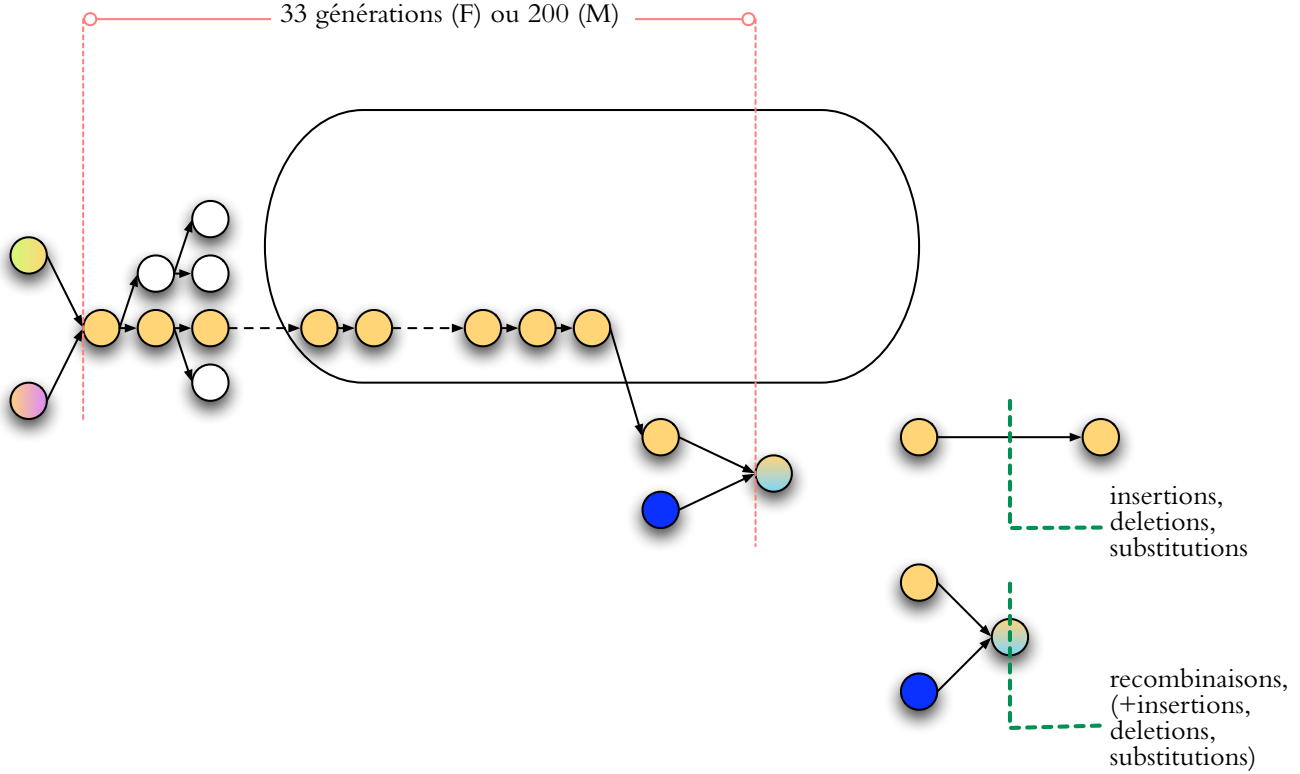
Alignement de séquences

ou comment comparer des Livres sur le même sujet

Mutations



les mutations s'accumulent pendant les années → divergence de séquences d'origin commun

grand divergence ⇒ plus de temps et/ou moins de sélection négative

Distance entre deux mots

Distance d'édition : nombre d'opérations pour transformer un mot à un autre.

Opérations : sur caractères

- insertion (I *insert*)
- suppression (D *delete*)
- substitution (S *substitute*)
- identité (M *match*)

Calcul de distance

But : calculer le nombre *minimal* d'opérations pour la transformation.

Minimal : autant de M (identité) que possible.

Exemples : docment-document, docment-dorment, docment-doucement, docment-moments

On peut avoir plusieurs suites de la même taille minimale : abbc-axc

Calcul de distance 2

Idée : calculer les distances entre les préfixes successivement

Nos notions formelles :
- alphabet fini Σ (p.e. $\Sigma = \{\mathtt{A}, \ldots, \mathtt{z}\}$ ou $\Sigma = \{\mathtt{T}, \mathtt{G}, \mathtt{A}, \mathtt{C}\}$)
- mot ou séquence : une suite de caractères de Σ

Soit S une séquence.
- taille de S, denotée par $|S|$ = nombre de caractères
- caractère en position i : $S[i]$
- sous-mot $S[i..j]$: le mot formé par les caractères $S[i]$, $S[i+1]$,..., $S[j]$
- préfixe : sous-mot de forme $S[1..i]$.

Calcul de distance 3

Def. Distance d'édition entre deux séquences S_1 et S_2 : nombre minimal de I, D, S dans une suite d'opérations qui transforme S_1 en S_2.

Thm. La distance d'édition est une fonction symétrique.
Preuve. I $\Leftrightarrow$ D.

Thm. La distance d'édition satisfait l'inégalité du triangle

Un fait intéressant :) la distance d'édition est parfois appellée
distance de Levenshtein

Calcul de distance 4

Def. Soit S et T deux séquences. On définit $D(i,j)$ par

$$D(i,j) = \begin{cases} \text{distance entre } S[1..i] \text{ et } T[1..j] & \text{si } i > 0 \text{ et } j > 0, \\ i & \text{si } j = 0 \\ j & \text{si } i = 0. \end{cases}$$

Donc $D(i,j)$ est la distance entre les deux préfixes de tailles i et j.

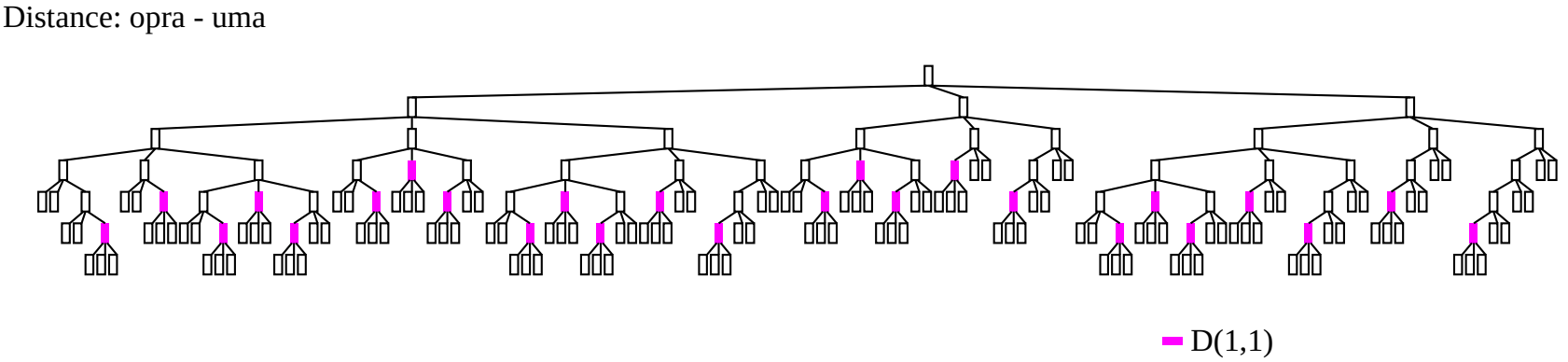
Thm. Si $i, j > 0$, on a

$$D(i,j) = \min\Big\{ D(i-1,j) + 1, \\ D(i,j-1) + 1, \\ D(i-1,j-1) + \{S[i] \neq T[j]\} \Big\}.$$

Calcul de distance 5

Preuve. 1. La dernière opération pour achever $D(i, j)$ doit être I, D, ou S/M.
2. Tous les trois sont possibles.

Donc on peut calculer $D(|S|, |T|)$ par récursion. . . pas une bonne idée. (La même valeur sera calculée plusieurs fois.)



Quand même on n'a que $(1 + |S|) \times (1 + |T|)$ appels récursifs possibles.

Au lieu d'explorer l'arbre de récursions en descendant, il est mieux de le faire de manière ascendante.

Tableau de calculs

Les cases contiennent les $D(i, j)$.

Parcours : ligne par ligne.

Exemple : `AAAC` $\longrightarrow$ `AGC`

Temps de calcul : $O(mn)$ (on remplit chaque case du tableau en un temps constant : trois comparaisons et deux ou trois additions)

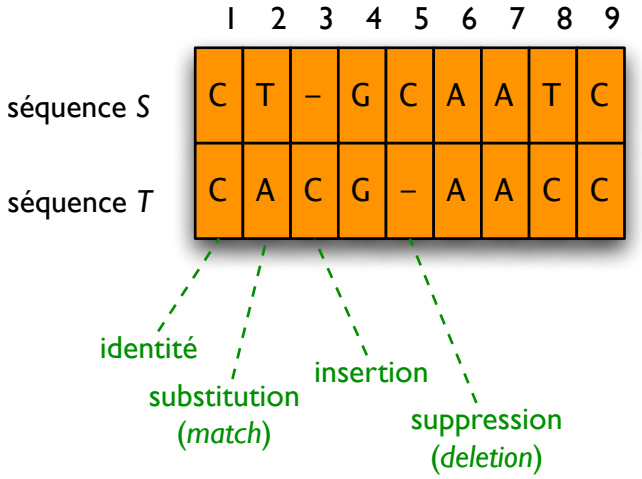
Comment trouver une suite d'opérations qui correspond à $D(i, j)$? Enregistrer dans chaque case la direction du $\min$ de la recurrence.

Alignement

Déf. alignement=vecteur d'appariements

Types d'appariements :

- (a, a) : occurence (*match*)
- (a, b) : erreur (*mismatch/substitution*)
- $(a, -)$ et $(-, a)$: indel



Alignement pondéré

Tableau $\mathbf{C}$ de pondération d'appariements

	A	C	G	T	–
A	1	−3	−3	−3	−5
C	−3	1	−3	−3	−5
G	−3	−3	1	−3	−5
T	−3	−3	−3	1	−5
–	−5	−5	−5	−5	X

ou :

	A	C	G	T	–
A	91	−114	−31	−123	−300
C	−114	100	−125	−31	−300
G	−31	−125	100	−114	−300
T	−123	−31	−114	91	−300
–	−300	−300	−300	−300	

Score ou valeur de l'alignement : somme des valeurs des appariements

Problème : trouver l'alignement avec le score maximal
PD pour maximiser la similarité

Chiaromonte et al. PSB 2002.

Génération d'une matrice de pondération

Modèle probabiliste pour une séquence : caractères aléatoires iid (indépendants et identiquement distribués) : $\mathbb{P}\{S[i] = \sigma\} = \pi_\sigma$

P.e., $\pi_{\mathtt{A}} = \pi_{\mathtt{T}} = 32\%$, $\pi_{\mathtt{C}} = \pi_{\mathtt{G}} = 18\%$: taux de $(\mathtt{G} + \mathtt{C})$ à 36%.

Modèle probabiliste pour évolution $S \to T$: substitutions aléatoires indépendantes : $\mathbb{P}\Big\{T[i] = \sigma' \;\Big|\; S[i] = \sigma\Big\} = p_{\sigma \to \sigma'}$ (pas de trous !)

Matrice de subsitutions : $\mathbf{M} = \Big[p_{\sigma \to \sigma'}\Big]_{\sigma, \sigma' \in \Sigma}$

Matrice de pondération 2

Probabilité d'un «vrai» alignement : $P_1 = \mathbb{P}\{S, T\}$

Probabilité d'un alignement au hasard : $P_0 = \mathbb{P}\{S\}\mathbb{P}\{T\}$

$$P_1 = \prod_{i=1}^{n} \mathbb{P}\{S[i], T[i]\} = \prod_{i=1}^{n} \pi_{S[i]} p_{S[i] \to T[i]};$$

$$P_0 = \prod_{i=1}^{n} \mathbb{P}\{S[i]\}\mathbb{P}\{T[i]\} = \prod_{i=1}^{n} \pi_{S[i]} \pi_{T[i]}.$$

Matrice de pondération 3

Rapport de P_0 et P_1 : LODS (logarithmes des chances) $\log \frac{P_1}{P_0}$.

On a

$$\mathrm{LODS} = \sum_{i=1}^{n} \log \frac{\pi_{S[i]} p_{S[i] \to T[i]}}{\pi_{S[i]} \pi_{T[i]}} = \sum_{i=1}^{n} \log \frac{p_{S[i] \to T[i]}}{\pi_{T[i]}}.$$

Score de substitution $\sigma \to \sigma'$:

$$\mathbf{C}\begin{bmatrix} \sigma \\ \sigma' \end{bmatrix} = \left\lceil \alpha \log \frac{p_{\sigma \to \sigma'}}{\pi_{\sigma'}} \right\rceil,$$

où α est une facteur d'échelle (notez qu'on utilise le plafond pour valeurs entières)

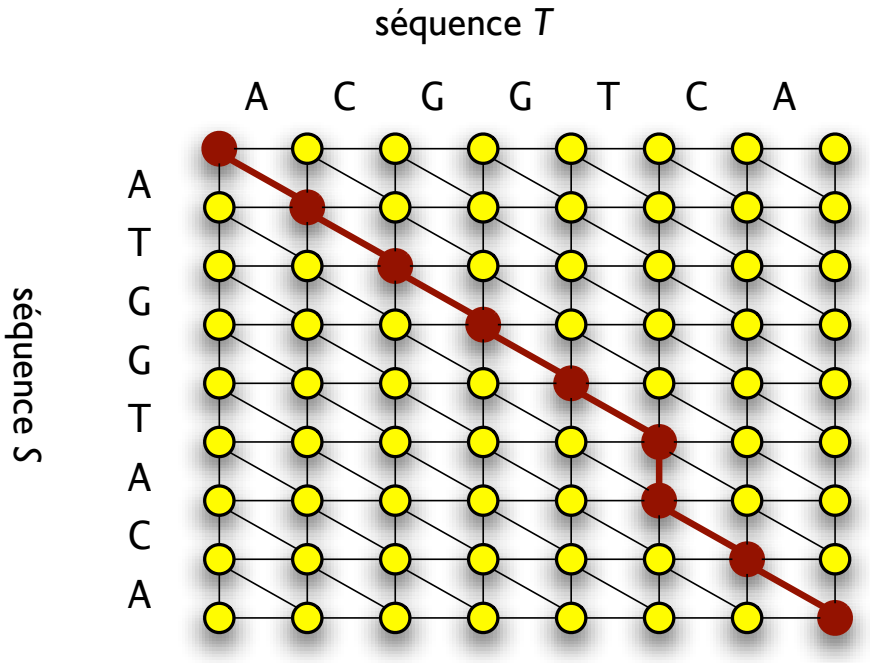
Matrice de pondération 4

Problème : comment estimer π_σ et $p_{\sigma \to \sigma'}$?

