence génomique (“superstring”)
- Jusqu’à la fin des années 1990s le “shotgun fragment assembly” du génome humain était vu comme un problème intraitable.

Shortest Superstring Problem

Problème: étant donné un ensemble de chaînes, trouver la chaîne la plus courte qui les contient toutes

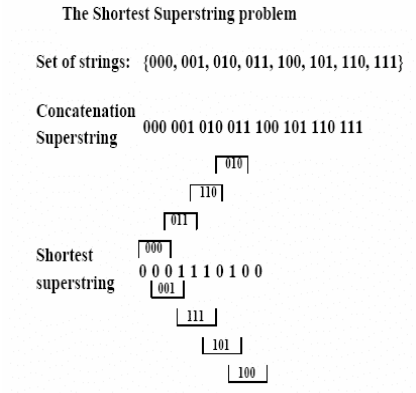
Entrée: Chaînes $s_1, s_2, \ldots, s_n$

Sortie: Une chaîne s qui contient toutes les chaînes $s_1, s_2, \ldots, s_n$ comme sous-chaînes, tel que la longueur de s est minimisée

Complexité: NP – complet

Note: cette formulation ne considère pas les erreurs de séquençage

Shortest Superstring Problem: exemple



Réduction de SSP à TSP

(où le Travel Salesman Problem est NP-complet)

- Définir *overlap* (s_i, s_j) comme la longueur du préfix le plus long de s_j qui s’apparie avec un suffixe de s_i (sans erreurs).

aaaggcatcaaatctaaaggcatcaaa

aaaggcatcaaatctaaaggcatcaaa

Qu’est-ce que *overlap* (s_i, s_j) ***pour ces chaînes ?***

Réduction de SSP à TSP

- Définir *overlap (s_i, s_j)* comme la longueur du préfix le plus long de s_j qui s'apparie avec un suffixe de s_i.

aaaggcatcaaatctaaaggcatcaaa
aaaggcatcaaatctaaaggcatcaaa
aaaggcatcaaatctaaaggcatcaaa

overlap=12

Réduction de SSP à TSP

- Définir *overlap (s_i, s_j)* comme la longueur du préfix le plus long de s_j qui s'apparie avec un suffixe de s_i.

aaaggcatcaaatctaaaggcatcaaa
aaaggcatcaaatctaaaggcatcaaa
aaaggcatcaaatctaaaggcatcaaa

- Construire un graphe avec *n* noeuds qui représente les *n* chaînes $s_1, s_2, \ldots, s_n$.
- Insérer des arcs pondérés de longueur (poids) *overlap (s_i, s_j)* entre les noeuds s_i et s_j.
- Chercher le plus court chemin qui visite chaque noeud exactement une fois. Il s'agit du **Problème du Voyageur de Commerce** (**Traveling Salesman Problem** - TSP), qui on sait être NP – complet.

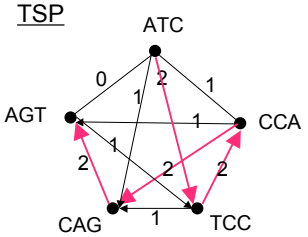
Réduction de SSP à TSP: un exemple

S = { ATC, CCA, CAG, TCC, AGT }

SSP

AGT
CCA
ATC
ATCCAGT
TCC
CAG

TSP



ATCCAGT

La technologie permet de déterminer tous les l-mers, pour un l fixé, qui sont contenus dans une chaîne S d'ADN

Le but est de reconstruire la chaîne S à partir de ses l-mers.

On peut formuler le problème comme un problème du chemin Hamiltonien sur un graphe, mais c'est plus productif de le voir comme un problème de chemin Eulérien.

Graphe de reconstruction: composition des *l-mer*

Etant donnée une chaîne *s* de longueur *n :*

- ***Spectre(s,l)*** – multi-ensemble non ordonné de tous les possibles *(n – l + 1)* *l*-mers

Pour *s* = TATGGTGC tous les *Spectre(s,3)* suivants sont équivalents

{TAT, ATG, TGG, GGT, GTG, TGC}
{ATG, GGT, GTG, TAT, TGC, TGG}
{TGG, TGC, TAT, GTG, GGT, ATG}

- Des séquences différentes peuvent avoir le même spectre:

Spectre(GTATCT,2)=
Spectre(GTCTAT,2)=

{AT, CT, GT, TA, TC}

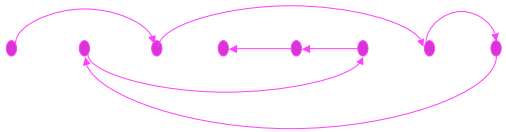
- Problème: Reconstruire une chaîne à partir de ses *l*-mers
- Entrée: Un ensemble *S*, représentant tous les *l*-mers d'une chaîne *s* inconnue
- Sortie: la chaîne *s* tel que *Spectre(s,l)* = *S*

Approche du chemin hamiltonien

Chaque nœud représente un k-mer et un arc est dirigé d'un nœud a l'autre si le deuxième k-mer peut être obtenu du premier en supprimant le dernier caractère et en ajoutant un caractère a la droite

S = { ATG AGG TGC TCC GTC GGT GCA CAG }

H ATG AGG TGC TCC GTC GGT GCA CAG

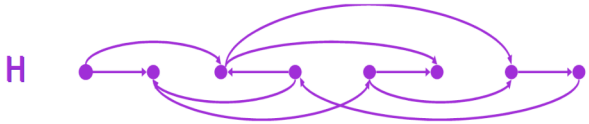


ATGCAGGTCC

 Le chemin a visité chaque NOEUD qu'une fois

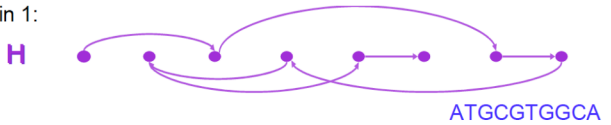
Un graphe plus compliqué:

S = { ATG TGG TGC GTG GGC GCA GCG CGT }

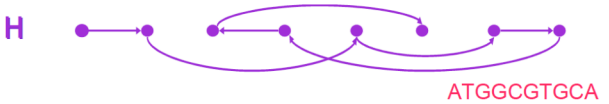


S = { ATG TGG TGC GTG GGC GCA GCG CGT }

Chemin 1:



Chemin 2:

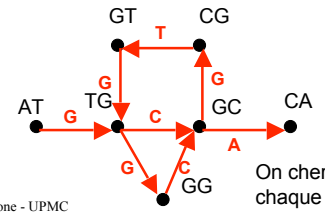


Approche du chemin Eulérien

S = { ATG, TGC, GTG, GGC, GCA, GCG, CGT }

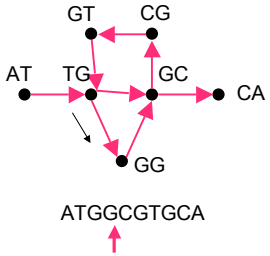
Noeuds correspondent à (*k* – 1) – mers : { AT, TG, GC, GG, GT, CA, CG }

