

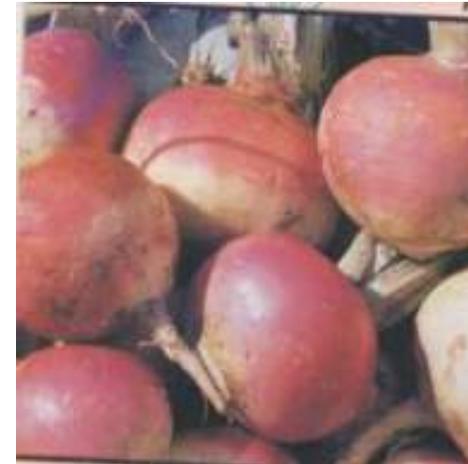
M1 - AAGB - BIM

Algorithmes sur les arbres et les graphes en bioinformatique

Algorithmes pour analyser
les réarrangements des génomes

Alessandra Carbone
Université Pierre et Marie Curie

Navets et choux sont différent en apparence et en goût,
mais ils partagent un ancêtre récent



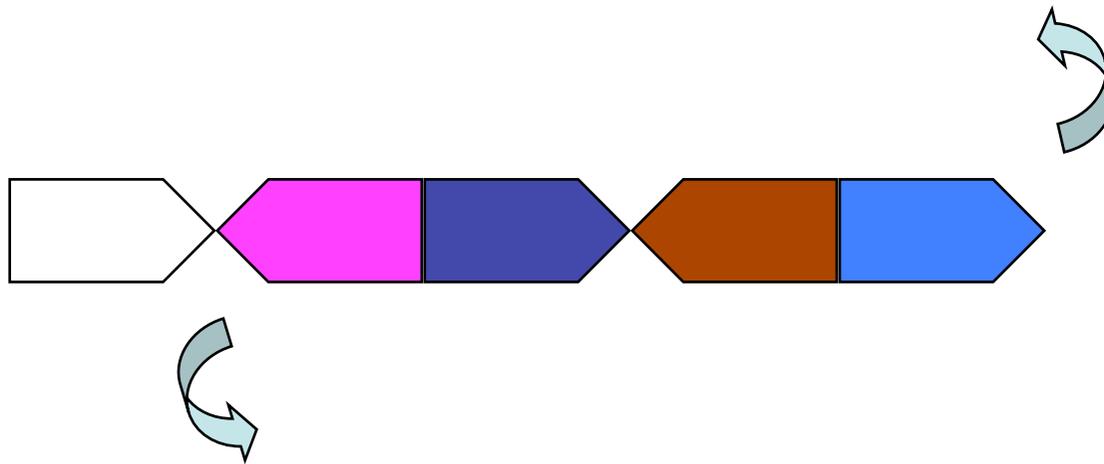
La comparaison des séquences de **gènes** n'amène pas de l'information
évolutive intéressante

Navet vs Chou : gènes de l'ADN mitochondriale quasiment identiques

- En 1980s Jeffrey Palmer a étudié l'évolution des organelles chez les plantes en comparant les génomes mitochondriaux de navet et chou.
- 99% de similarité entre les gènes
- Ces séquences étaient différentes dans l'ordre des gènes.
Surprise!
- Cet étude a démarré l'analyse systématique des réarrangements des génomes dans l'évolution moléculaire.

Navet vs Chou : ordre différent des gènes dans le mtDNA

Comparaison de l'ordre des gènes :



Navet vs Chou : ordre différent des gènes dans le mtDNA

Comparaison de l'ordre des gènes :



Navet vs Chou : ordre différent des gènes dans le mtDNA

Comparaison de l'ordre des gènes :