Solution : prendre de «bons alignements»

Exemples : BLOSUMnn (nn=45,62,80 pour nn% d'identité, protéines) ;
PAMxxx (xxx=100,250,. . . , mesure divergence, protéines) ;
Chiaromonte (pour ADN)

Graphe d'édition



ACGGT–CA
ATGGTACA

Graphe d'édition 2

pondération des arêtes :

$$\mathsf{poids}(v_{i-1,j-1} \to v_{i,j}) = \mathbf{C}\begin{bmatrix} S[i] \\ T[j] \end{bmatrix};$$

$$\mathsf{poids}(v_{i-1,j} \to v_{i,j}) = \mathbf{C}\begin{bmatrix} S[i] \\ - \end{bmatrix};$$

$$\mathsf{poids}(v_{i,j-1} \to v_{i,j}) = \mathbf{C}\begin{bmatrix} - \\ T[j] \end{bmatrix}.$$

alignement global : trouver le chemin de $v_{0,0}$ à $v_{|S|,|T|}$ avec poids maximal.

Alignement : variation

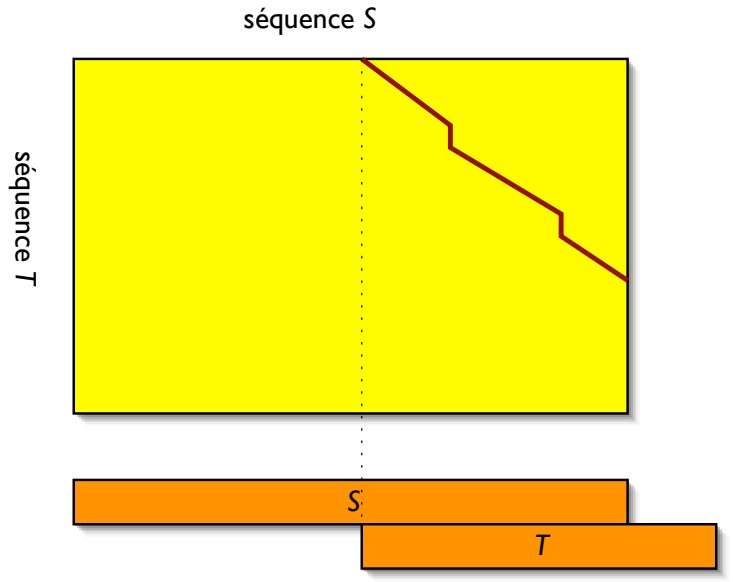
Trouver la région de similarité maximale entre deux séquences —
alignement local

Dans le graphe d'édition : trouver le chemin de $v_{i,j}$ à $v_{i',j'}$ avec poids maximal pour $i' \geq i, j' \geq j$ quelconques

Les noms :
Needleman-Wunsch : problème de l'alignement global
Smith-Waterman : algorithme de l'alignement local

Alignement : chevauchements

Trouver le meilleur chevauchement entre un suffixe (de S) et un préfixe (de T).



Trous

Pénalisation d'un trou par sa taille :

- quelconque : $\delta(\mathsf{longueur})$
- constante
- linéaire : $\delta(\ell) = \delta_{\mathsf{ouvrir}} + \ell\delta_{\mathsf{cont}}$ (avant on avait le cas spécial $\delta_{\mathsf{ouvrir}} = 0$)

Récurrence pour pondération «quelconque» :

$$A(i,j) = \max\Big\{A(i-1,j-1) + \mathbf{C}\begin{bmatrix} S[i] \\ T[j] \end{bmatrix}, \max_{\ell=1,\ldots,i}\{A(i-\ell,j) + \delta(\ell)\}, \max_{\ell=1,\ldots,j}\{A(i,j-\ell) + \delta(\ell)\}\Big\}.$$

Cas de base : $A(i,0) = \delta(i)$; $A(0,j) = \delta(j)$.

Temps de calcul : $O(nm^2 + mn^2)$ pour $|S| = n$, $|T| = m$.

Pondération — exemple

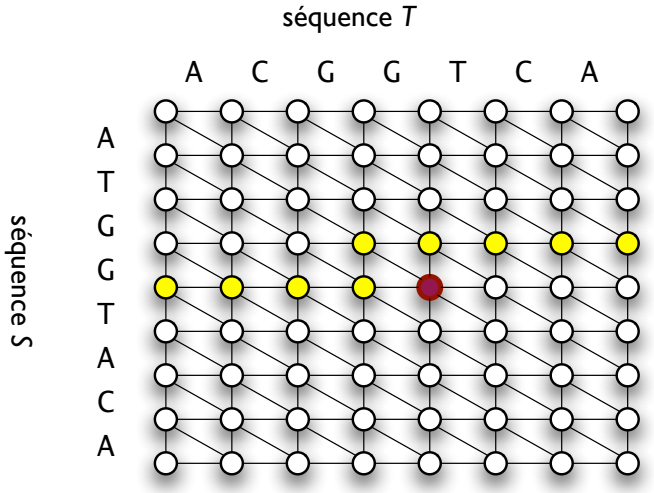
Trous longs (disons $\ell \geq 5$) en S avec pénalisation constante δ_{long}, trous courts avec pénalisation quelconque $\delta(\ell)$; trous en T avec pénalisation simple $\delta'\ell$.

Définir graphe d'alignement avec deux genres de sommets : $v_{i,j}$ pour alignements de préfixes qui finissent pas par trous longs et $t_{i,j}$ pour ceux qui finissent par trous longs.

Arêtes : $v_{i,j} \searrow v_{i+1,j+1}$ ponderée par $\mathbf{C}\begin{bmatrix} S[i] \\ T[j] \end{bmatrix}$, $v_{i,j} \downarrow v_{i+k,j}$ pour $k = 1, \ldots, 4$ ponderée par $\delta(k)$, $v_{i,j} \rightarrow v_{i,j+1}$ ponderée par δ', $t_{i,j} \rightarrow v_{i,j}$ ponderée par 0, $v_{i,j} \downarrow t_{i+5,j}$ ponderée par δ_{long}, $t_{i,j} \downarrow t_{i+1,j}$ ponderée par 0.

Calcul en espace linéaire

Calcul de score $V(i,j)$ rangée par rangée de gauche à droite (trou de taille ℓ pondéré par $\delta\ell$)



On n'a besoin que de $V(i-1,j')\colon j' \geq j$ et de $V(i,j'')\colon j'' \leq j$.

Espace linéaire 2

Algo ALI-LINESPACE
Entrée : séquences S, T avec $n = |S|$ et $m = |T|$; matrice de coûts $\mathbf{C}$

L1 $U[0] \leftarrow 0$; **for** $j \leftarrow 1..m$ **do** $U[j] \leftarrow U[j-1] + \mathbf{C}\left[\begin{smallmatrix} - \\ T[j] \end{smallmatrix}\right]$

L2 **for** $i \leftarrow 1..n$ **do**

L3 $\quad x \leftarrow U[0]$; $U[0] \leftarrow U[0] + \mathbf{C}\left[\begin{smallmatrix} S[i] \\ - \end{smallmatrix}\right]$

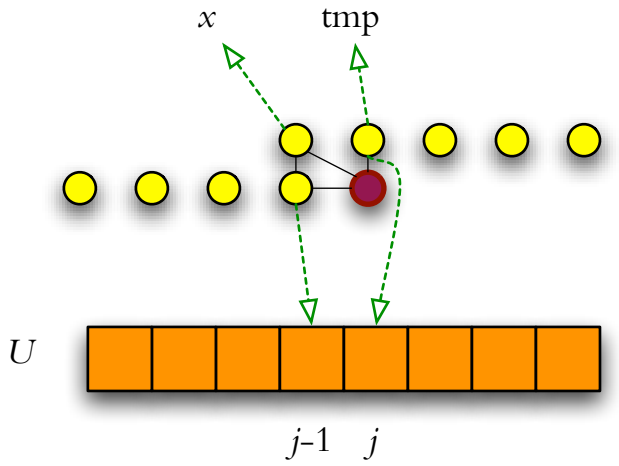
L4 $\quad$ **for** $j \leftarrow 1..m$ **do**

L5 $\quad\quad$ tmp $\leftarrow U[j]$

L6 $\quad\quad U[j] \leftarrow \min\left\{U[j-1] + \mathbf{C}\left[\begin{smallmatrix} S[i] \\ - \end{smallmatrix}\right], x + \mathbf{C}\left[\begin{smallmatrix} S[i] \\ T[j] \end{smallmatrix}\right], U[j] + \mathbf{C}\left[\begin{smallmatrix} - \\ T[j] \end{smallmatrix}\right]\right\}$

L7 $\quad\quad x \leftarrow$ tmp

Affectations en Ligne L6



Espace linéaire 3

Mais comment trouver le meilleur alignement (et pas seulement son score) en espace linéaire ?

Idée de clé : trouver le noeud en ligne i du graphe par lequel le chemin du meilleur alignement doit passer

1. calculer $\forall j : V_{\mathsf{pre}}(i, j)$, le score du meilleur alignment de $S[1..i]$ et $T[1..j]$ ($V_{\mathsf{pre}}(i, j) = U[j]$ en ALI-LINESPACE si appellé avec $S[1..i]$ et T)

2. calculer $\forall j : V_{\mathsf{suf}}(i, j)$, le score du meilleur alignement de $S[i + 1..n]$ et $T[j + 1..m]$.

3. le meilleur alignement doit croiser rangée i en colonne j^* qui maximise $V(i, j^*) = V_{\mathsf{pre}}(i, j^*) + V_{\mathsf{suf}}(i, j^*)$

On peut calculer V^* en espace linéaire avec un algorithme similaire à ALI-LINESPACE

Espace linéaire 4 — Hirschberg

Algo ALI-HIRSCHBERG
Entrée : séquences S, T avec $n = |S|$ et $m = |T|$; matrice de coûts $\mathbf{C}$

H1 **if** $m = 0$ **then return** alignement $\begin{bmatrix} S[1] & \dots & S[n] \\ - & \dots & - \end{bmatrix}$
H2 **if** $n = 1$ **then**
H3 trouver $j \in 1, \dots, m$ t.q. $\mathbf{C}\begin{bmatrix} S[1] \\ T[j] \end{bmatrix}$ is maximal
H4 **return** alignement $\begin{bmatrix} - & \dots & - & S[1] & - & \dots & - \\ T[1] & \dots & T[j-1] & T[j] & T[j+1] & \dots & T[m] \end{bmatrix}$
H5 soit $i \leftarrow \lfloor n/2 \rfloor$
H6 calculer $U[j] = V_{\mathsf{pre}}(i, j)$ et $W[j] = V_{\mathsf{suf}}(i, j)$ pour tout $j = 0, \dots, m$
H7 soit $M \leftarrow \mathsf{max}_{j=0,\dots,m}\{U[j] + V[j]\}$
H8 soit $j^* \leftarrow \mathsf{min}\{j : U[j] + W[j] = M\}$
H9 calculer l'alignement $A_1 \leftarrow$ ALI-HIRSCHBERG$(S[1..i], T[1..j^*], \mathbf{C})$
H10 calculer l'alignement $A_2 \leftarrow$ ALI-HIRSCHBERG$(S[i+1..n], T[j^*+1..m], \mathbf{C})$
H11 **return** la concaténation de $A_1 \cdot A_2$

(En lignes H9–H10, $T[1..0] = \varepsilon$ et $T[m+1..m] = \varepsilon$)

Hirschberg, *Comm. ACM*, 18 : 341

Alignement plus rapide

1. méthodes heuristiques : hachage, arbres de suffixe, PD limitée (taille totale de trous bornée)

2. PD éparse (pour sous-séquence commune ou *chaînage* en alignement global heuristique)

Hachage

Deux séquences S, T

Fonction de hachage : $h : \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k \mapsto \mathcal{H}$

hit : (i, j) avec $h(S[i..i+k-1]) = h(T[j..j+k-1])$

Technique : listes $\mathsf{Occ}(u)$ de positions où $h^{-1}(u)$ apparaît

1. **pour** $i \leftarrow 1, \ldots, |S| - k + 1$ **faire**
2. $\quad \mathsf{clé} \leftarrow h(S[i..i+k-1])$
3. $\quad$ ajouter i à la fin de la liste $\mathsf{Occ}(\mathsf{clé})$
4. **pour** $j \leftarrow 1, \ldots, |T| - k + 1$ **faire**
5. $\quad \mathsf{clé} \leftarrow h(T[j..j+k-1])$
6. $\quad$ traitement [extension ?] des *hits* $(i, j) : i \in \mathsf{Occ}(\mathsf{clé})$

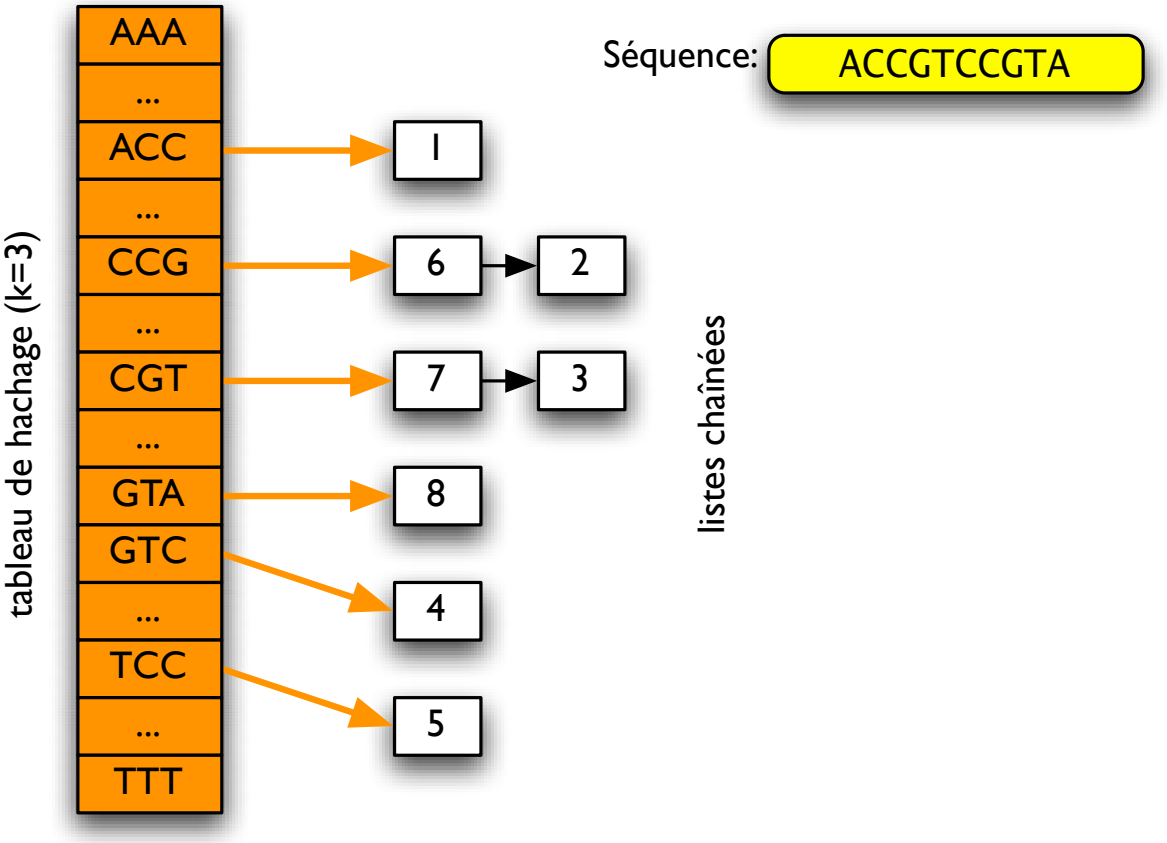
Hachage — structure de données

Trouver les clés partagés : stocker les occurrences (Occ) de tous les clés de S en un tableau de hachage

Implantation facile en Java : on peut utiliser les sous-mots w comme clés directement, `Hashtable` calcule des clés de hachage automatiquement, liste chaînée pour chaque $\mathsf{Occ}(w)$

(utilise plus de mémoire que nécessaire...)