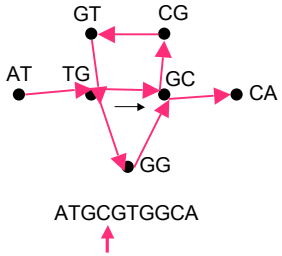
Arcs correspondent à *k* – mers observés



On cherche un chemin qui visite chaque ARC une fois

Le graphe ayant nœuds { AT, TG, GC, GG, GT, CA, CG } corresponde a deux différents chemins :



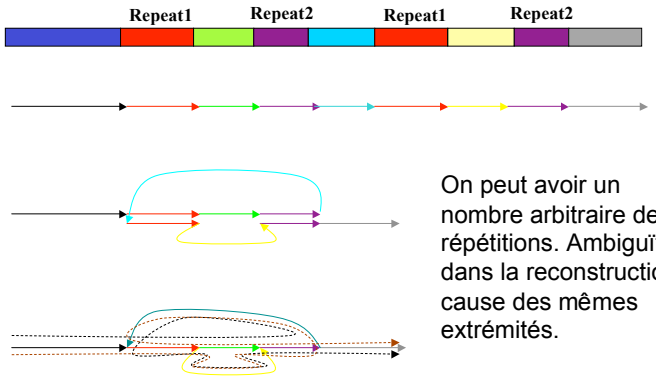


La réduction de SSP à EPP permet de trouver une solution polynomiale

En réalité on a plusieurs contraintes à résoudre:

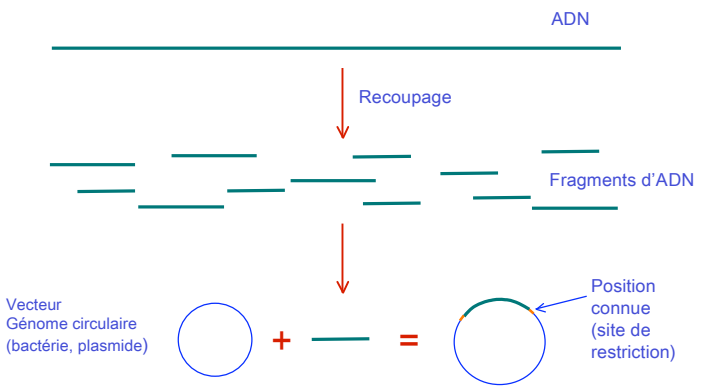
- erreurs de séquençage
- k-mer manquants
- répétitions

Répétitions multiples



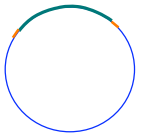
On peut avoir un nombre arbitraire de répétitions. Ambiguïté dans la reconstruction à cause des mêmes extrémités.

Schéma de séquençage d'ADN

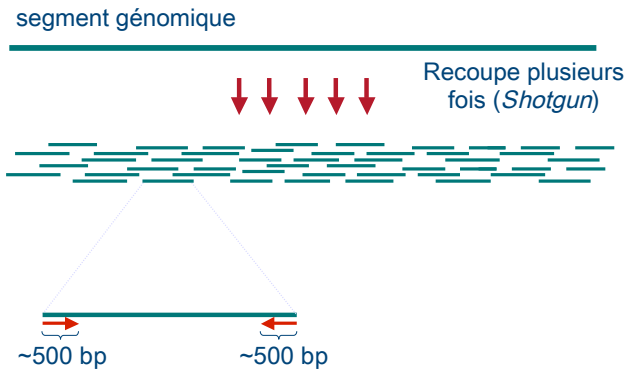


Different Types de Vecteurs

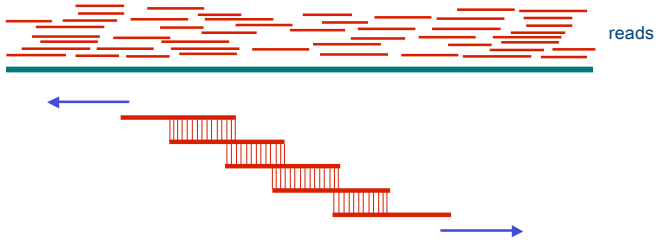
VECTEUR	Tailles des fragments insérés (bp)
Plasmide	2,000 - 10,000
Cosmide	40,000
BAC (Bacterial Artificial Chromosome)	70,000 - 300,000
YAC (Yeast Artificial Chromosome)	> 300,000 Utilisé de moins en moins



Shotgun Sequencing



Assemblage des fragments



Recouvrir la région avec une redondance de ~7-fold

Chevaucher les reads et les étendre pour reconstruire la région génomique originale

Recouvrement des Reads

Longueur du segment génomique: ***L***
Nombre de reads: ***n***
Longueur de chaque read: ***l***

Recouvrement ***C = n l / L***

Combien de recouvrements sont nécessaires ?

Modèle de **Lander-Waterman**

Scénario théorétique, modèle de Lander-Waterman

Taille du génome : 1million de paires de bases (~25 longueurs de clone)

Longueur moyenne d'un clone: 40kb
Couverture des clones = 10x

Sondes : 500
Longueur des sondes : 10-40bp.

La première couverture du maïs FPC, 2001

Taille du génome: 2500 Mb

Longueur moyenne d'un clone : 150kb
Couvertures de clones : 17x

Couverture pas suffisante

414 contigs ont été assignés aux chromosomes
Les contigs couvrent approximativement 2148 Mb,
Les contigs assignées aux chromosomes couvrent approximativement 1839 Mb.

Modèle de **Lander-Waterman**

Supposons L = longueur d'un clone
G = longueur du génome

On choisi N clones aléatoirement

Couverture : quelle est la fraction du génome qui est couverte par les clones ?

Si b est un point aléatoire du génome et c un clone arbitraire, alors la probabilité d'un point b d'être inclus dans le clone c est

$$Pr(b \in c) = L/G$$

et la probabilité que b soit hors des N clones choisis est

Longueur totale des clones

$$Pr(b \notin c, \text{ pour tout } c) = (1 - L/G)^N = (1-L/G)^{GN/G} \sim e^{-(NL)/G}$$

L<<G et N<<G

R= NL/G **facteur de redondance** ou **«équivalent du génome»**

La région du génome attendue ne pas être couverte est $E(\text{région pas couverte}) = e^{-R}$

R	Couverture
1	0.63
2	0.865
3	0.95
4	0.98
5	0.993

R > 2 donne une bonne couverture !