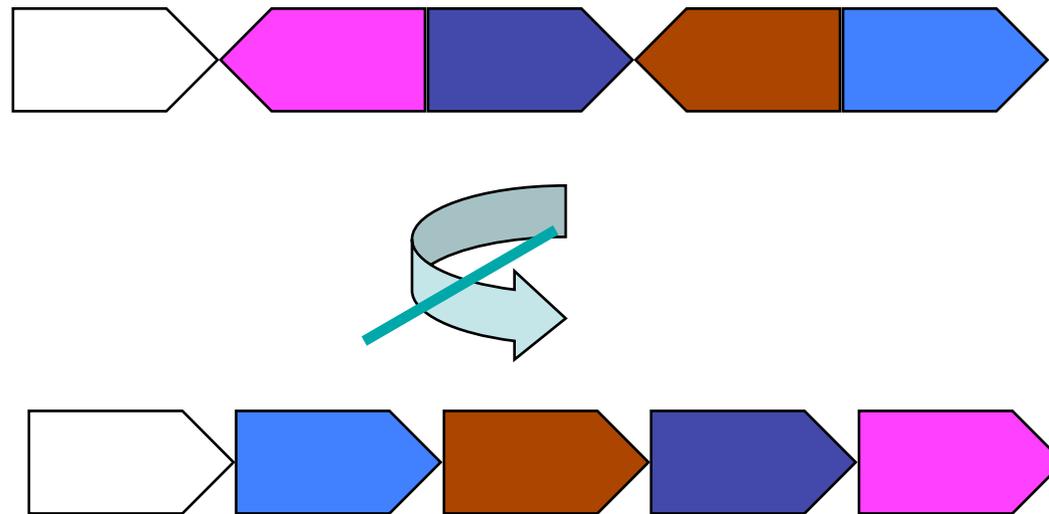
Navet vs Chou : ordre différent des gènes dans le mtDNA

Comparaison de l'ordre des gènes :



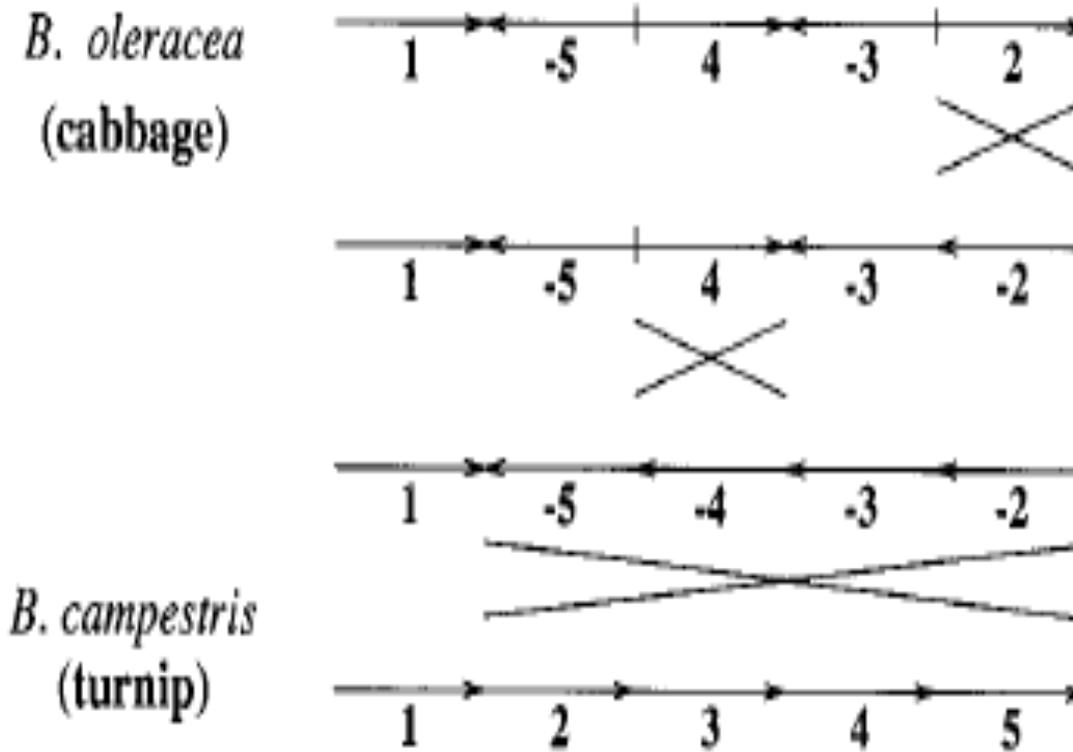
Navet vs Chou : ordre différent des gènes dans le mtDNA