Tableau de k-mers



Ma, Tromp et Li. *Bioinformatics* 18 :440.

Implantation

1. Encodage des k-mers en $2k$ bits :
$\texttt{A} \rightarrow \texttt{00}, \texttt{C} \rightarrow \texttt{01}, \texttt{G} \rightarrow \texttt{10}, \texttt{T} \rightarrow \texttt{11}$.

Java `int` : 32 bits ($k \leq 16$) ; `long` : 64 bits ($k \leq 32$).

2. Encodage des listes chaînées :
chaque position de la séquence n'apparaît qu'une fois !
Définir un tableau `int[] successeur` où `successeur`$[i]$ donne la position qui suit i dans une des listes chaînées ou égale à -1 si i est la dernier objet dans une liste.
Tête de chaque liste est trouvée par un tableau `int[] tete` où `tete`$[i]$ donne la première position ou le k-mer encodé par i se trouve.

Mémoire : $(4^k + |S|)$ fois taille de `int` (4 octets).

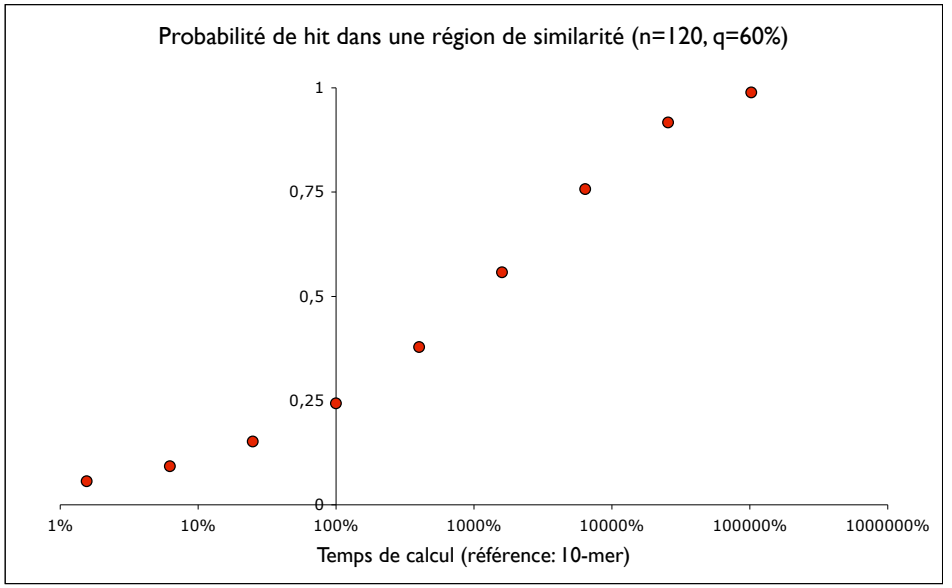
Implantation — Java

```
HT1  int[] tete=new int[1<<(2*k)];
HT2  int[] successeur=new int[S.length()-k+1];
HT3  pour tout i, tete[i] ← -1
HT4  for (int i=0; i<S.length()-k+1; i++){
HT5      calcul de l'encodage w pour le sous-mot S[i..i+k-1];
HT6      successeur[i]=tete[w];
HT7      tete[w]=i;
HT8  }
HT9  for (int j=0; j<T.length()-k+1; j++){
HT10     calcul de l'encodage w pour le sous-mot T[j..j+k-1];
HT11     int i=tete[w];
HT12     while (i != -1){
HT13         extension du hit (i,j)
HT14         i=successeur[i];
HT15     }
HT16 }
```

Hachage — performance

Spécificité : mesurée par nombre de *hits* entre deux séquences sans homologies (p.e., aléatoires)

Sensibilité : mesurée par la probabilité de *hit* dans une région de homologie



Hachage — nombre de *hits*

Modèle : S aléatoire, avec nucléotides iid selon p ; T aléatoire, avec nucléotides iid selon q : $\mathbb{P}\{S[i] = c\} = p_c$ et $\mathbb{P}\{T[j] = c'\} = q_{c'}$.

Fonction de hachage : identité $h(u) = u$, $\mathcal{H} = \Sigma^k$ ($\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$)

Thm. Soit $\beta = \sum_{c \in \Sigma} p_c q_c$. Alors le nombre de *hits* en espérance est $st\beta^k$ où $s = |S| - k + 1$ et $t = |T| - k + 1$.

Détour — nombre de hits

L'espérance de nombre de hits est la même pour tous les sous-mots $w \in \Sigma^k$. Est-ce que la distribution est la même aussi ? Non !

Exemple : «pas tous les mots sont créés égaux»

Exemple : T est une séquence de longueur n «au hasard»
au hasard : chaque caractère de T est $\mathtt{0}$ ou $\mathtt{1}$ avec probabilités $\frac{1}{2}$-$\frac{1}{2}$.

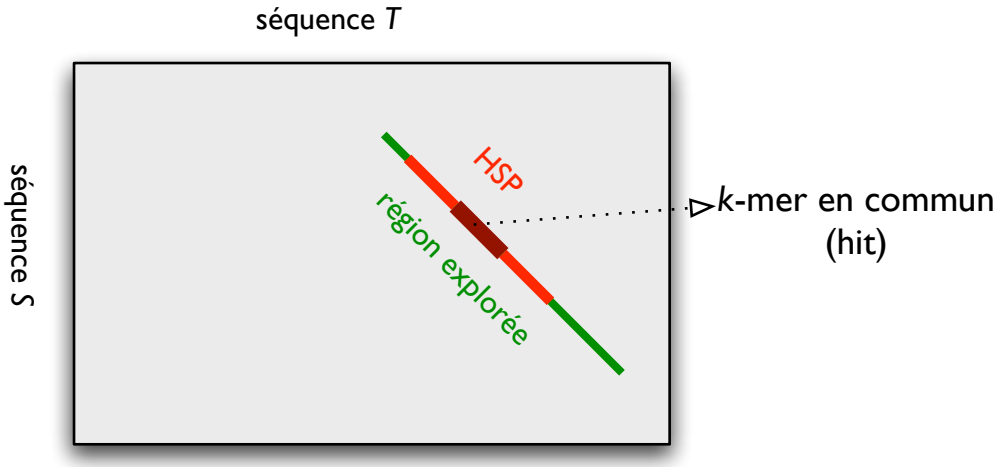
Quelle est la probabilité de voir $w = \mathtt{00}$ ou $w = \mathtt{01}$?

Extension d'un hit

Techniques :

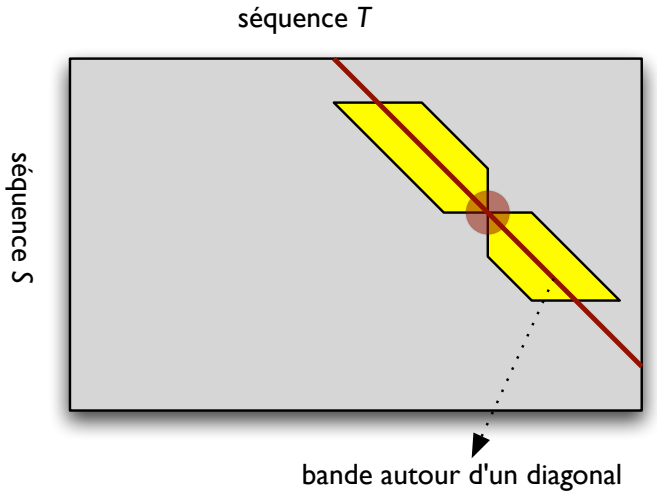
- extension rapide sur une diagonale
- X-drop
- programmation dynamique dans une bande

Extension rapide



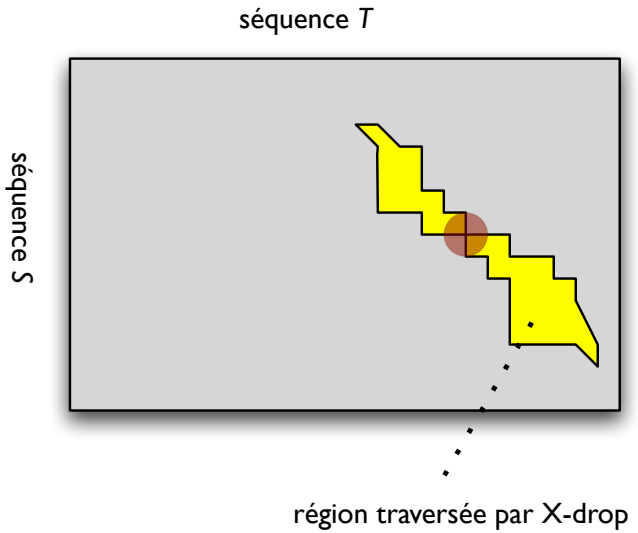
Rester sur la même diagonale ; explorer jusqu'à ce que le score devienne 0, prendre le meilleur segment (*high-scoring segment pair*, HSP)

PD dans une bande



Bande de $\pm d$ sommets proches de la diagonale D : $\{v_{i,j} : |(i-j) - D| \leq d\}$

X-drop



à partir d'une case initiale, explorer vers $v_{0,0}$ et $v_{|S|,|T|}$; arrêter si le score tombe par X

(exploration de toute une région ou quelques [même 1] diagonales)

Altschul et al, *Nucleic Acids Res.* 25 : 3389.

Détails : extension rapide [vers sud-est]

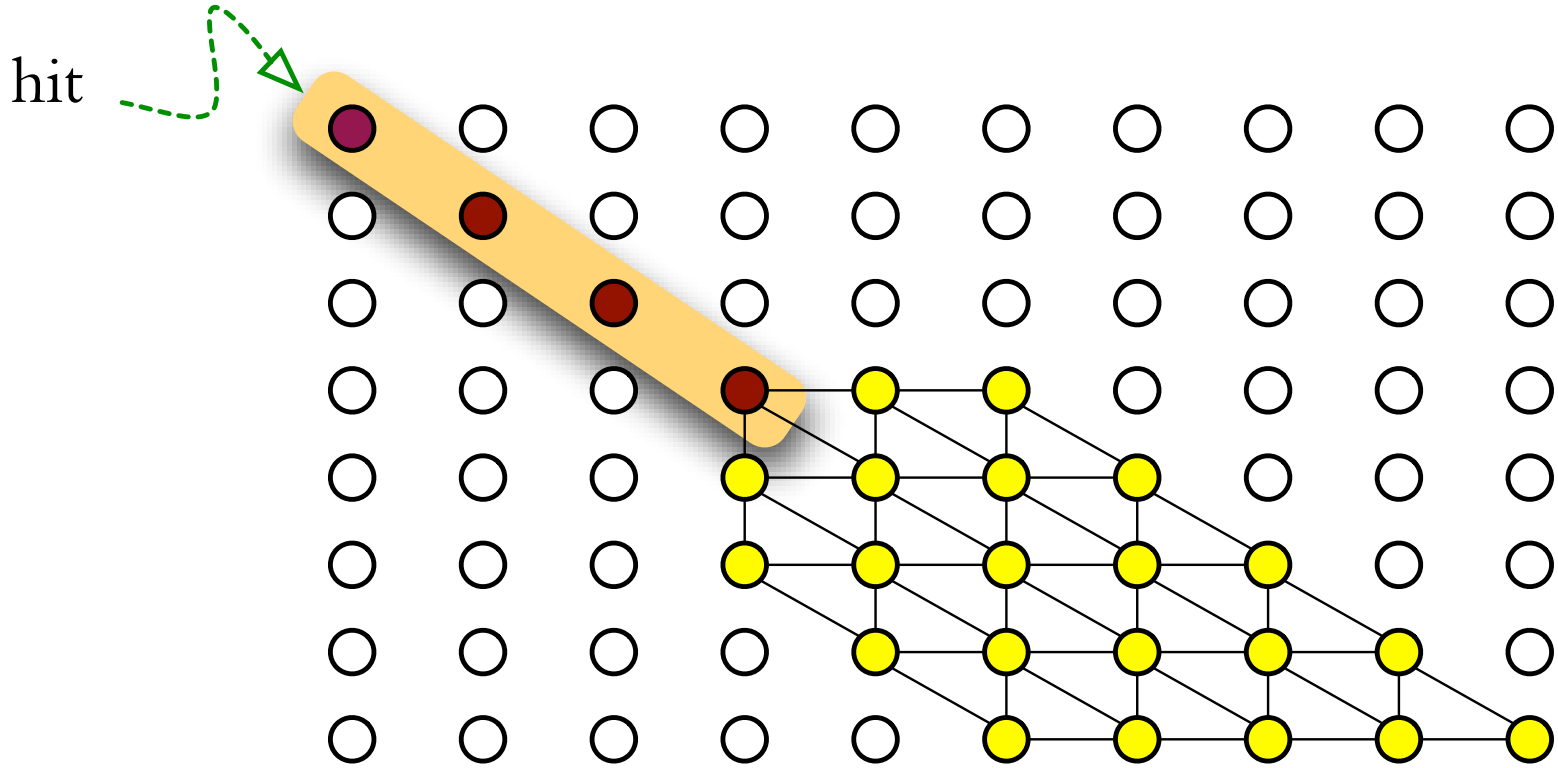
ER1 **Entrée** i_0, j_0 départ de l'extension, s_0 score initial
ER2 meilleur $\leftarrow s_0$; extension $\leftarrow 0$
ER3 $i \leftarrow i_0 + 1$; $j \leftarrow j_0 + 1$; score $\leftarrow s_0$
ER4 **tant que** $i \leq |S|, j \leq |T|$, score ≥ 0
ER5 score $\leftarrow$ score $+ \mathbf{C}\begin{bmatrix} S[i] \\ T[j] \end{bmatrix}$
ER6 **si** score $\geq$ meilleur **alors** meilleur $\leftarrow$ score, extension $\leftarrow j - j_0$
ER7 $i \leftarrow i + 1, j \leftarrow j + 1$
ER8 **retourner** meilleur, extension

Détails : bande [vers sud-est]

B1 **Entrée** i_0, j_0 départ de l'extension, s_0 score initial, d épaisseur
B2 $A^* \leftarrow s_0, i \leftarrow i_0, D \leftarrow i_0 - j_0$
B3 **tant que** $i \leq |S|$
B4 $\quad j \leftarrow \max\{j_0, i - D - d\}$
B5 $\quad$ **tant que** $j \leq \min\{|T|, i - D + d\}$
B6 $\quad\quad$ calculer $A(i,j)$; **si** $A^* < A(i,j)$ **alors** $A^* \leftarrow A(i,j)$
B7 $\quad\quad j \leftarrow j + 1$
B8 $\quad$ **si** $\forall j : A(i,j) \leq 0$ **alors** sauter à Ligne B10.
B9 $\quad i \leftarrow i + 1$
B10 reporter A^*

Détails bande 2 — graphe

Calcul en ligne B6 ($d = 2$)



Détails bande 2 — code

Calcul en ligne B6 : pondération par $\mathbf{C}$

- si $i = i, j = j_0$, alors $A(i, j) = s_0$
- si $i = i_0$, et $j > j_0$, alors $A(i, j) = A(i, j-1) + \mathbf{C}\left[\begin{smallmatrix} - \\ T[j] \end{smallmatrix}\right]$
- si $i > i_0$ et $j = j_0$, alors $A(i, j) = A(i-1, j) + \mathbf{C}\left[\begin{smallmatrix} S[i] \\ - \end{smallmatrix}\right]$
- si $i > i_0 + d$ et $j = i - D - d$, alors
  $A(i, j) = \mathsf{max}\left\{ A(i-1, j-1) + \mathbf{C}\left[\begin{smallmatrix} S[i] \\ T[j] \end{smallmatrix}\right], A(i-1, j) + \mathbf{C}\left[\begin{smallmatrix} S[i] \\ - \end{smallmatrix}\right] \right\}$

Détails bande 2 — code (cont.)