Pour la reconstruction on doit considerer les **chevauchement des clones**:

Supposons que la longueur d'un clone est 1
N =nombre des clones
R = facteur de redondance
la position de début des clones suit un processus de Poisson de taux λ

On défini un facteur de chevauchement minimale θ entre clones et on dit que deux clones chevauchent ssi ils partagent un interval d'une longueur au moins θ.

Terminologie : **contig** (ou **île**) = un ensemble de clones couvrant un segment continu du génome + leurs distances physiques.

Théorème (Lander-Waterman 1988) :

1. Le nombre attendu de contig est

$$N e^{-R(1-\theta)}$$

2. Le nombre attendu de contig avec exactement $j \geq 1$ clones est

$$N e^{-2R(1-\theta)} (1-e^{-R(1-\theta)})^{(j-1)}$$

3. Le nombre attendu de clones dans un contig paru est

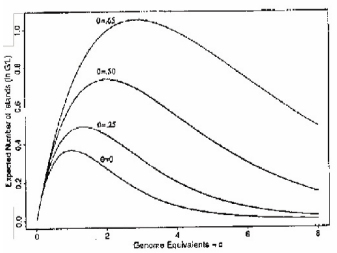
$$e^{R(1-\theta)}$$

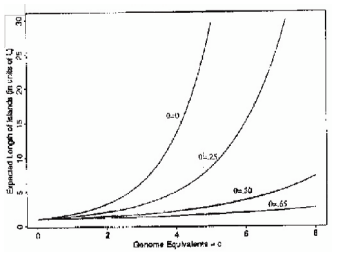
4. La longueur attendue d'un contig paru est

$$e^{R(1-\theta)} -1 /R + \theta$$

Plus grande est la couverture, plus petit est le nombre de contigs attendus

Plus grande est la couverture, plus grande est la longueur des contigs existants





Nombre de contig en fonction de R et θ.

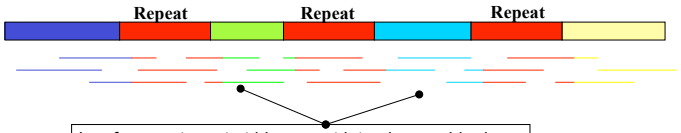
Longueur des contig en fonction de R et θ.

Facteur de redondance

Facteur de chevauchement des clones

Challenges dans l'assemblage des fragments

- Répétitions: un problème **majeur** pour l'assemblage des fragments
- > 50% du génome humain sont les répétitions:
  - plus que 1 million de répétitions *Alu* (environs 300 bp)
  - environs 200,000 répétitions LINE (1000 bp et plus)



Les fragments vert et bleue sont interchangeable dans l'assemblage d'ADN répété

Types de répétitions

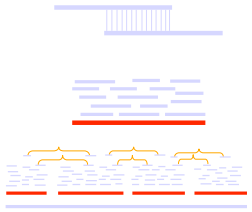
- **Low-Complexity DNA** (e.g. ATATATATACATA…)
- **Microsatellite repeats** $(a_1 \ldots a_k)^N$ où k ~ 3-6 (e.g. CAGCAGTAGCAGCACCAG)
- **Transposons/retrotransposons**
  - **SINE** Short Interspersed Nuclear Elements (e.g., *Alu*: ~300 bp longueur, 10^6 copies)
  - **LINE** Long Interspersed Nuclear Elements ~500 - 5,000 bp long, 200,000 copies
  - **LTR retroposons** Long Terminal Repeats (~700 bp) a chaque extrémité
- **Gene Families** gènes qui dupliquent et divergent
- **Segmental duplications** ~très longues, copies très similaires

Chevauchement à large echelle (Overlap,Layout et Consensus)

Assembleurs: ARACHNE, PHRAP, CAP, TIGR, CELERA

Overlap: trouver reads probablement chevauchants

Layout: fusionner reads dans des contigs et contigs dans des supercontigs



Consensus: deriver la sequence d'ADN et corriger les erreurs de reads

..ACGATTACAATAGGTT..

Overlap

- Trouver le chevauchement le meilleur entre le suffixe d'un read et le préfix d'un autre read
- Due à des erreurs de séquençage, il faut utiliser de la programmation dynamique pour trouver *l'alignement des chevauchements* optimale
- Appliquer une méthode de filtration pour exclure les paires de fragments qui ne partagent pas de façon significative une chaîne longue commune

Reads chevauchants

- Trier tous les *k*-mers dans les reads ($k \sim 24$)
- Trouver les paires de reads qui partagent un k-mer
- Etendre à un alignement complet – rejeter s'il n'y a pas une similarité >95%

Overlapping Reads and Repeats

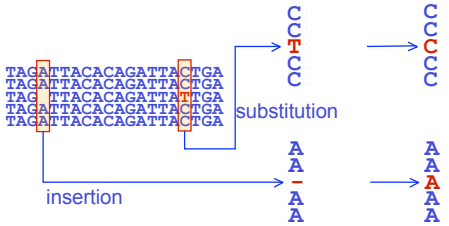
- Un *k*-mer qu'apparaît N fois dans le génome, comporte N^2 comparaisons
- Pour un *Alu* qu'apparaît 10^6 fois → 10^{12} comparaisons – beaucoup trop !!
- **Solution:**
  Eliminer tous les *k*-mers qu'apparaissent plus que
  $t \times$ Coverage, ($t \sim 10$)

Trouver les Reads chevauchants

Reconstruire des alignements multiples locaux à partir de reads chevauchants

Trouver les reads chevauchants

- Corriger les erreurs en utilisant l'alignement multiple



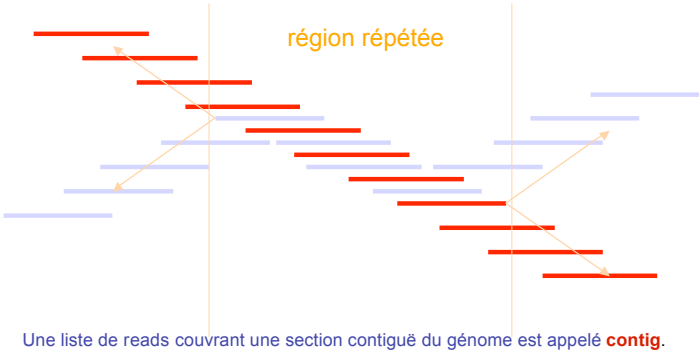
- Associer des scores d'alignement
- Accepter alignements avec des bons scores

Layout

Les répétitions représentent un défis majeur pour le layout aussi:

est-ce que 2 fragments alignés chevauchent vraiment, ou ils proviennent de 2 copies d'une répétition?