Comparaison de l'ordre des gènes :

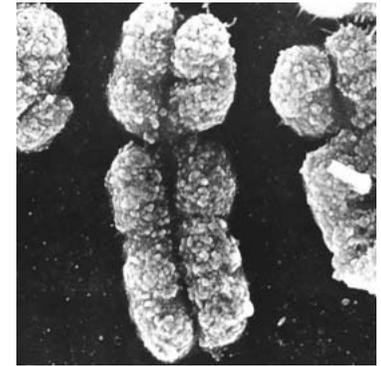


L'évolution est présente dans la divergence de l'ordre des gènes

Transformation du chou dans le navet



Opérations sur les chromosomes



Sur un seul chromosome

Délétion : une partie est perdue $abc \rightarrow ac$, ou a,b,c sont des sous-séquences

Insertion : une partie est ajoutée $ac \rightarrow abc$

Duplication : une partie est dupliquée $abc \rightarrow abbc$, tandem
 $abcd \rightarrow abcbcd$ ou $abcd \rightarrow acbcd$



Renversement / Inversion : une partie est renversée, tête-queue

$abc_1c_2c_3c_4de \rightarrow ab-c_4-c_3-c_2-c_1de$

Transposition : deux parties adjacentes échange leur place : $abcd \rightarrow acbd$

Cette opération est probablement la plus rare parce qu'elle demande au moins 3 points de rupture sur le chromosome

Transversale : deux parties échangent leur position et l'une est renversée

On connaît pas comment ces opérations sont physiquement implémentées dans la cellule.

Si deux régions sur le chromosome sont suffisamment similaires, alors on pense qu'elles puissent s'hybrider (de façon similaire a deux séquences d'ADN formant une double hélice). Quand elles s'hybrident il y a la formation d'une boucle. Cette boucle peut être éliminée (**délétion**), ou sa direction peut changer (**inversion**).

Une **insertion** peut se réaliser quand une nouvelle partie d'ADN rentre dans la cellule (pendant une infection virale ou un transfert horizontal) et elle se recombine avec l'ADN de la cellule.

Une **duplication** peut se réaliser pendant la mitose par erreur.

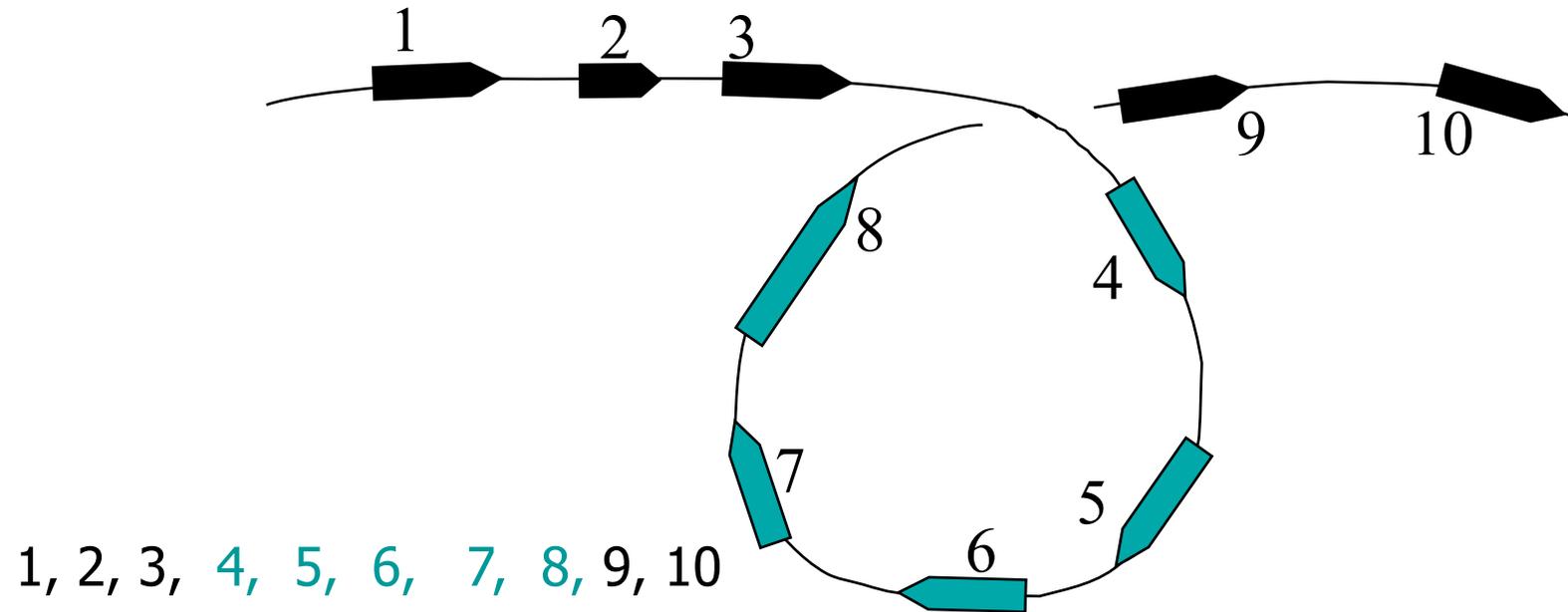
Les **transpositions** sont rares, et on n'a pas d'idée sur leur formation.