– si $i > i_0$ et $\max\{j_0, i-D-d\} < j < \min\{|T|, i-D+d\}$, alors $A(i,j) = \max\left\{A(i-1,j)+\mathbf{C}\begin{bmatrix} S[i] \\ - \end{bmatrix}, A(i-1,j-1)+\mathbf{C}\begin{bmatrix} S[i] \\ T[j] \end{bmatrix}, A(i,j-1)+\mathbf{C}\begin{bmatrix} - \\ T[j] \end{bmatrix}\right\}$

– si $i > i_0$ et $j = \min\{|T|, i-D+d\}$, alors $A(i,j) = \max\left\{A(i-1,j-1)+\mathbf{C}\begin{bmatrix} S[i] \\ T[j] \end{bmatrix}, A(i,j-1)+\mathbf{C}\begin{bmatrix} - \\ T[j] \end{bmatrix}\right\}$

Détails X-drop

Idée : maintenir A^* score du meilleur alignement et ne pas continuer l'extension si $A(i,j) < A^* - X$

Stocker col_g, col_d : colonnes de la dernière rangée que l'on a explorée.

(Ou code plus simple si l'exploration est dans une bande seulement : on n'a pas besoin de col_g, col_d)

[Code pour extensions vers sud-est seulement]

Détails X-drop 2

XD1 **Entrée** i_0, j_0 départ de l'extension, s_0 score initial, X
XD2 $A^* \leftarrow s_0$, $\mathsf{col_g} \leftarrow j_0$, $\mathsf{col_d} \leftarrow |T|$
XD3 $i \leftarrow i_0$
XD4 **tant que** $i \leq |S|, \mathsf{col_g} \leq \mathsf{col_d}$
XD5 $\quad j \leftarrow \mathsf{col_g}$
XD6 $\quad$ **tant que** $j \leq \min\{\mathsf{col_d} + 1, |T|\}$
XD7 $\quad\quad$ calculer $A(i, j)$
XD8 $\quad\quad$ **si** $A(i, j) > A^*$ **alors** $A^* \leftarrow A(i, j)$
XD9 $\quad\quad$ **si** $A(i, j) < A^* - X$ **alors** $A(i, j) \leftarrow -\infty$
XD10 $\quad\quad$ $j \leftarrow j + 1$
XD11 $\quad$ **tant que** $\mathsf{col_g} \leq \mathsf{col_d}$ et $A(i, \mathsf{col_g}) = -\infty$, $\mathsf{col_g} \leftarrow \mathsf{col_g} + 1$
XD12 $\quad$ $\mathsf{col_d} \leftarrow \mathsf{col_d} + 1$; **tant que** $\mathsf{col_d} \geq \mathsf{col_g}$ et $A(i, \mathsf{col_d}) = -\infty$, $\mathsf{col_d} \leftarrow \mathsf{col_d} - 1$
XD13 $\quad$ $i \leftarrow i + 1$
XD14 retourner A^*

Détails X-drop 3

Calcul en ligne XD7 : pondération par $\mathbf{C}$

- si $i = i_0$ et $j > j_0$, alors $A(i,j) \leftarrow A(i,j-1) + \mathbf{C}\begin{bmatrix} - \\ T[j] \end{bmatrix}$
- si $i > i_0$ et $j = \mathsf{col_g}$ alors $A(i,j) \leftarrow A(i-1,j) + \mathbf{C}\begin{bmatrix} S[i] \\ - \end{bmatrix}$
- si $i > i_0$ et $j = \mathsf{col_d} + 1$ alors $A(i,j) \leftarrow \mathsf{max}\Big\{A(i-1,j-1) + \mathbf{C}\begin{bmatrix} S[i] \\ T[j] \end{bmatrix}, A(i,j-1) + \mathbf{C}\begin{bmatrix} - \\ T[j] \end{bmatrix}\Big\}$
- sinon $A(i,j) \leftarrow \mathsf{max}\Big\{A(i-1,j) + \mathbf{C}\begin{bmatrix} S[i] \\ - \end{bmatrix}, A(i-1,j-1) + \mathbf{C}\begin{bmatrix} S[i] \\ T[j] \end{bmatrix}, A(i,j-1) + \mathbf{C}\begin{bmatrix} - \\ T[j] \end{bmatrix}\Big\}$

Un exemple : chevauchements en CAP3

Séquence combinée de tous les fragments:

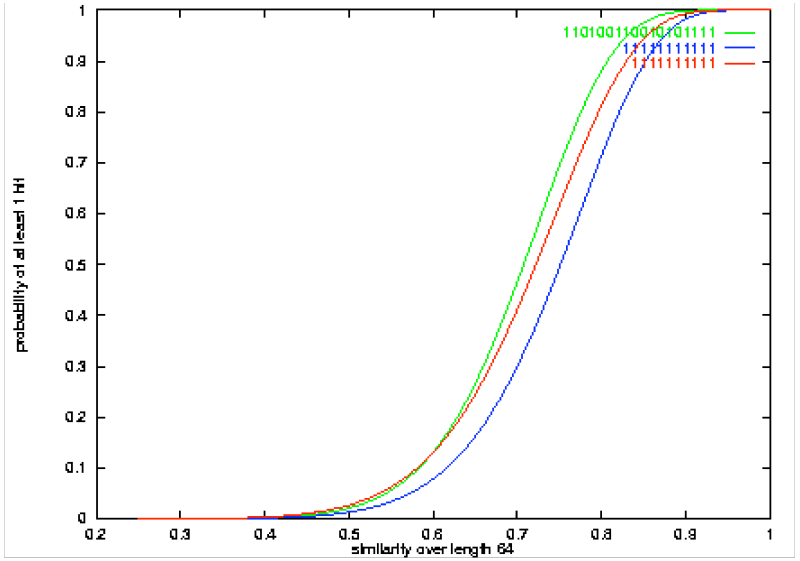
1. Tableau de hachage des k-mers de la séquence combinée.
2. Chaque fragment f ainsi que son complément sont comparés à la séquence combinée, pour trouver des HSPs.
3. Les chevauchements potentiels de 2) sont évalués par alignement dans une bande.

Huang et Madan, *Genome Res.* 9 : 868.

Meilleur hachage : graines espacées

$\mathcal{S} = \{s_1, \ldots, s_k\}$: graine (*seed*) (ℓ, k) ;
fonction de hachage : $h(u) = u[s_1] \cdot u[s_2] \cdots u[s_k]$.

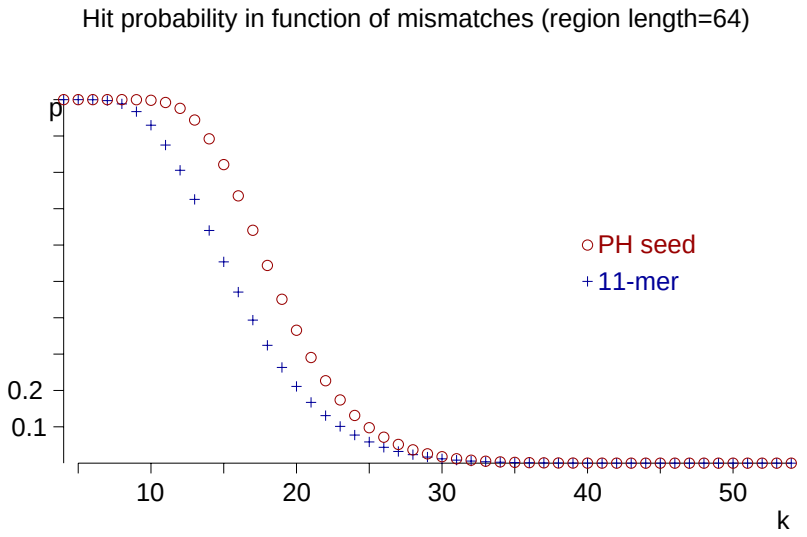
BLAST : graine contigue (11-mer) $\mathcal{S} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
PatternHunter : graine espacée $\mathcal{S} = \{1, 2, 3, 5, 8, 10, 13, 14, 16, 17, 18\}$



Ma et al. *Bioinformatics* 18 :440

Graines espacées

Graine espacée est meilleure pour toutes distributions de similarité



Problème : trouver la meilleure graine (NP-difficile), calculer la sensibilité (NP-difficile)

Calculer la sensibilité de la graine

Modèle : région de similarité de longueur ℓ, niveau de similarité p. Séquence d'indicateurs $X_1 X_2 \cdots X_\ell$ où $X_i = 1$ ssi un match en position i (avec probabilité p).

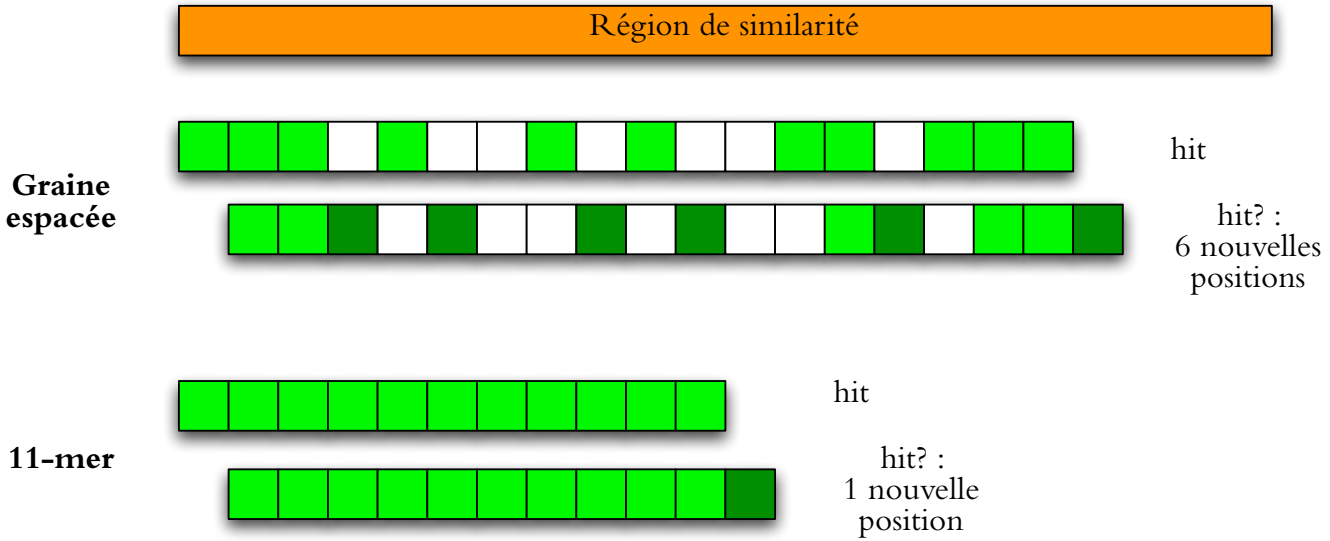
Hit en position i : si $X_{i+k-1} = 1$ pour tout $k \in \mathcal{S}$.

Calculer cette probabilité ...

Indépendance de positions

Nombre de hits en espérance à peu près la même

hits sont plus dense avec graine contigue

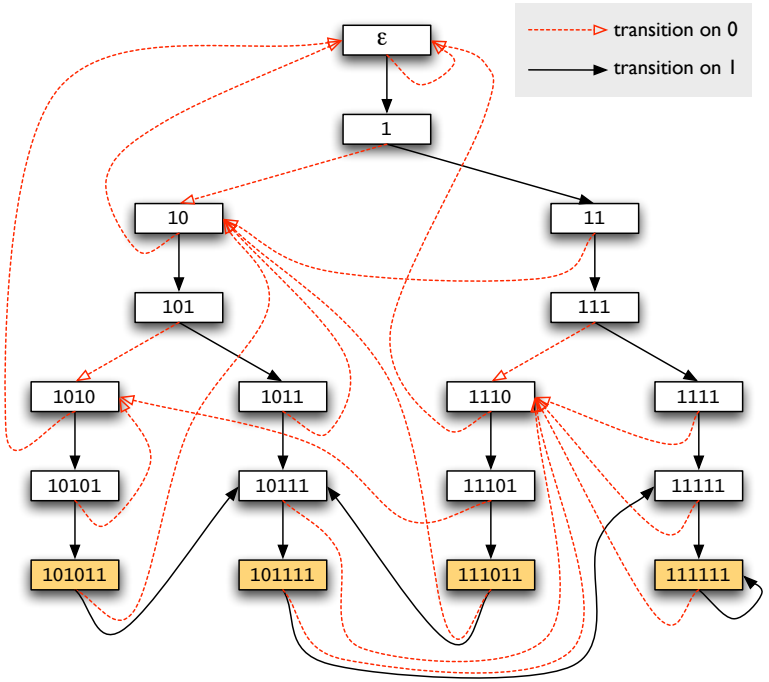


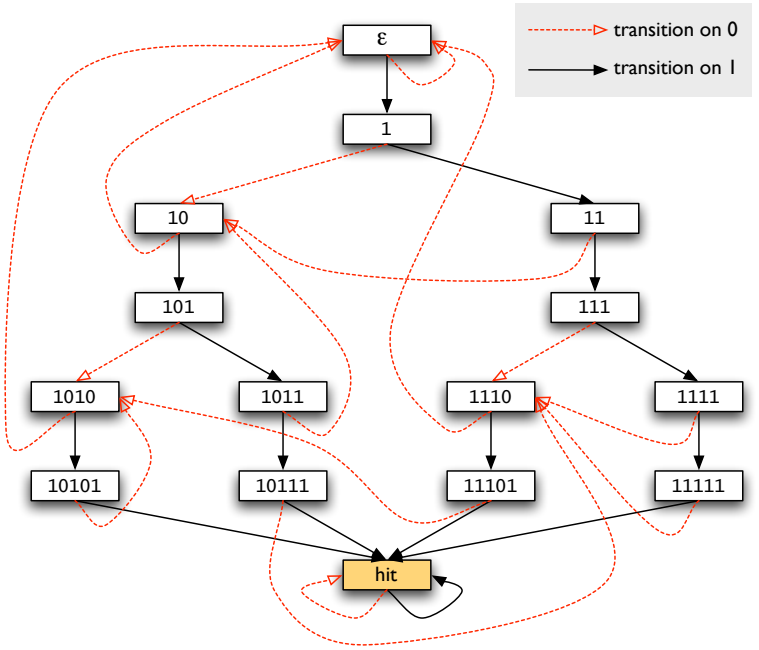
probabilité d'un deuxième hit : p^6 ou p^1