Rejoindre les Reads dans les Contigs



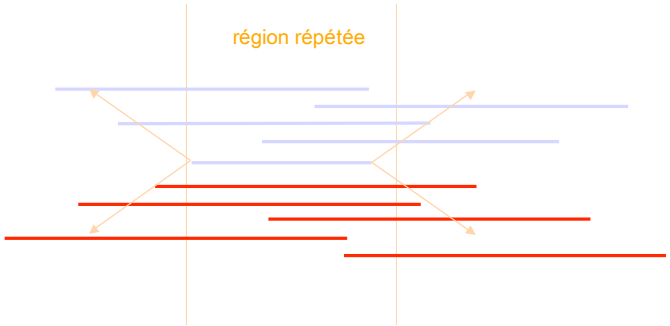
Une liste de reads couvrant une section contiguë du génome est appelé **contig**.

Rejoindre les reads jusqu'a quand on trouve des **bornes** pour les répétitions potentielles

Répétitions, erreurs, et longueurs des Contig

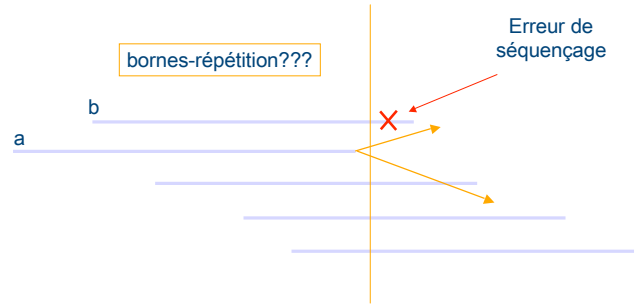
- Répétitions qui sont plus courtes que la longueur d'un read ne posent pas de problème
- Répétitions qui sont constituées par un nombre de bases différentes plus grand que le nombre d'erreurs de séquençage ne posent pas de problème
- Pour que une petite portion de génome **apparaît** non-répétitive, essayer de:
  - Augmenter la longueur des reads
  - Décrémenter le taux d'erreurs du séquençage

1ere étape: Fusion des Reads dans les Contigs



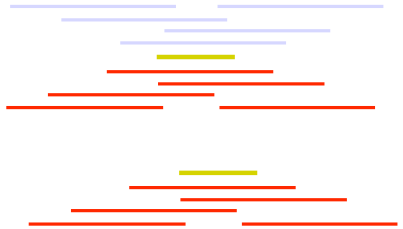
- Ignorer reads non-maximaux (par rapport a une longueur fixée)
- Fusionner seulement reads maximaux dans des contigs

2eme étape: fusion des Reads dans les Contigs



Ignorer reads “pendants”, pour la détection des bornes des répétitions

3eme étape: fusion des Reads dans les Contigs

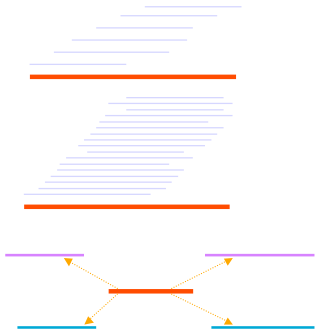


?????

Non ambigus

- Insérer les reads non-maximaux si non-ambigus

4eme étape: lier Contigs dans des Supercontigs



Densité normale

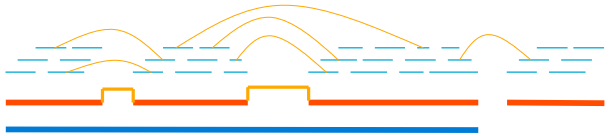
Trop dense: surcollapse?

Liens inconsistants: surcollapse?

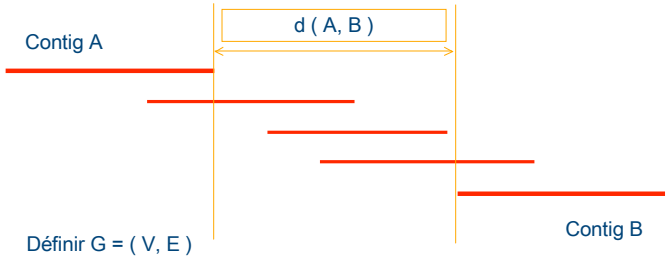
Lier Contigs dans des Supercontigs

Trouver tous les liens entre contigs uniques

Connecter contigs incrementalement, si ≥ 2 liens



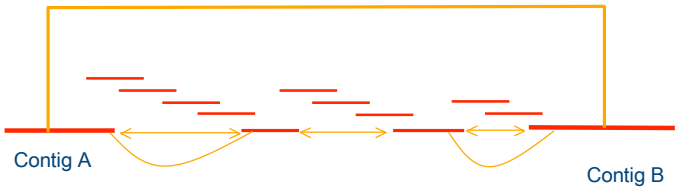
Lier Contigs dans des Supercontigs



Définir G = (V, E)
V := contigs
E := (A, B) tel que d(A, B) < C

Raison pour le faire: Efficacité; le chemin le plus court ne peux pas être calculé

Lier Contigs dans des Supercontigs



Définir T: contigs reliés soit à A soit à B

Remplir le “trou” entre A et B s’il y a un chemin dans G qui passe seulement par des contigs dans T

Consensus

- Une séquence consensus est dérivée d’un profile de fragments assemblés
- Un nombre suffisant de reads est demandé pour assurer un consensus statistiquement significatif
- Erreurs de lecture sont corrigés

Dérivation de séquences Consensus

Dériver un alignement multiple à partir d’alignements par paires des reads

Dériver chaque base de consensus en utilisant des poids de fréquences

Reconstruction des positions des fragments à partir de positions clés dans le génome

Une vue a large échelle de la reconstruction de sections dans la carte physique du génome

Une liste, donnant pour chaque read/clone la position estimée sur le génome est une *solution* au problème de l'assemblage du génome, appelée **carte physique**.

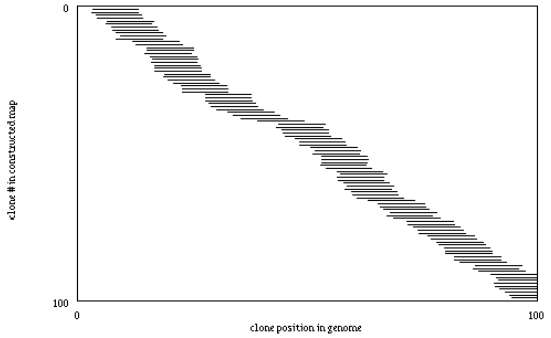
Avec une couverture suffisante la carte complète devient 1 contig. Un tel ordonnancement de clones une fois comparé (dans un diagramme) a la position réelle des clones donne une mesure visuelle de la qualité de la carte. En particulier, une solution correcte aura la propriété que la position des extrémités gauches des clones est monotone décroissante quand y augmente.

a. Des petits erreurs, qui ne changent pas l'ordre des clones, ne peuvent pas être vus par le diagramme.