Exemple de Réarrangements

Renversement/Inversion

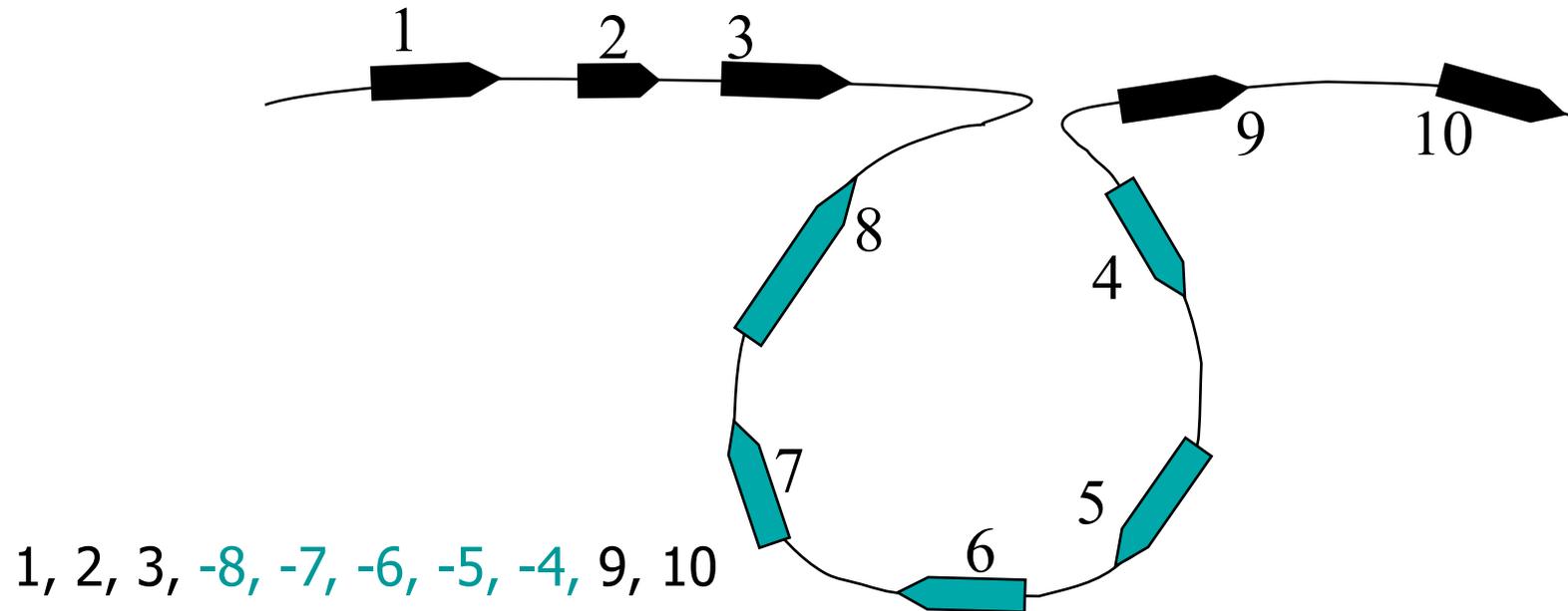
1 2 3 4 5 6  1 2 -5 -4 -3 6