nombre de hits H : $\mathbb{E}H = \mathbb{E}\left[H \;\middle|\; H > 0\right]\mathbb{P}\{H > 0\}$

or, $\mathbb{E}\left[H \;\middle|\; H > 0\right]$ est plus large avec k-mer

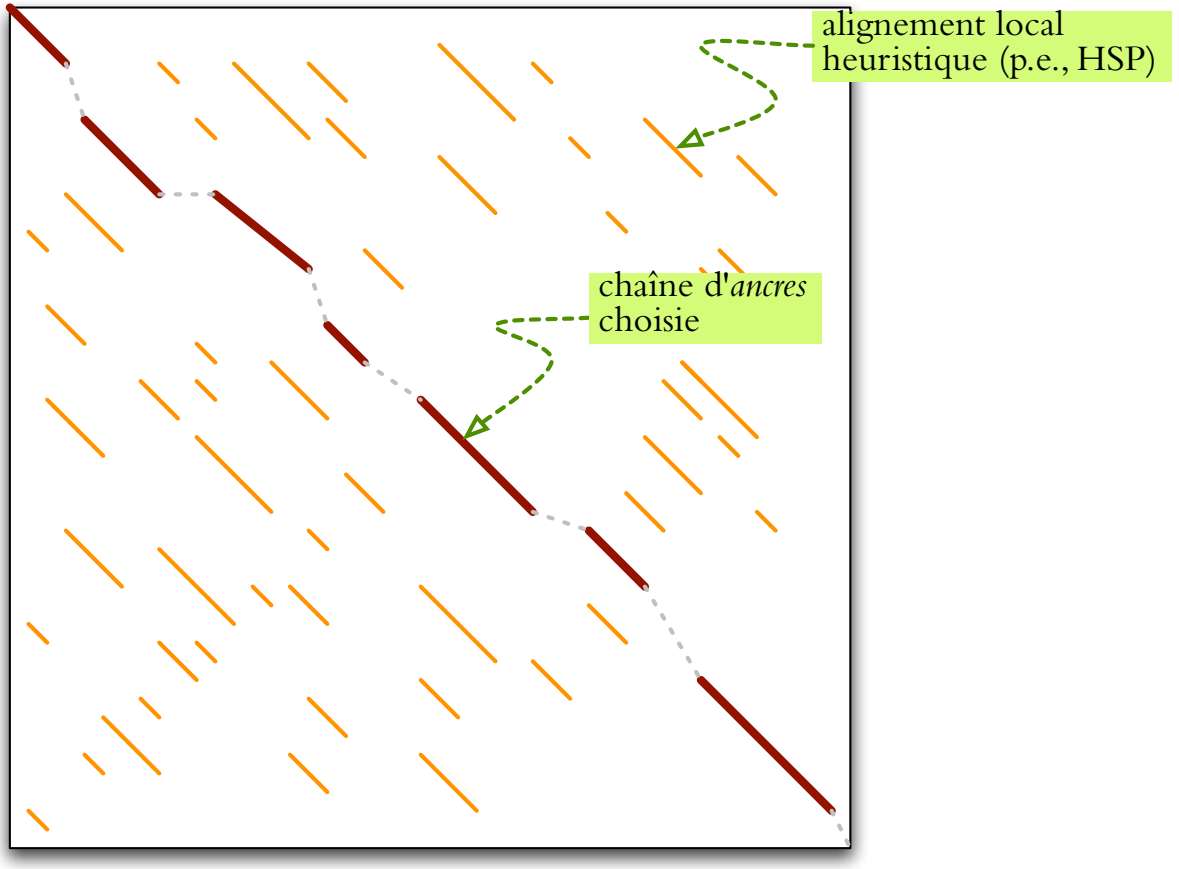
Sensibilité de la graine 2





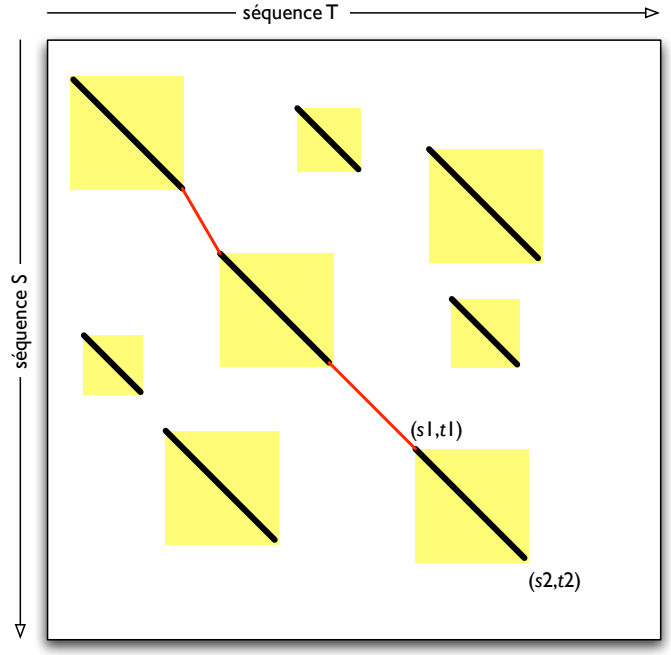
Graine = `101011`

Alignement global dans les Megas



Procédure : (1) recherche de homologies (hachage+extension), (2) sélection d'ancres, (3) PD dans une bande autour de la chaîne d'ancres et dans les régions entre ancres

Programmation dynamique éparse

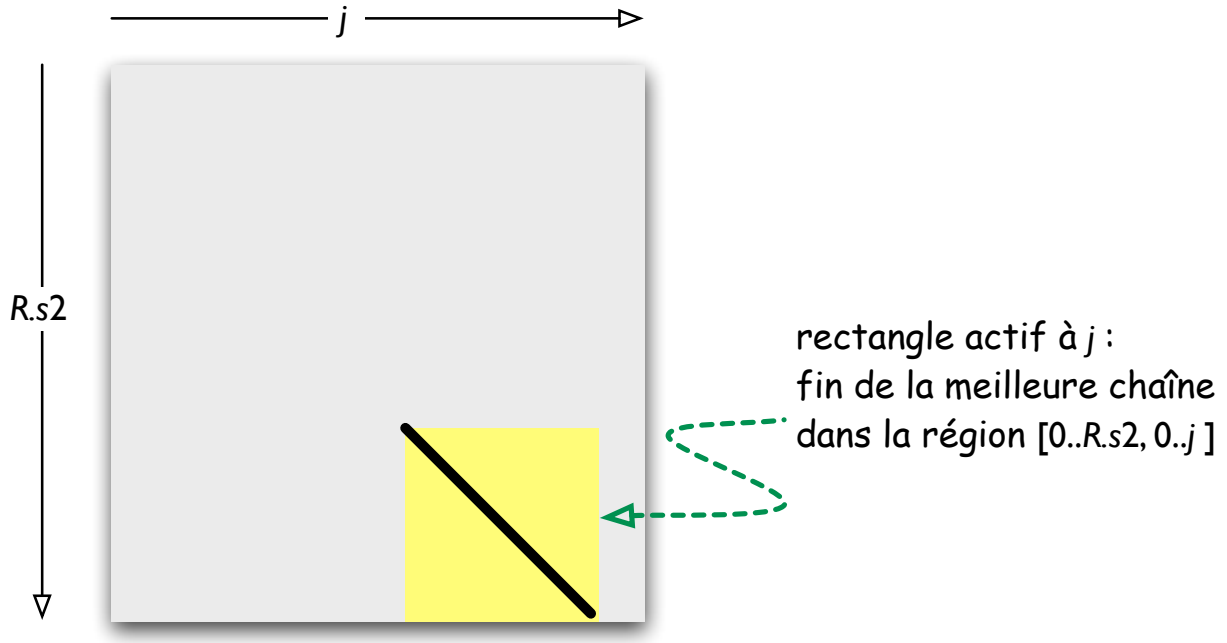


Liste de rectangles (alignements locaux) :
rectangle R en position $(R.s1, R.t1), (R.s2, R.t2)$ avec score $R.v$

chaîne : ensemble de rectangles *non-chevauchant*

PD éparse 2

Récurrence pour calculer $V(R)$: score maximum d'une chaîne qui finit par rectangle R



Clé : maintenir la liste $\mathcal{L}_j$ de rectangles *actifs* pour $j = 0, \ldots, |T|$

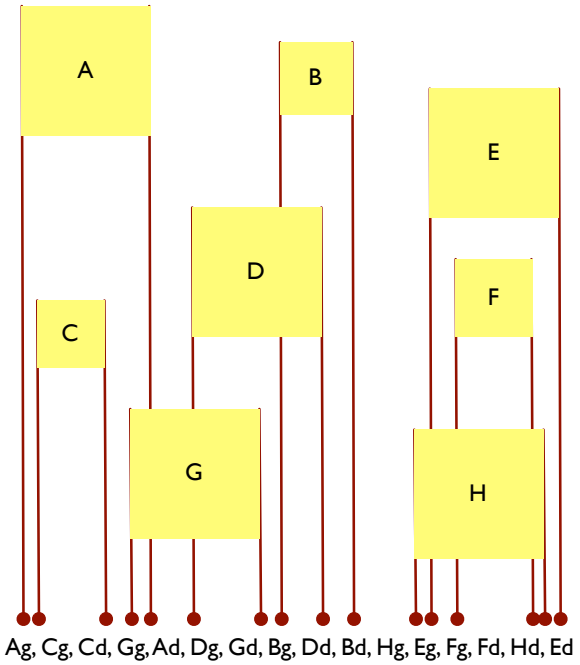
PD éparse 3

Déf. Rectangle R est actif pour j ssi
(1) $R.t2 \leq j$; et
(2) $\forall R'$ avec $R'.t2 \leq j$: si $R'.s2 \leq R.s2$ alors $V(R') \leq V(R)$.
Lemme. Si $R'.s2 = R.s2$ et tous les deux sont actifs, alors $V(R') = V(R)$.

$\Rightarrow \mathcal{L}$ contient des rectangles actifs avec $.s2$ différents — choisir R avec $R.t2$ minimum si > 1 rectangles actifs avec le même $.s2$.

PD éparse 4

Trier les $.t1, .t2$: liste $\mathcal{J}$, calculer $\mathcal{L}_j$ pour $j \in J$ en ordre ascendant



PD éparse 5

en arrivant au côté gauche du rectangle R :
- trouver $R' \in \mathcal{L}_{R.t1}$ t.q. $R'.s2 \leq R.s1$ avec $V(R')$ maximal
- $V(R) \leftarrow V(R') + R.v$ (si aucun tel R', $V(R) \leftarrow R.v$

Lemme. Si $R, R' \in \mathcal{L}_j$, alors $R.s2 \neq R'.s2$ et $R.s2 < R'.s2$ ssi $V(R) < V(R')$.

Preuve. Par définition de $\mathcal{L}_j$.

$\Rightarrow$ stocker $\mathcal{L}_j$ dans l'ordre de $.s2$ et $V(\cdot)$ en même temps

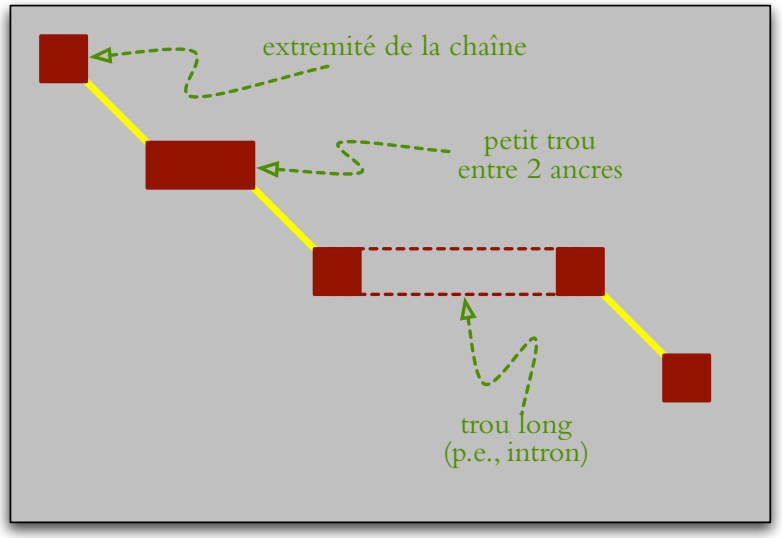
PD éparse 6

en arrivant au côté droit du rectangle R — mise à jour de $\mathcal{L}$:
- supprimer $R' \in \mathcal{L}$ t.q. $R'.s2 \geq R.s2$ et $V(R') < V(R)$
- trouver $R' \in \mathcal{L}$ t.q. $R'.s2 \leq R.s2$ et $V(R')$ max. : si $V(R') = V(R)$ et $R'.s2 < R.s2$, ou si $V(R') < V(R)$, alors insérer R dans $\mathcal{L}$ juste après R' (ou s'il n'y a pas de R' avec $R'.s2 \leq R.s2$, insérer au début)

Temps de calcul : $O(r \log r)$ pour r rectangles (utiliser un arbre binaire balancé ; chaque rectangle est inclus à ou est supprimé de $\mathcal{L}$ une fois au plus)

Chaînage avec pénalisation de trous

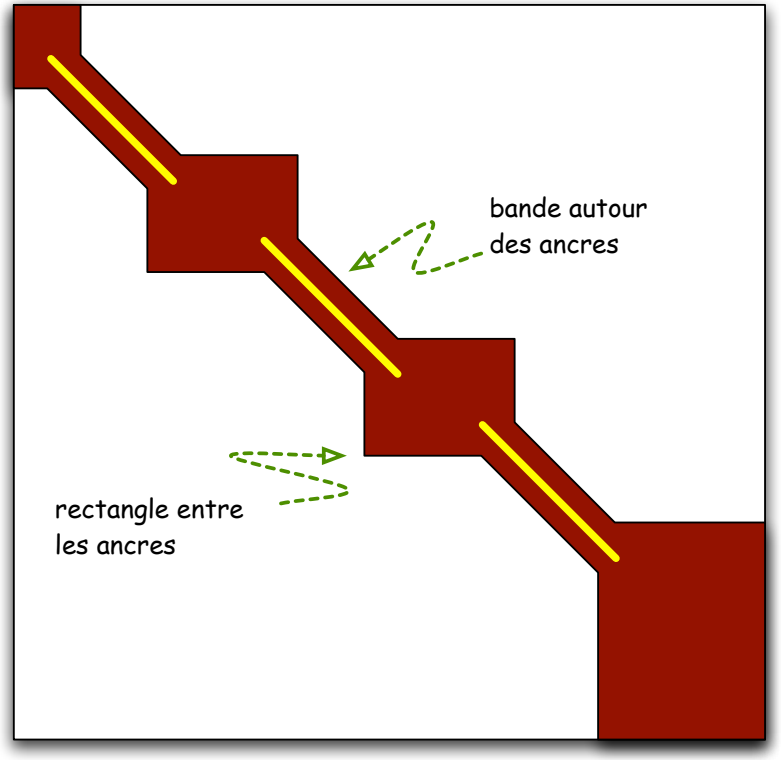
Régions de PD complète



donne le score de joindre deux HSPs : modifier l'algorithme de sélection de chaîne pour inclure ce score aussi