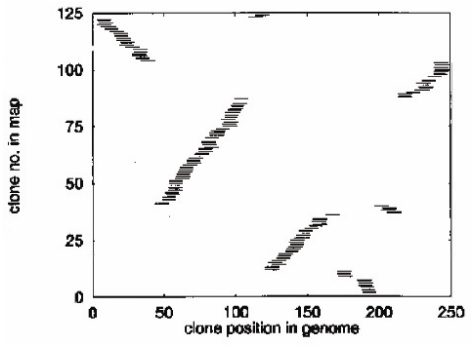
b. Une solution complètement aléatoire correspondra à un positionnement complètement aléatoire des clones

c. Une solution non-aléatoire contenant plusieurs grands erreurs se traduira dans plusieurs contigs, fragmentés et positionnés aléatoirement avec un ordre entre contig qui est à peu près correcte.

Section du génome



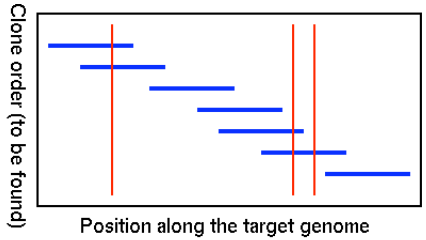
Exemple d'une « section » de carte physique. Les lignes horizontales sont les clones/reads ayant leur coordonnée *x* correspondant à la position sur le génome. La coordonnée *y* correspondent à l'ordre des clones sur la carte reconstruite.



Exemple de cartographie physique pas trop réussie (!). Obtenue avec un bas facteur de couverture ou un bruit des données de sondes. La pluspart des données dans les 8 contig est correcte, mais on observe des inversions de contigs.

Oligos Fingerprinting (Poustka, 1986)

Pour trouver le placement correcte des clones, de l'information sur les chevauchements des clones doit être connue. Dans la technique de l'oligos fingerprint des séquences courtes d'ADN, ou sondes, s'attachent (ou mieux, s'hybrident) à des positions spécifiques sur l'ADN. Les sondes ne sont pas uniques, et en fait elles apparaissent à plusieurs points dans le génome, et typiquement elles hybrident avec 10-50% des clones.



Les clones sont les lignes horizontales. Les occurrences aléatoires d'une seule sonde non-unique sont marquées par des lignes verticales. La sonde apparaît 3 fois sur une section du génome couverte par 7 clones et son vecteur d'occurrence est (1,1,0,0,1,2,0). Si on considère pas le bruit expérimental, l'occurrence des sondes détermine leur hybridation et dans ce cas la, le vecteur d'hybridation d'une sonde est (1,1,0,0,1,1,0).

Le résultat du laboratoire est une matrice binaire *B* représentant l'hybridation, telle que *B(i,j)=1* si la sonde *j* hybride avec le clone *i*.

La *hybridization fingerprint* d'un clone est la ligne dans la matrice d'hybridation correspondant à l'hybridation de toutes les sondes avec le clone. Clones chevauchants sont susceptibles d'avoir des fingerprints similaires, et ce fait est à la base de la reconstruction de la position relative des clones. Dans les scénarios réalistes, du bruit expérimentale peut être présent et il peut affecter les données : on peut avoir des hybridations **faux positives** (hybridations sans occurrence d'une sonde) et **faux négatives** (occurrence des sondes sans hybridation).

Dans la suite on va voir comment les statistiques Bayesiennes peuvent être utilisées pour identifier les clones chevauchants, sous l'hypothèse que un nombre suffisamment grand de sondes soit utilisé.

Modèles probabilistes de construction des cartes

Rappel:

Un processus de Poisson de taux λ est décrit comme suit:

- Une fonction décroissante $N : R_0^+ \rightarrow N$,
  où N(t)= nombre des événements jusqu'au moment t.
- N(0)=0
- Le nombre d'événements dans des intervalles disjoints est indépendant:

$$\Pr(N(t+s)-N(s)=n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!} \quad \text{pour } n=0,1,\ldots \text{ et } s \geq 0$$

λ est un réel positif, égale au nombre d'occurrences d'un événement attendue dans un intervalle de temps donné. Par exemple, si un événement se passe en moyenne chaque 4 minutes, et que l'on est intéressé à un intervalle de 10 minutes, on doit utiliser une distribution de Poisson avec $\lambda = 2.5$.

En conséquence la distribution du nombre des événements dans un intervalle est **stationnaire**, cad dépendant seulement de la longueur de l'intervalle. Le nombre attendu d'événements dans un intervalle de longueur t est donné par $E(N(t))= \lambda t$.

On dénote:

T_n = temps entre les deux événements n-1 et n

$S_0 = 0$

$S_i = \sum_{i=1}^{n} T_i$

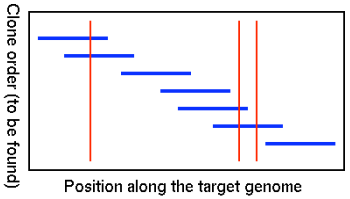
Les temps entre événements dans un processus de Poisson sont des variables aléatoires indépendantes, distribuées exponentiellement avec paramètre λ, cad

$$\Pr(T_i > t) = e^{-\lambda t}$$

Si n (≥ 1) événements se passent dans un processus de Poisson jusqu'au moment t, alors les temps de réalisation des événements $\{S_1 \ldots S_n\}$ sont distribuées uniformément et indépendamment dans [0,t].

Le modèle statistique utilisé pour la cartographie :

1. Les clones sont distribués uniformément et indépendamment sur le génome
2. Les clones ont la même longueur l
3. L'occurrence des sondes sur le génome est modélisée par un processus de Poisson
4. Le taux de Poisson est identique (paramètre λ) et indépendant pour toutes les sondes
5. Le bruit statistique est représenté comme suit:
   - erreurs dus aux faux positifs: un processus de Poisson avec paramètre β
   - erreurs dus aux faux negatifs : ils apparaissent avec probabilité α pour chaque hybridation



vecteur d'occurrence (1,1,0,0,1,2,0)
vecteur d'hybridation (1,1,0,0,1,1,0)

Soit A la matrice des occurrences (0/n) sonde-clone et B la matrice d'hybridation (0/1) sonde-clone.

$$Pr(A_{ij}=k) = \frac{(\lambda l)^k e^{-\lambda l}}{k!}$$

La probabilité que une sonde j apparaît k fois dans un clone i

$Pr(B_{ij}=1|A_{ij})$ = Pr(faux positif) + (1-Pr(faux positif))(1-Pr(faux négatif|A_{ij}))= $(1-e^{-\beta l})+(1-(1-e^{-\beta l}))(1-\alpha^{Aij}) = 1 - e^{-\beta l \alpha}\,\alpha^{Aij}$

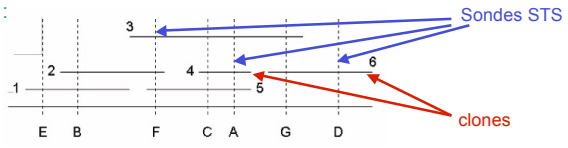
Carte de sondes (« probes ») **uniques**

Une sonde STS (Sequence Tagged Site) ou discriminateur STS, est un filtre qui permet de déterminer de façon unique si une séquence courte d'ADN est une sous-séquence d'une séquence longue. Ce filtre identifie l'existence mais pas la position de la sous-séquence. Si on considère une séquence courte de 200-300nt on sait que la probabilité d'une erreur est suffisamment faible.