Renversement



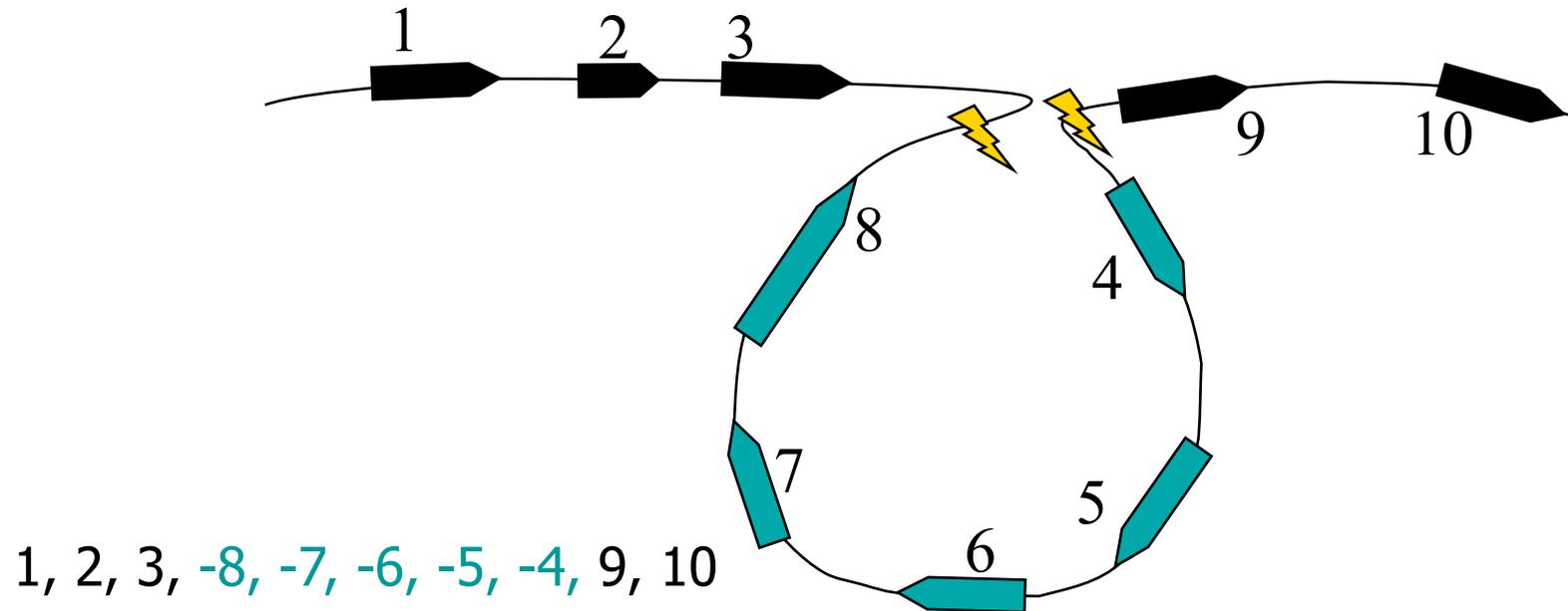
- Les blocs représentent les gènes conservés.