Slater & Birney *BMC Bioinformatics* 6 :31

LAGAN : alignement global avec ancres



sélection d'ancres : chaîne optimale d'alignements locaux

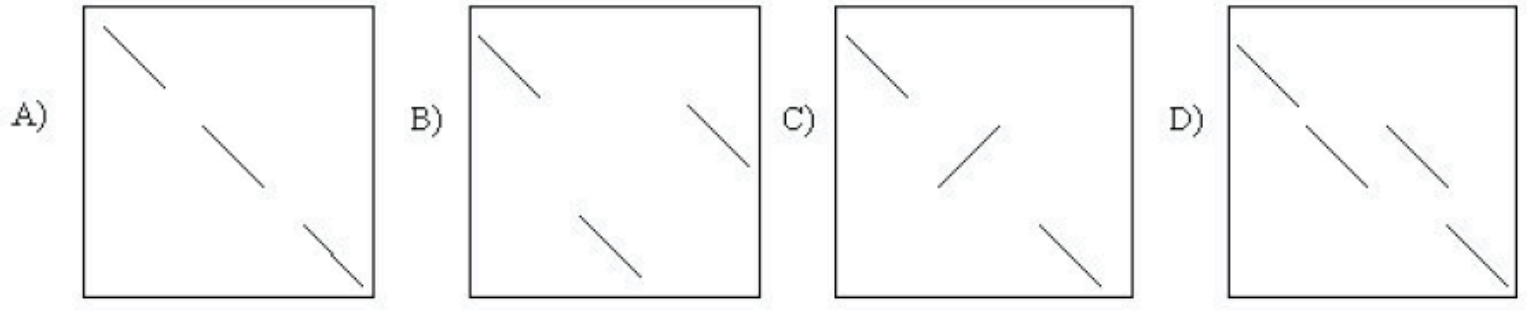
Brudno & al. *Genome Res.* 13 : 721

Quelques autres idées

- recalcul de scores (alignement local avec trous : trouver le placement optimal du trou entre deux hits)

- génération de nouveaux ancres plus épais dans des régions sans ancres par hachage plus sensible

Alignement — réarrangements

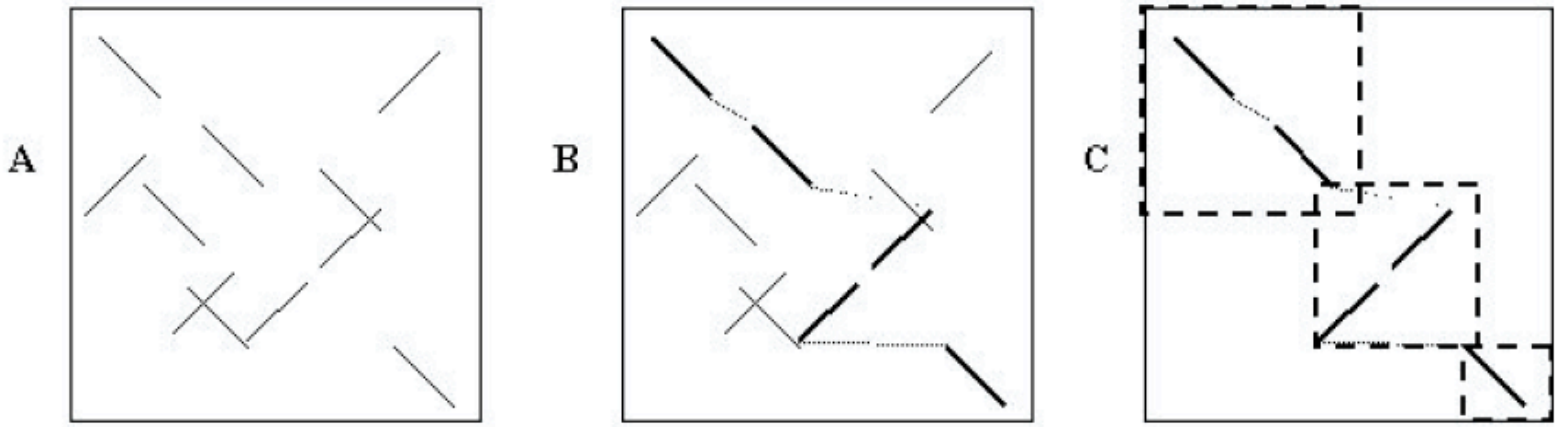


(B. transposition, C. inversion, D. duplication)

Dans des synténies humain-souris : par 1 Mbp on a en moyenne
2 inversions, 17 duplications, 7 transpositions, 200 deletions de longueur $>$ 100 pb,

Brudno & al. *Bioinformatics* 19 :i54 ; Kent & al. *PNAS* 100 : 11484

Alignement glocal



réarrangements : inversions et translocations sont permises

A. alignements locaux

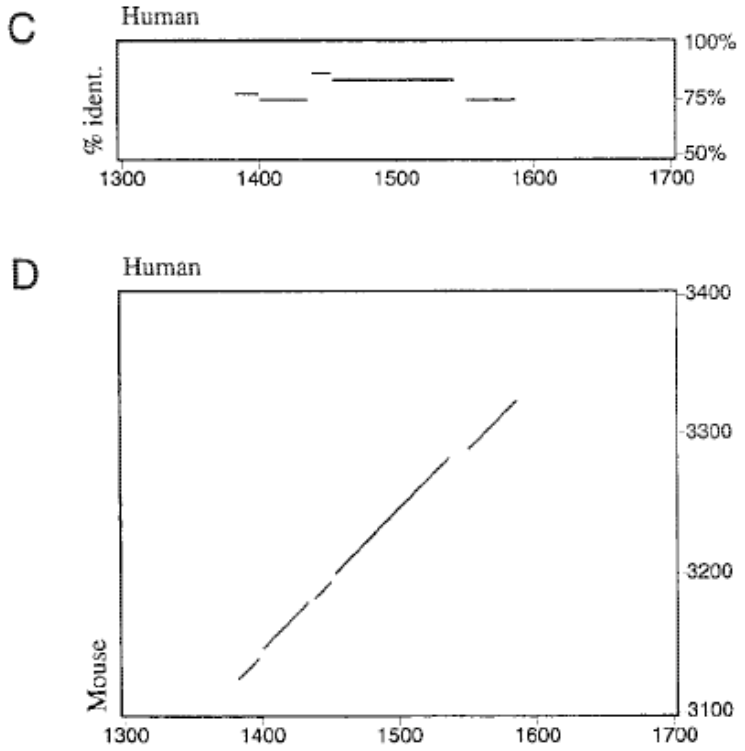
B. chaîne 1-monotone

C. LAGAN dans les blocs sans inversions

Brudno & al. *Bioinformatics* 19 : i54

Visualisation des alignements : PIPs

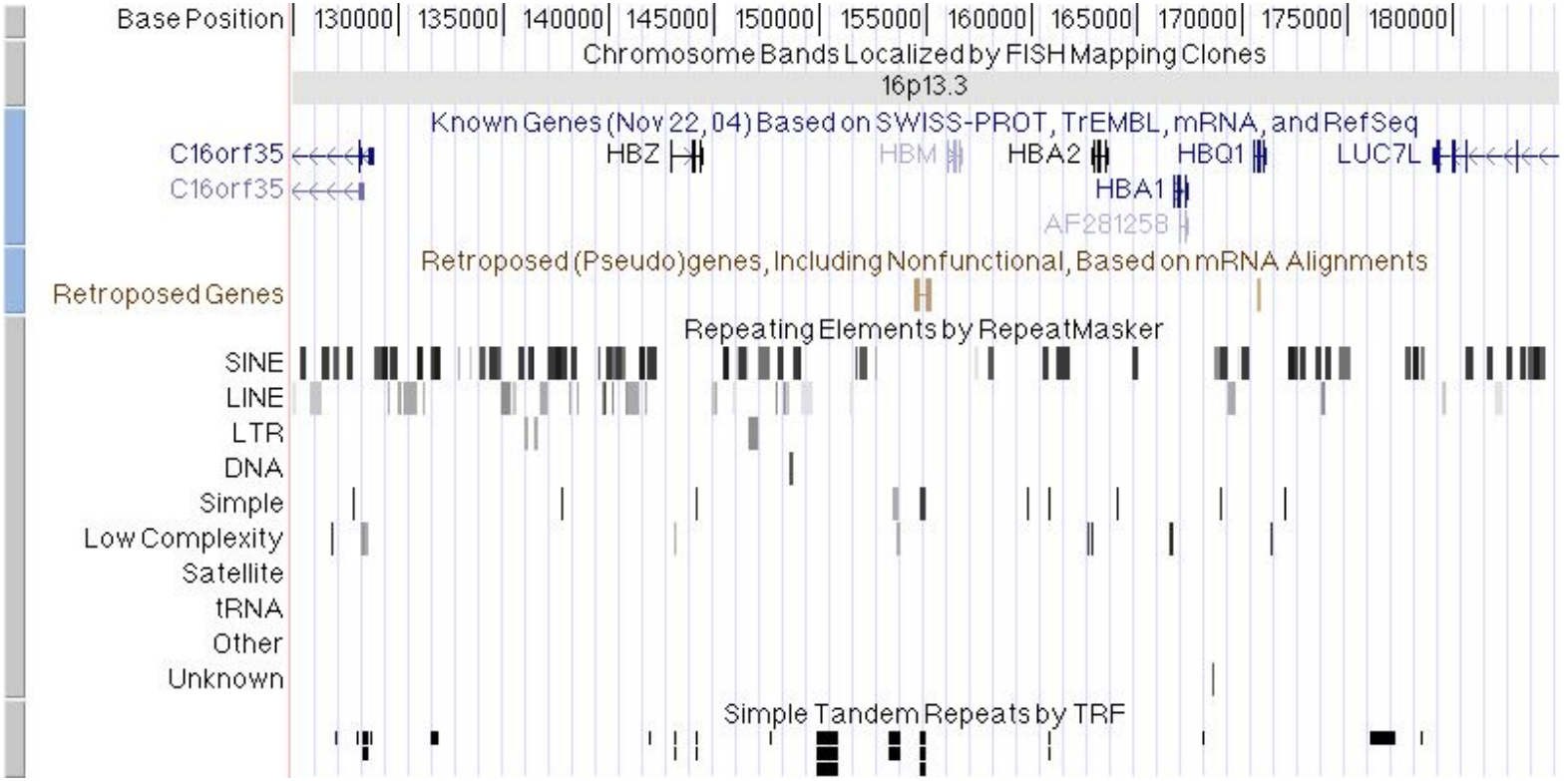
PIP — *percent identity plot*



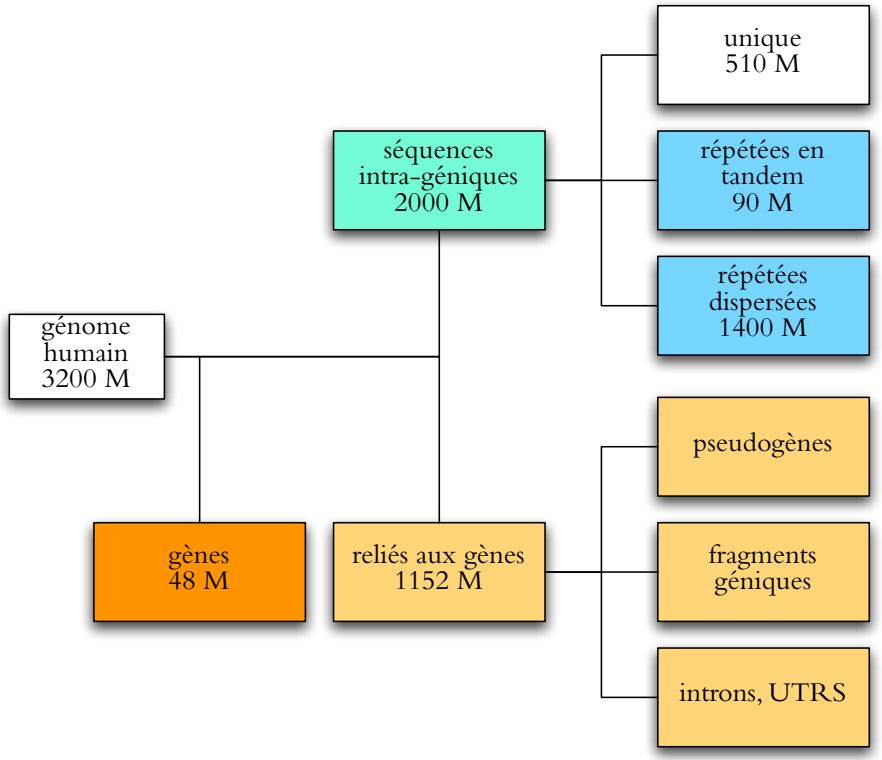
(on considère les blocs entre les trous, et on affiche le pourcentage de matchs dans chaque bloc)

Schwartz & al. *Genome Res.* 10 : 577

Constitution d'un génome



Constitution d'un génome 2



en Procaryotes : 90–97% codant

après Watson et al *Molecular Biology of the Gene* (2004)