Comparer plusieurs sondes STS à plusieurs clones reviens à construire une matrice M où

$$M_{ij} = \begin{cases} 1 & \text{si la sonde j reconnaît une sous-séquence du clone i} \\ 0 & \text{si non} \end{cases}$$

Exemple :



Clone / Probe	A	B	C	D	E	F	G
1	0	1	0	0	1	0	0
2	0	1	0	0	0	1	0
3	1	0	1	0	0	1	1
4	1	0	1	0	0	0	0
5	1	0	1	0	0	1	0
6	0	0	0	1	0	0	1

Matrice STS

Clone / Probe	E	B	F	C	A	G	D
1	1	1	0	0	0	0	0
2	0	1	1	0	0	0	0
3	0	0	1	1	1	1	0
4	0	0	0	1	1	0	0
5	0	0	1	1	1	0	0
6	0	0	0	0	0	1	1

Matrice STS résolue. L'ensemble des matrices STS résolues est équivalent a la solution de l'algorithme PQ-tree.

Problème (carte des sondes uniques)
Entrée : un ensemble d'éléments U (sondes) et une collection de sous-ensembles S = {S_1,S_2,…, S_n}, où $S_i \subset U$ pour chaque i.
Sortie : l'ensemble Π(S) de toutes les permutations sur U où les éléments de S_i sont contigus.

Problème équivalent au problème du réarrangement des colonnes de la matrice STS de telle façon que tous les 1 dans chaque ligne de la matrice soient consécutifs. (La propriété des 1 consécutifs.)

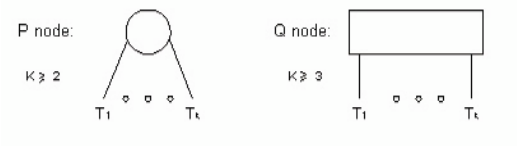
Algorithme pour trouver les permutations : problème bien connu en informatique.

Algorithme linéaire, Booth et Lueker, 1976.

Une représentation explicite de la collection de toutes les permutations résultantes est obtenue avec les **arbres-PQ**.

Arbres-PQ : arbre raciné, orienté

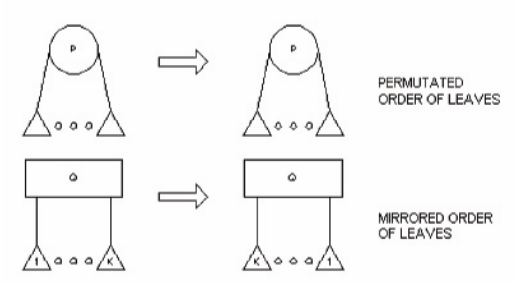
Nœuds terminaux : élément de U
Nœuds intermédiaires : nœud-P et nœud-Q



Un nœud-P représente k sous-ensembles de U (les feuilles de $T_1 \ldots T_k$, pour k>=2), dont chacun est un bloque d'éléments consécutifs, et ayant un ordre de bloques inconnu.

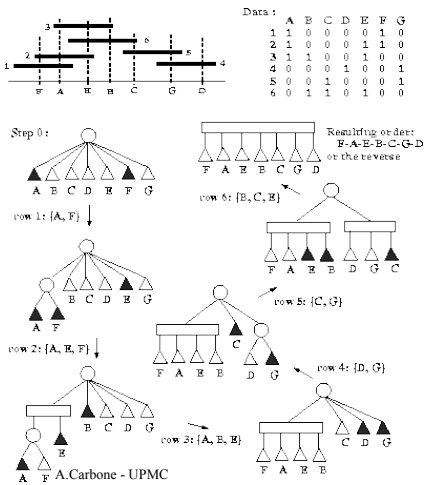
Un nœud-Q ayant sous-nœuds $T_1 \ldots T_k$, pour k>=3, représente le fait que l'ordre des feuilles de $T_1, \ldots T_k$ est fixé, à l'exception d'une renversion complète.

Opérations sur les feuilles permises :



A partir des deux règles précédentes on déduit un ensemble de transformations légales, qui seront utilisées pour re-arranger un arbre-PQ une fois que des nouvelles contraintes (provenant de la matrice STS) sur l'ordre des feuilles sont imposées.

Idée de l'algorithme :

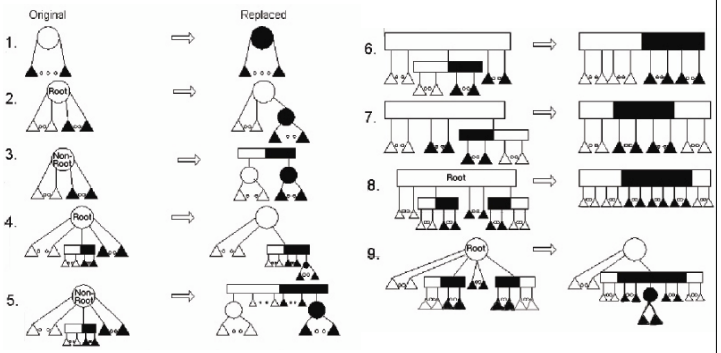


Le processus illustre l'usage d'une structure de données arbre-PQ pour permuter les colonnes (STSs) de la matrice clone-STS.
i=0: ajouter toutes les colonnes STS comme fils d'un noeud-P racine.
étape i: grouper les STS qui touchent le même clone i, cad que les nœuds avec des 1 dans la ligne i doivent devenir des voisins. Si au moins 3 nœuds peuvent être arrangés de façon unique (à renversement près), arranger-les comme fils d'un nœud-Q.

Itérer les étapes jusqu'à quand toutes les lignes ont été considérées.

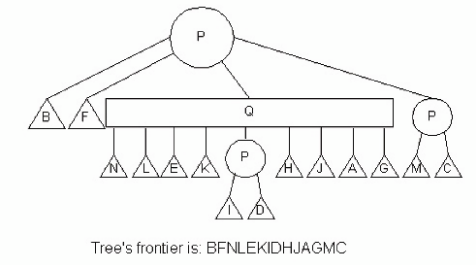
Le dernier arbre-PQ représente tous les possibles ordres. Dans cet exemple, seulement deux ordres sont possibles: FAEBCDG, et son renversement GDCBEAF.