Renversement



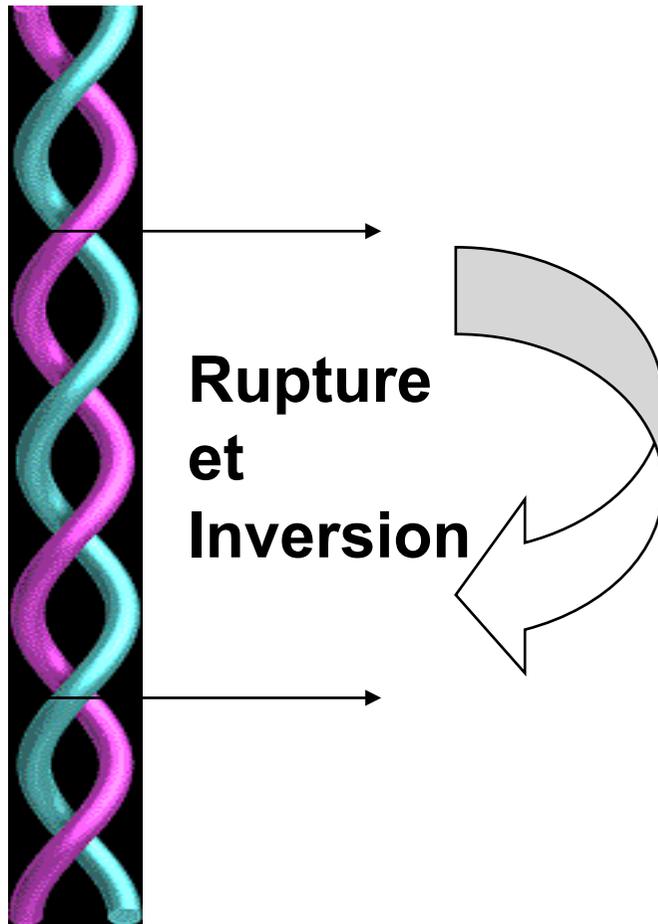
- Les blocs représentent les gènes conservés.
- Dans le cours de l'évolution ou dans des contextes cliniques, les blocs 1,...,10 peuvent être mal lus comme 1, 2, 3, -8, -7, -6, -5, -4, 9, 10.

Renversement et points de rupture



Le renversement a introduit deux *points de rupture* ⚡ (boulevserement dans l'ordre).

Renversement: opération sur deux brins



5' ATG CCTGTA CTA 3'

3' TAC GGACAT GAT 5'

5' ATG TACAGG CTA 3'

3' TAC ATGTCC GAT 5'

Opérations sur deux chromosomes:

Translocation : deux chromosomes échangent leurs queues.

Attention: pas toutes les translocations sont possibles. Un chromosome contient une partie appelée **centromère** qui est cruciale pendant le processus de division de la cellule. Le centromère habituellement se trouve localisé au milieu du chromosome, et si après translocation sa position est perdue dans l'un des chromosomes, la cellule peut mourir.

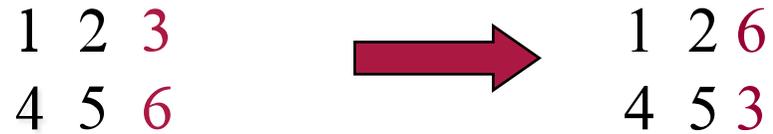
Fusion : deux chromosomes se mélangent en un chromosome seul

Fission : un chromosome se divise en deux chromosomes.

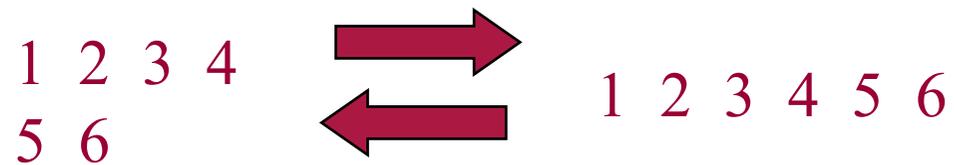
Il n'est pas connu ce que se passe avec les centromères dans les derniers deux cas

Exemples de Réarrangements

Translocation



Fusion



Fission

Réarrangement des génomes et évolution

Ils existent plusieurs facteurs qui rendent l'étude des réarrangement important pour l'évolution.

Les opérations indiquées sont **beaucoup plus rares** que les mutations ponctuelles. Cela implique que l'on peut essayer de tracer les réarrangements des génomes pendant leur évolution, plus en arrière que pour les mutations régulières.