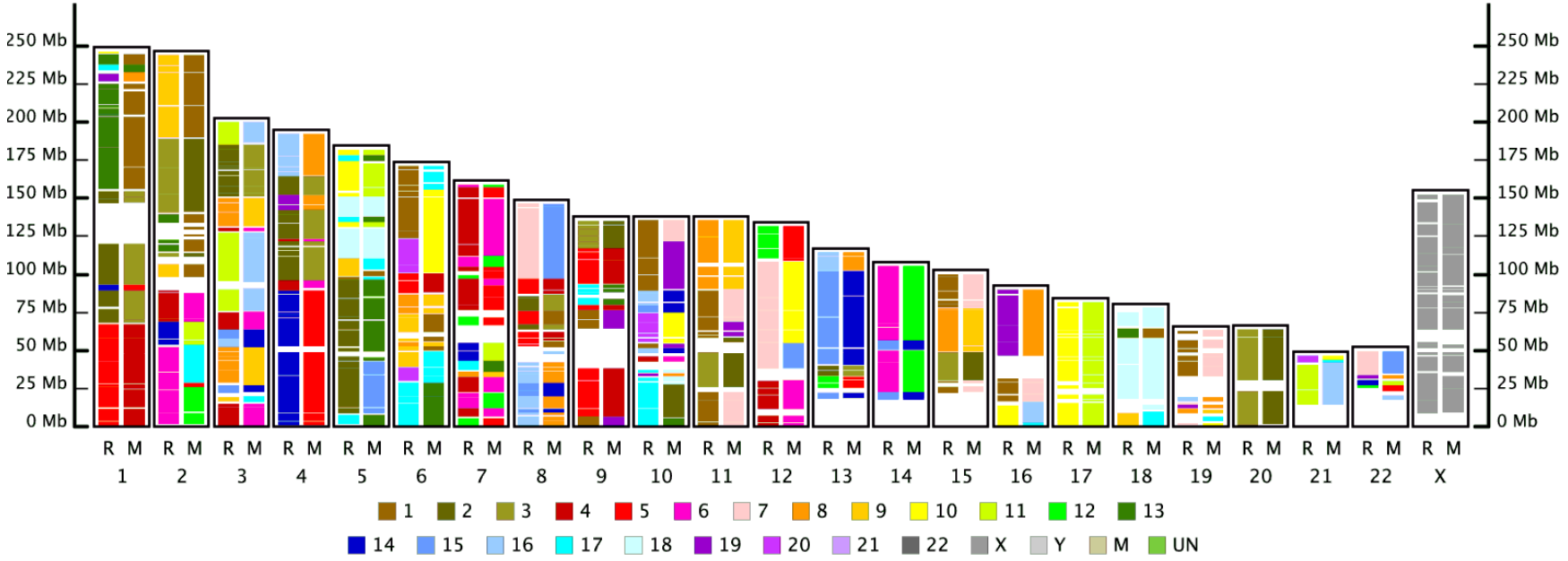
Alignement de génomes

Problème : beaucoup de réarrangements, insertions indépendantes dans les deux génomes

→ n'utiliser que les chaînes d'alignements locaux

on veut détecter des régions synténiques — déscendantes de la même région dans l'ancêtre commun

Synténies

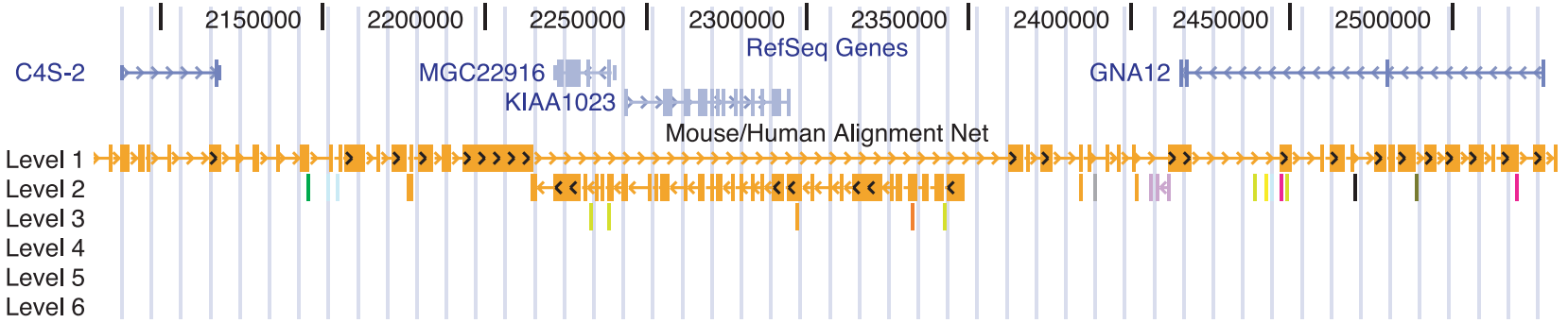


(chromosomes humains coloriés par syntenies dans souris et rat)

http://www.genboree.org/

Micro-réarrangements

beaucoup de réarrangements à tous les niveaux



Kent & al. *PNAS* 100 : 11484

BLASTZ

1. masquer les régions répetées

2. hits avec la graine espacée 1110100110010101111 (12 sur 19 positions) + une transition A-G ou T-C permise

3. extension sans trou + extension avec trou si score est assez grand

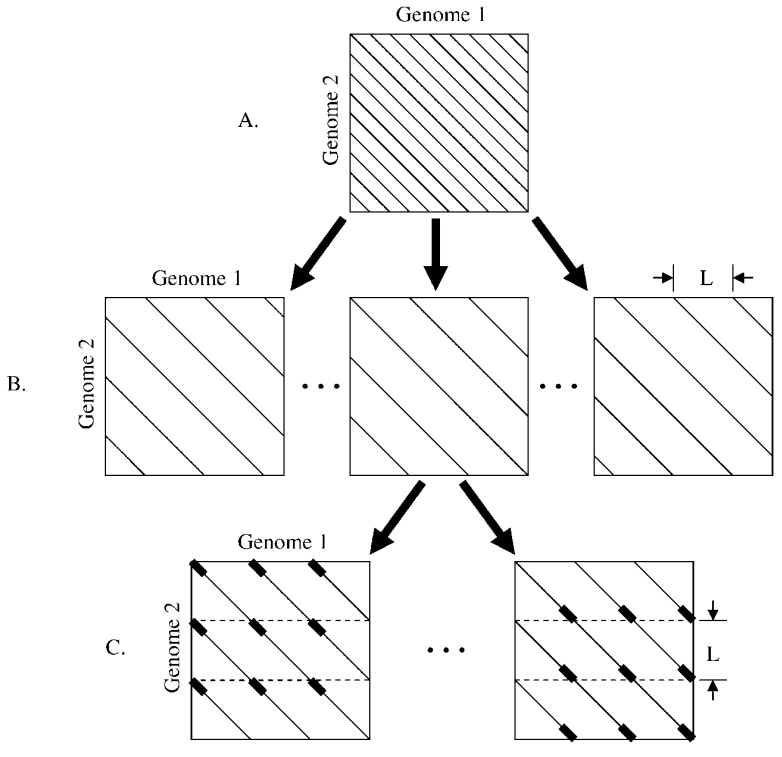
4. hit+extend dans les régions entre alignements locaux (hits avec 7-mers)

5. chaînage d'alignements proches

Schwartz & al *Genome Res.* 13 : 103

PASH

Parallèlisation par diagonales + aucune extension

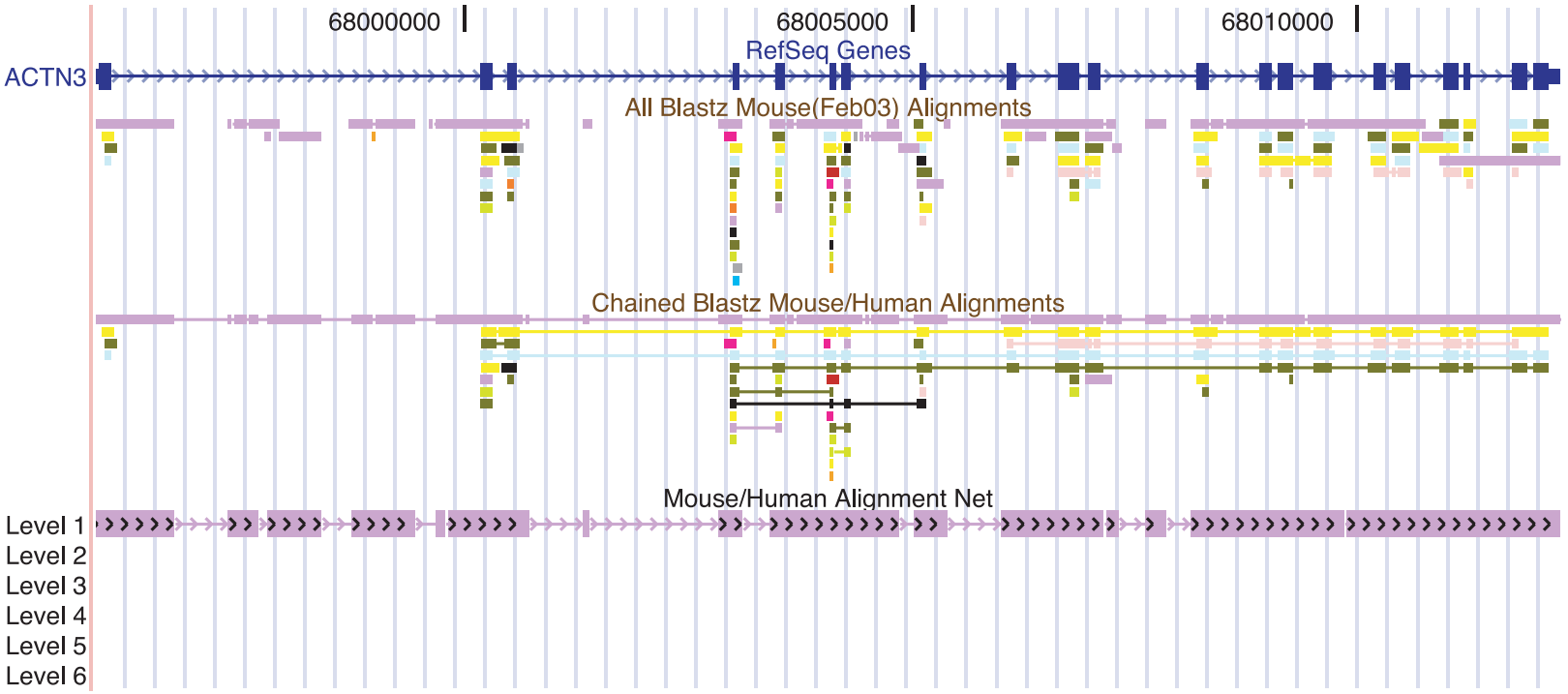


(humain-souris : 24 CPU jours vs. 481 avec BLASTZ)