Règles de remplacement



- Pas toutes les combinaisons sont possibles.
- La coloration indique le sous-arbre minimale qui contient les feuilles sur lesquelles la nouvelle contrainte (provenant de la matrice STS) est appliquée.
- Quand on re-arrange l'arbre-PQ, les transformations doivent être appliquées seulement à un sous-arbre colorié pour assurer la légalité de l'arbre.

La frontière d'un arbre-PQ est l'ensemble de toutes les feuilles lu de la gauche vers la droite.



Tree's frontier is: BFNLEKIDHJAGMC

Deux arbres-PQ T et T'sont équivalents s'il existe un ensemble de transformations légales qui permettent de transformer l'un dans l'autre. Dans ce cas on écrit T ≡T'.

L'ensemble de toutes les frontières de T est appelé **Consistant(T) = {Frontière(T') | T ■ T'}**

Théorème (Booth-Lueker1976)
1. Pour tout U et S, il existe un arbre-PQ T tel que Consistant(T)= Π(S).
2. Pour chaque arbre-PQ T, il existe U,S tel que Consistant(T)= Π(S).

Le problème de permuter les sondes pour rejoindre la propriété des 1 consécutifs dans la matrice STS est équivalent à trouver un arbre-PQ représentant Π(S).

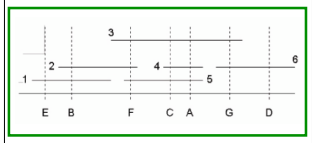
Algorithme Arbre-PQ pour la reconstruction d'une carte de sondes d'ADN

1. Initialiser l'arbre comme une racine ayant nœud-P avec tous les éléments de U comme sous-nœuds (feuilles)
2. Pour chaque i=1…n, Reduce(T,S_i)

Reduce(T,S_i)

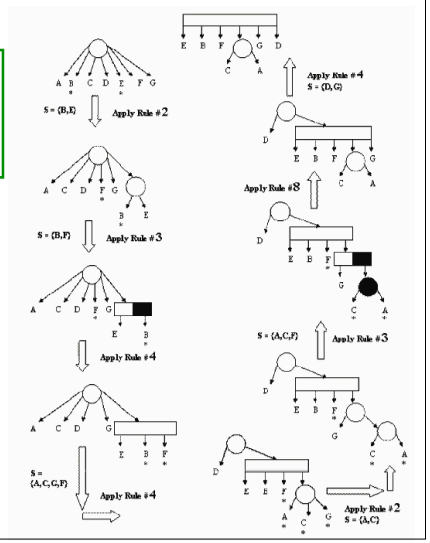
1. Colorier toutes les feuilles de S_i
2. Appliquer une transformation pour remplacer T avec un arbre-PQ équivalent tel que toutes les feuilles coloriées soient consécutives sur la frontière.
3. Identifier le nœud le plus profond Root(T,Si) tel que son sous-arbre contient toutes les feuilles coloriées.
4. Appliquer les règles de remplacement: traverser les noeuds du sous-arbre, « bottom-up » jusqu'à rejoindre Root(T,Si), quand toutes les feuilles coloriées sont adjacentes. Avancer dans le sous-arbre bottom-up, en re-arrangeant les feuilles de chaque nœud visité en respectant les règles légales. Pour un nœud-P vide, et pour tous nœuds-Q (pleins ou vides), aucun changement est nécessaire. S'il n'y a pas de « matching pattern », retourner l'arbre nul (cad, il n'y a pas d'alignement correcte).
5. Enlever le coloriage (toute l'information de S_i est maintenant incluse dans la structure de l'arbre)

Exemple 1:



Clone / Probe	A	B	C	D	E	F	G
1	0	1	0	0	1	0	0
2	0	1	0	0	0	1	0
3	1	0	1	0	0	1	1
4	1	0	1	0	0	0	0
5	1	0	1	0	0	1	0
6	0	0	0	1	0	0	1

Clone / Probe	E	B	F	C	A	G	D
1	1	1	0	0	0	0	0
2	0	1	1	0	0	0	0
3	0	0	1	1	1	1	0
4	0	0	0	1	1	0	0
5	0	0	1	1	1	0	0
6	0	0	0	0	0	1	1



Applications réelles

Bruit

L'algorithme ne considère pas le bruit des données: **étant données des erreurs de mesure, la matrice d'entrée a d'habitude des 1 en plus et des 1 manquant.** Dans ce cas, l'arbre-PQ résultant ne produira pas la meilleure solution (cad la solution ayant l'erreur minimum), mais plutôt une solution arbitraire qui dépend de l'ordre choisi des clones.

Sondes non-uniques : avantages

La cartographie physique utilisant des sondes non-uniques partage avec la technique plus populaire utilisant des probes uniques une manipulation simple en laboratoire, et aussi la possibilité d'analyser plusieurs clones en parallèle. Elle est aussi insensible a des séquences repetées dans le gel.

Les avantages par rapport à la technique des sondes uniques sont:
(1) La **génération des sondes est facile** et beaucoup moins chère,
(2) Le nombre de sondes est indépendant de la taille du génome
(3) Le **contenu informatif (en sense théorétique) de chaque expérience est plus élevé**. Cela permet la conception d'algorithmes robustes aux erreurs expérimentaux et une interprétation probabiliste des données d'hybridation.

Avantages évidents des sondes non-uniques mais manque d'outils d'analyse:

La manque d'algorithmes pour l'analyse des données issue des expérience avec sondes non-uniques, a empêché les laboratoire d'utiliser la technique pour des années.

Guy Mayraz and Ron Shamir 1998 ont proposé une solution à ce problème, basée sur une approche statistique solide. L'algorithme pour ce problème a été testé avec des simulations sur des séquences d'ADN synthétiques et réelles.

Voir aussi d'autres travaux par **R.Karp et al.** (1998) et **B.Mishra et al.** (1998).

Construction de carte physiques à partir de sondes non-uniques et avec bruit

Problème (Cartographie avec sondes non-uniques et bruit)

Entrée:
1. matrice d'hybridation sonde-clone B; elle contient des faux positifs (B_{ij}=1 quand A_{ij}=0) et des faux négatifs (B_{ij} =0 quand A_{ij}>0)
2. Taille du génome
3. Longueur des sondes (en nombre de 200-500) ayant une taille typique de 8-10bp
4. Longueur des clones
5. Estimations de α, β représentant le bruit statistique.

Sortie: trouver la position relative des clones.