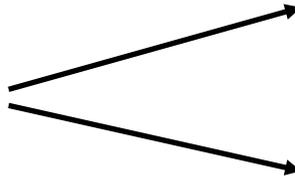
Par exemple, dans la comparaison de l'homme et de la souris il y a 80 millions d'années de distance évolutive entre les deux, mais seulement à peu près 200 opérations de réarrangement.

Il y a une **très faible possibilité que les mutations renversées affectent la même location** sur le génome une deuxième fois. Donc on a une faible ambiguïté dans l'interprétation de ces mutations.

Comme les réarrangements sont des mutations macroscopiques, on peut les localiser dans des portions assez grandes du génome et les analyser à une plus grande échelle. Cela aide à rendre plus accessible/clair notre compréhension du processus d'évolution.

Réarrangements du génome

Ancêtre inconnu
~ **75 million d'années**
(distance évolutive
entre homme et souris)



Souris (chrom. X)

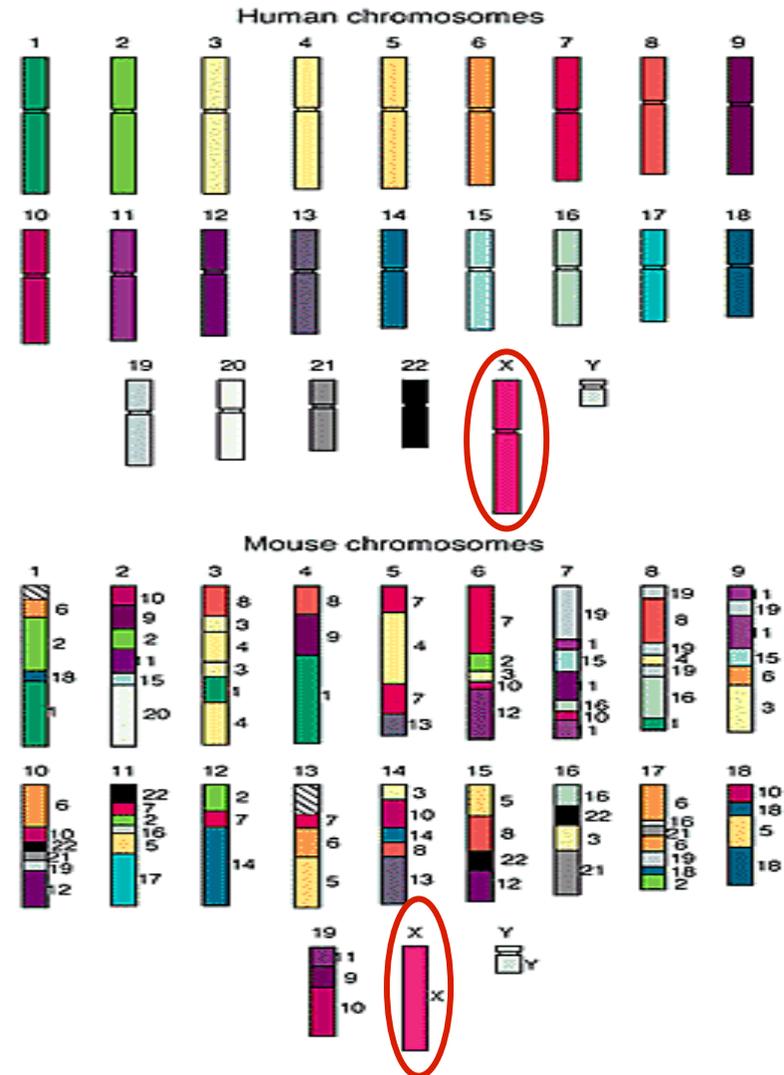
Homme (chrom. X)