Kalafus & al. *Genome Res.* 14 : 672

Chains et nets

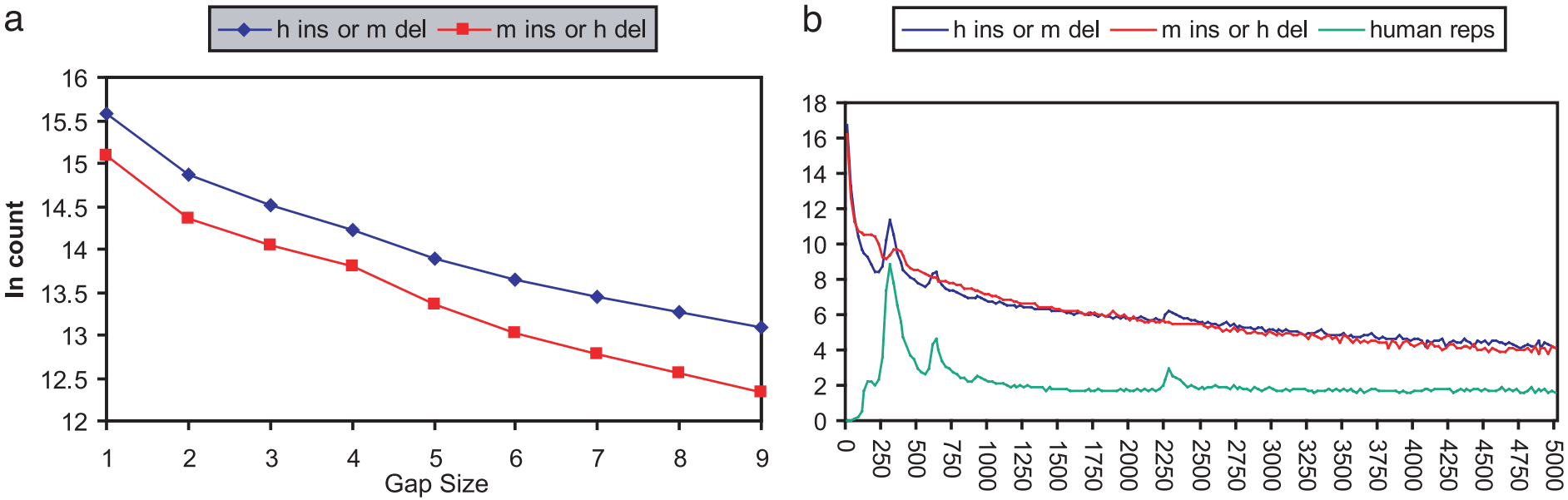
Chaînage récursive — chaînes imbriquées



Kent & al. *PNAS* 100 : 11484

Chains et nets 2

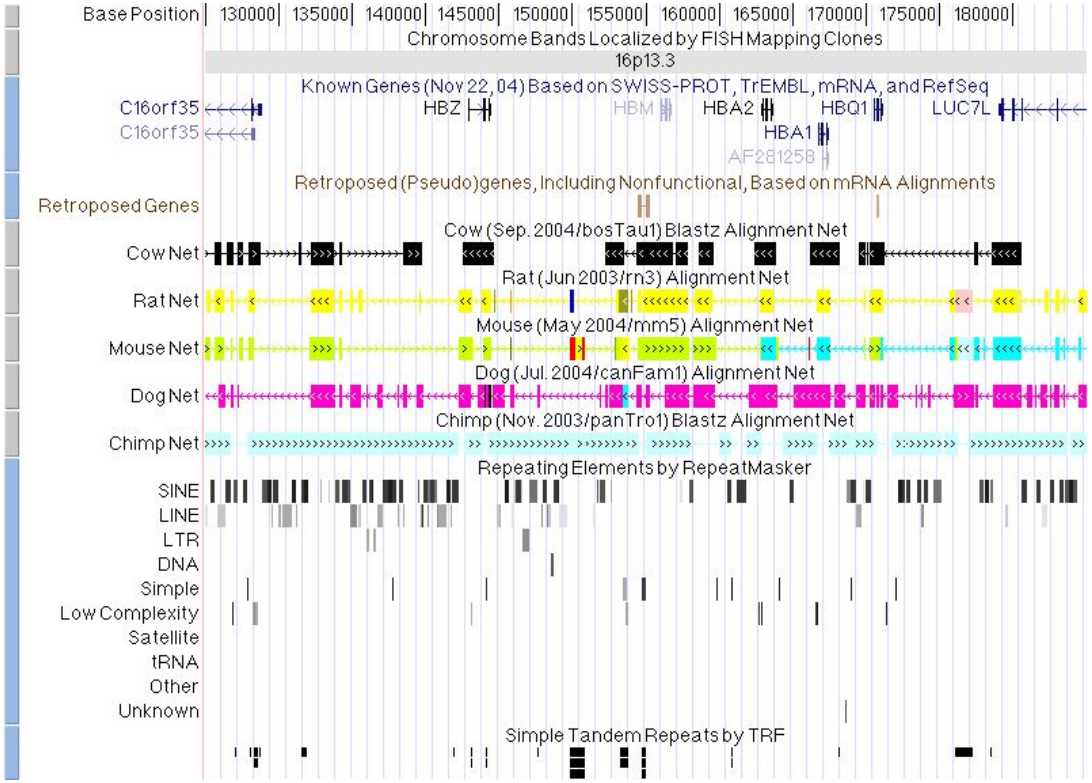
distribution de trous



Kent & al. *PNAS* 100 : 11484

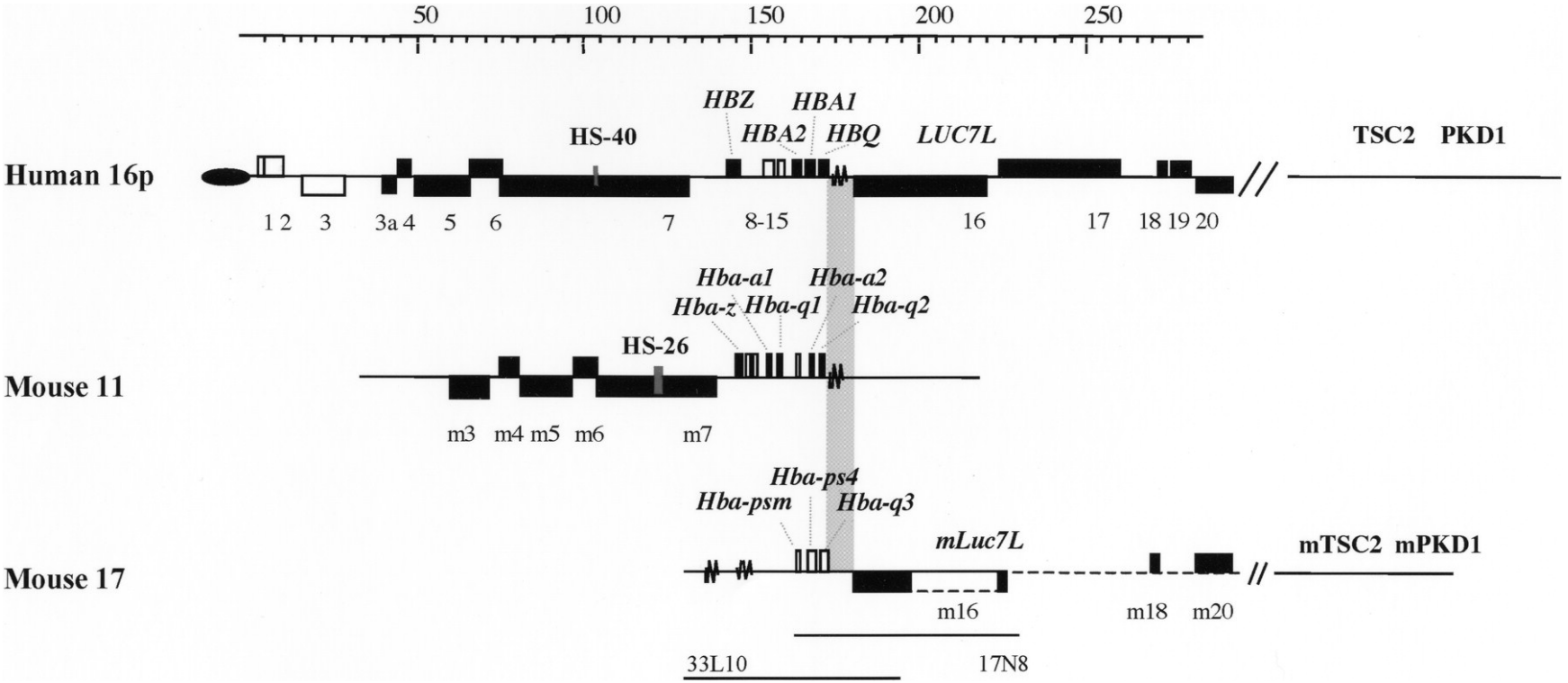
Exemple : α-globines dans le souris

[breakpoint dans le souris]



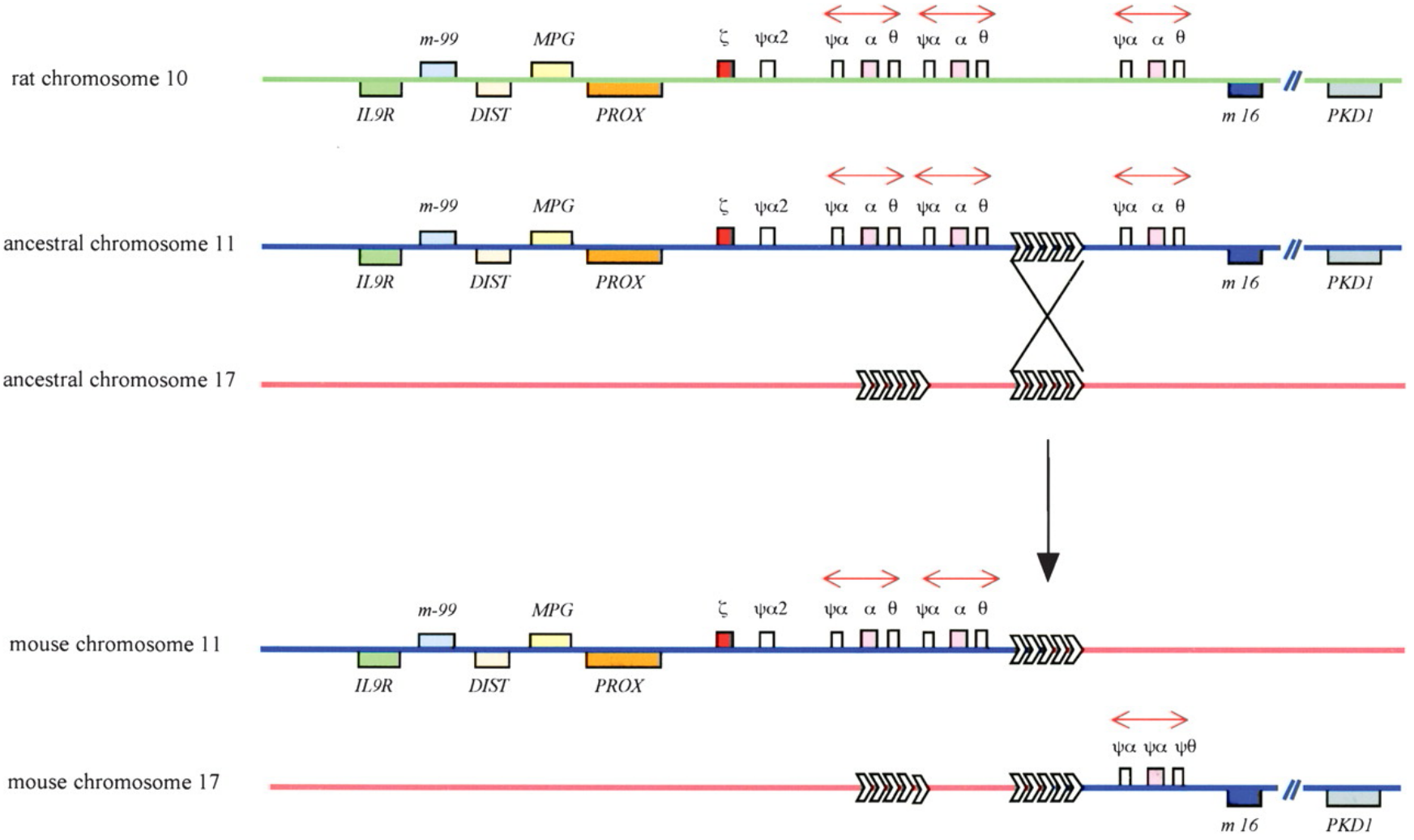
Tufarelli & al. *Genome Res.* 14 :623 (2004)

α-globines dans le souris



Tufarelli & al. *Genome Res.* 14 :623 (2004)

α-globines

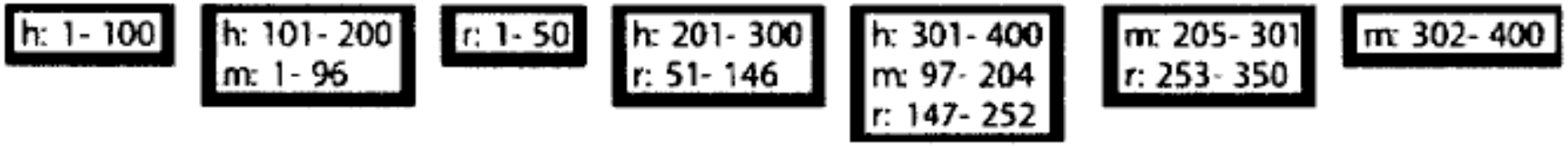


Tufarelli & al. *Genome Res.* 14 :623 (2004)

Alignment comme annotation

Annotation d'un génome : segments avec des propriétés (gène, promoteur, exon, intron, . . .)

Alignment de deux ou plusieurs génomes : peut être consideré comme l'annotation d'un génome par les autres

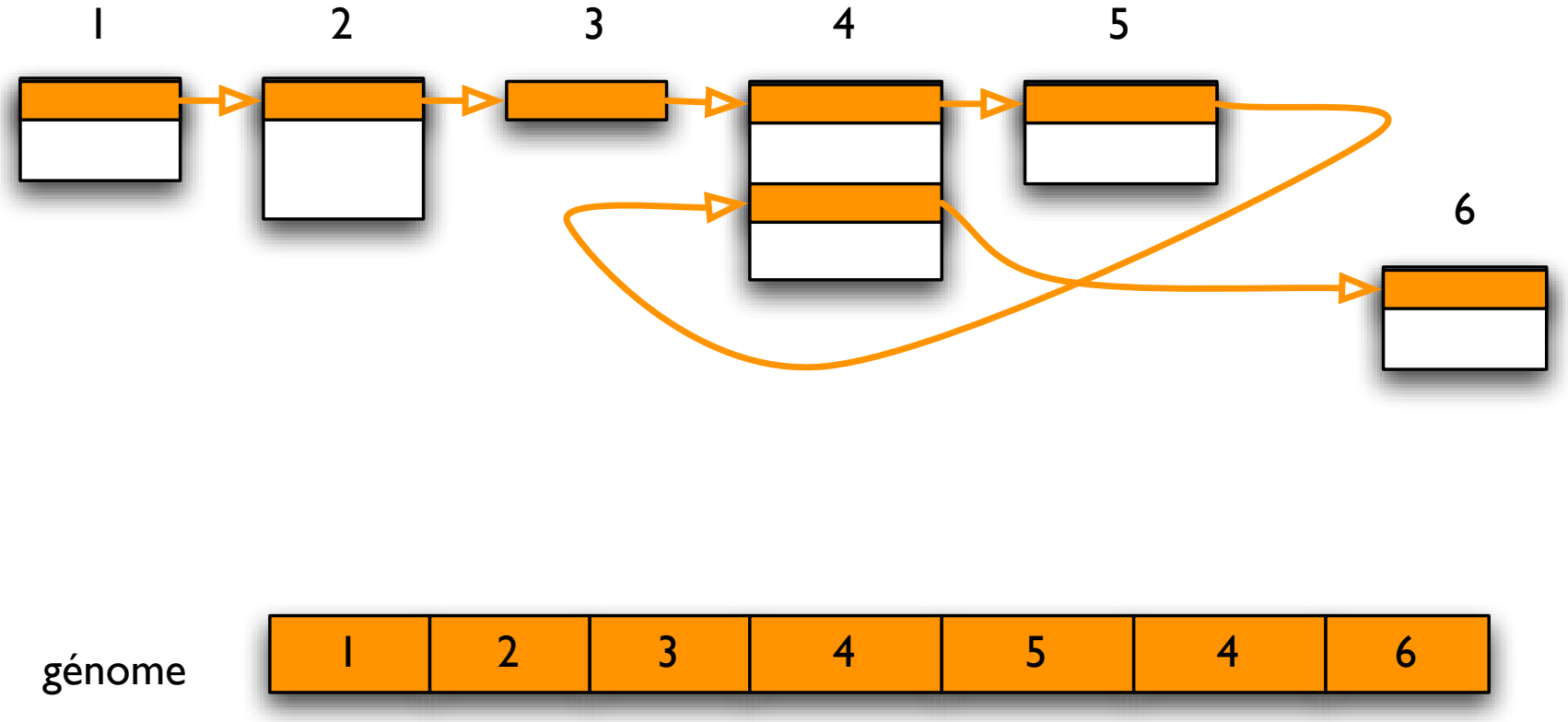


Alignement : ensemble de blocs où chaque bloc est une région d'un seul génome, ou l'alignement local de multiples génomes

Blanchette & al. *Genome Res.* 14 : 708

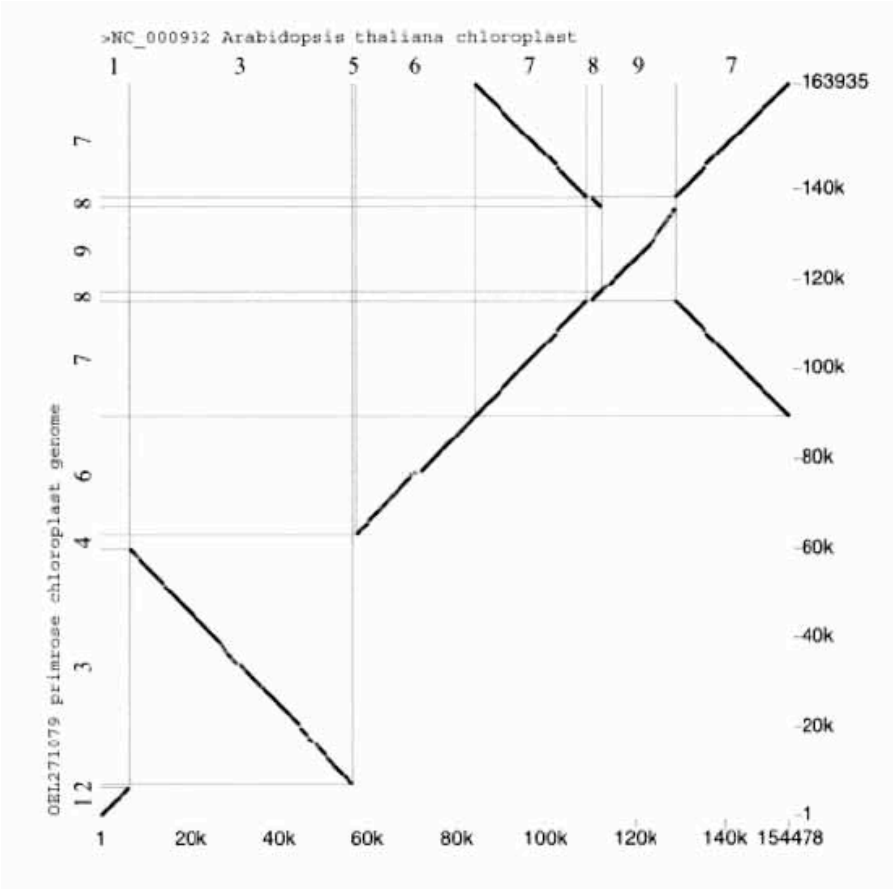
Ensemble de blocs

Projection des blocs sur un génome



Blanchette & al. *Genome Res.* 14 : 708

Exemple



Blanchette & al. *Genome Res.* 14 : 708