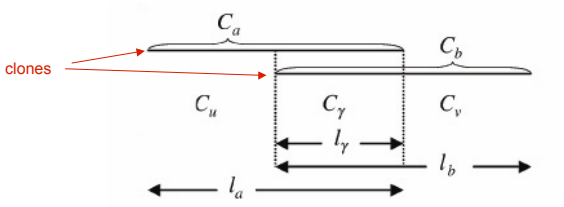
L'approche Bayesienne
L'approche Bayesienne se propose de calculer la **distribution de probabilité des données hybridés comme fonction de l'ordre des clones**. A partir de probabilités initiales sur des ordonnancements possibles des clones, la loi Bayesienne permet de calculer les probabilités des différents ordonnancements.

L'algorithme
Un calcul bayesien exacte est infaisable quand on doit le faire sur plusieurs clones. Le calcul pour des paires de clones peut être par contre réalisé. L'algorithme donne une très bonne **estimation des chevauchements entre paires de clones**, sous forme de distribution de probabilité des possibles chevauchements.

L'estimation des chevauchements des paires de clones a été employée en suite pour définir un score de chevauchement pour les paires de contigs, chacun formé par un nombre possiblement large de clones. Encore une fois, le score de paires de contigs ne donne pas une probabilité exacte, mais il donne quand même une très bonne approximation (d'après une analyse mathématique approfondie, Shamir 1998).

L'algorithme sort la position physique des clones et pas seulement leur ordonnancement.

Score pour une paire chevauchante



On peux définir un vecteur de probabilités pour chaque longueur l_γ du chevauchement:

$$\Pr(l_\gamma = t_0 | \mathbf{B}_a, \mathbf{B}_b) = \frac{\Pr(\mathbf{B}_a, \mathbf{B}_b \mid l_\gamma = t_0)\ \Pr(l_\gamma = t_0)}{\sum_{t=0}^{la} \Pr(\mathbf{B}_a, \mathbf{B}_b \mid l_\gamma = t)\ \Pr(l_\gamma = t)}$$

où $\mathbf{B}_a$,$\mathbf{B}_b$ sont deux vecteurs d'hybridation associés aux clones C_a, C_b.

L'algorithme de construction:

1. Initialiser une collection de contigs, chacun étant un ensemble de clones. Au départ, la collection contient pour chaque clone, un contig constitué par ce clone.

2. Calculer pour toutes paires de contigs et pour leur deux possibles orientations leur vecteur de probabilités de placement relatif. Les résultats sont stockés dans une table.

3. Trouver pour toutes les paires de contigs et pour leur deux possibles orientations relatives le meilleur placement relatif, et sa probabilité.

4. Jusqu'à quand on a plus qu'un contig:
   a. Trouver deux contigs a et b ayant une orientation relative et un placement que rejoint la probabilité la plus élevée
   b. Fusionner b dans a
   c. Changer la table calculée a l'étape 2 pour tenir en compte le dernier fusionnement. Seulement les lignes de la table concernant (a,c) ont besoin d'être changée. Le changement est une simple combinaison des entrées précédentes pour (a,c) et (b,c)
   d. Trouver, pour les paires de contig affectés par le fusionnement, leur nouveau meilleur placement.

Complexité: $O(m^2(L+d+n))$

où L = longueur du génome
d = longueur des clones
m = nombre de clones
n = nombre de sondes

Comme le nombre m de clones peut être important, « m^2 » peut induire une complexité trop grande. Pour cette raison, il est possible d'améliorer l'algorithme en identifiant des partitions des clones dans des sous-groupes ayant une forte probabilité que chaque clone dans l'ensemble chevauche les autres membre de l'ensemble. L'algorithme sera alors appliqué avant tout a chaque sous-ensemble et le nombre de contigs sera minimisé.

Regarder dans les détails, les mathématiques de la construction décrites dans l'article de Mayraz et Shamir, 1999.

Metagénomique: reconstruction des génomes et défis algorithmiques

Métagénomique = étude des métagénomes
ou un métagénome est l'ensemble des séquences d'ADN extraites de communautés multi-espèces prélevées dans l'environnement.

Ces communautés sont constituées d'organismes non cultivables. Il a été estimé que seul 1% des procaryotes de la plupart des environnements peuvent être cultivés nécessitant pour la plupart des organismes que le clonage soit effectué directement sur les échantillons prélevés dans l'environnement.

Le problème d'assemblage des fragments d'ADN

En métagénomique le problème est difficile puisqu'il est nécessaire de pouvoir associer le fragment de séquence à l'espèce duquel il provient.

Les programmes d'assemblage comme Phred/Phrap et Sequencher ont été mis au point pour combiner des séquences issues d'un même génome et sont fondés sur des hypothèses fausses lorsqu'elles sont appliquées à un métagénome.

Exemple: une différence de base entre 2 séquences est considérée comme une erreur de séquençage par ces programmes, alors que cela pourrait être le signe d'une différence d'origine en métagénomique.

Le grand nombre de génomes viraux (jusqu'à plusieurs millions) prélevés dans l'environnement ainsi que les répétitions de séquences de type transposons rendent la discrimination entre les organismes et l'assemblage des séquences d'autant plus difficile.

Ces problèmes nécessitent des approches algorithmiques fines, des fragments d'ADN plus longs, ainsi que des taux de couverture plus importants.

Récemment, la différence de nombre de répétitions de tétranucléotides entre génomes a pu être utilisée pour discriminer ces génomes, mais cette approche nécessite un échantillon de faible complexité et une répartition équitable des génomes.

Le polymorphisme des séquences nucléotidiques, la duplication des gènes et le transfert horizontal de gène sont tous des obstacles à la fiabilité de l'assemblage des génomes.

Références bibliographiques

Jones, Pevzner, *An introduction to bioinformatics algorithms*, MIT Press, 2004

Booth, Lueker, Testing for the consecutive ones property, interval graphs and planarity using PQ-tree algorihms. *J.Comput.Sys.Sci.* 13:335-379, 1976.

Fulkerson, Gross, Incidence matrices and interval graphs. *Pacific Journal Math*, 15:835-855, 1965.

Lander, Waterman, Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, 2:231-239, 1988.

Mayraz, Shamir, Construction of physical maps from oligonucleotide fingerprints data, In *Proceedings of the Third Annual International Conference on Computational Molecular Biology* (RECOMB '99). Aussi dans *Journal of Computational Biology* , 6 (2) 237--252 (1999).

Poustka, Pohl, Barlow, Zehetner, Craig, Michiels, Ehrich, Frischauf, Lehrach. Molecular approaches to mammalian genetics. *Cold Spring Harbor Symp. Quant. Biol.* 51:131-139, 1986.

Baussand, Carbone, Métagénomique bactérienne et virale - Nouvelles définitions d'espace microbien et nouveaux défis algorithmiques, *Technique et science informatiques,* Hermes, 2007, a paraître.