Mais seulement 245 opérations de réarrangement chromosomiques

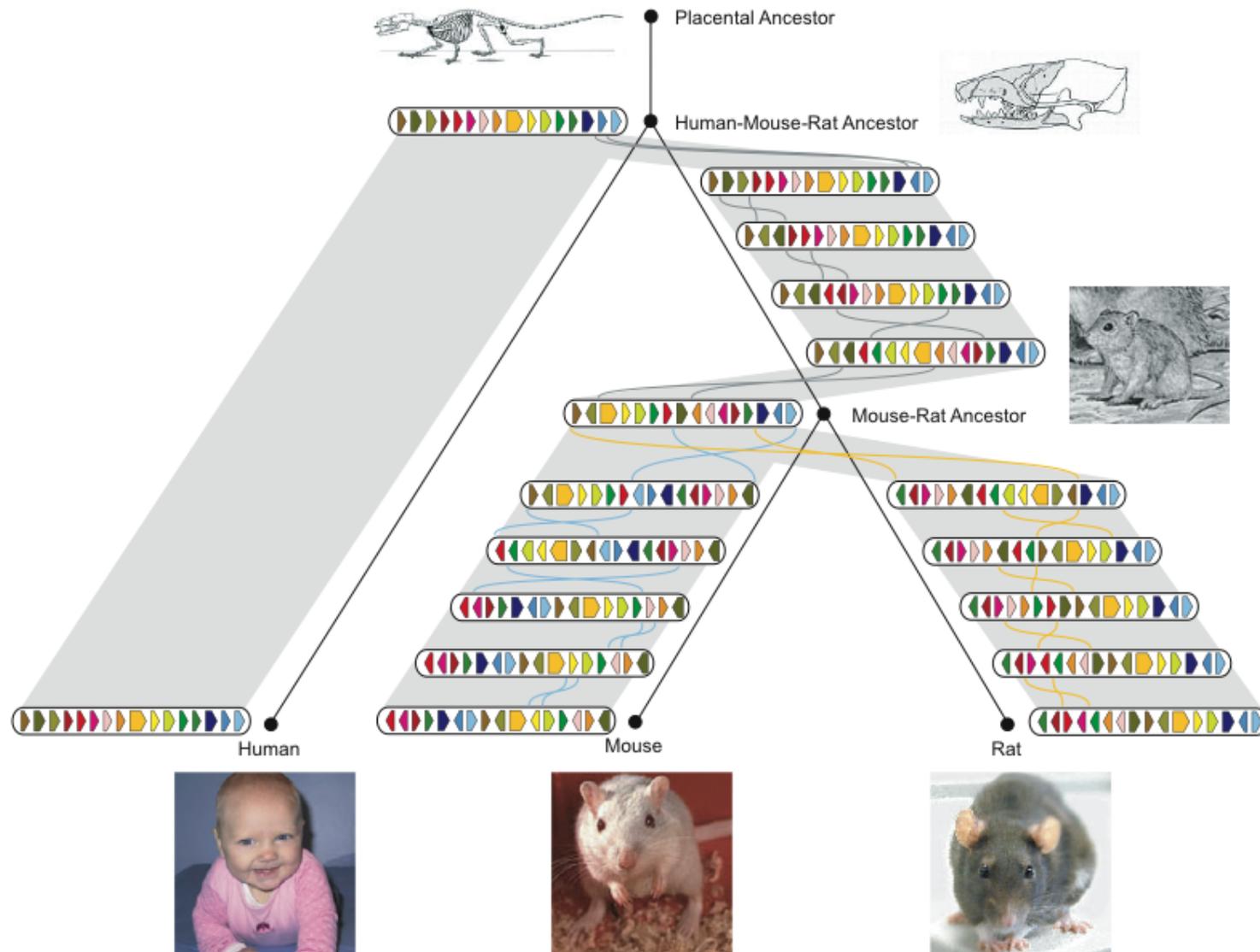
- Quels sont les blocs similaires et comment les trouver ?
- Quelle est l'architecture du **génom**e ancestrale?
- Quel est le **scénario évolutif** justifiant la transformation d'un génome dans un autre?

Comparaison des architectures génomiques: souris vs homme

- Homme et souris ont un génome similaire, mais leurs gènes sont ordonnés différemment
- ~245 réarrangements
 - Renversement
 - Fusions
 - Fissions
 - Translocation

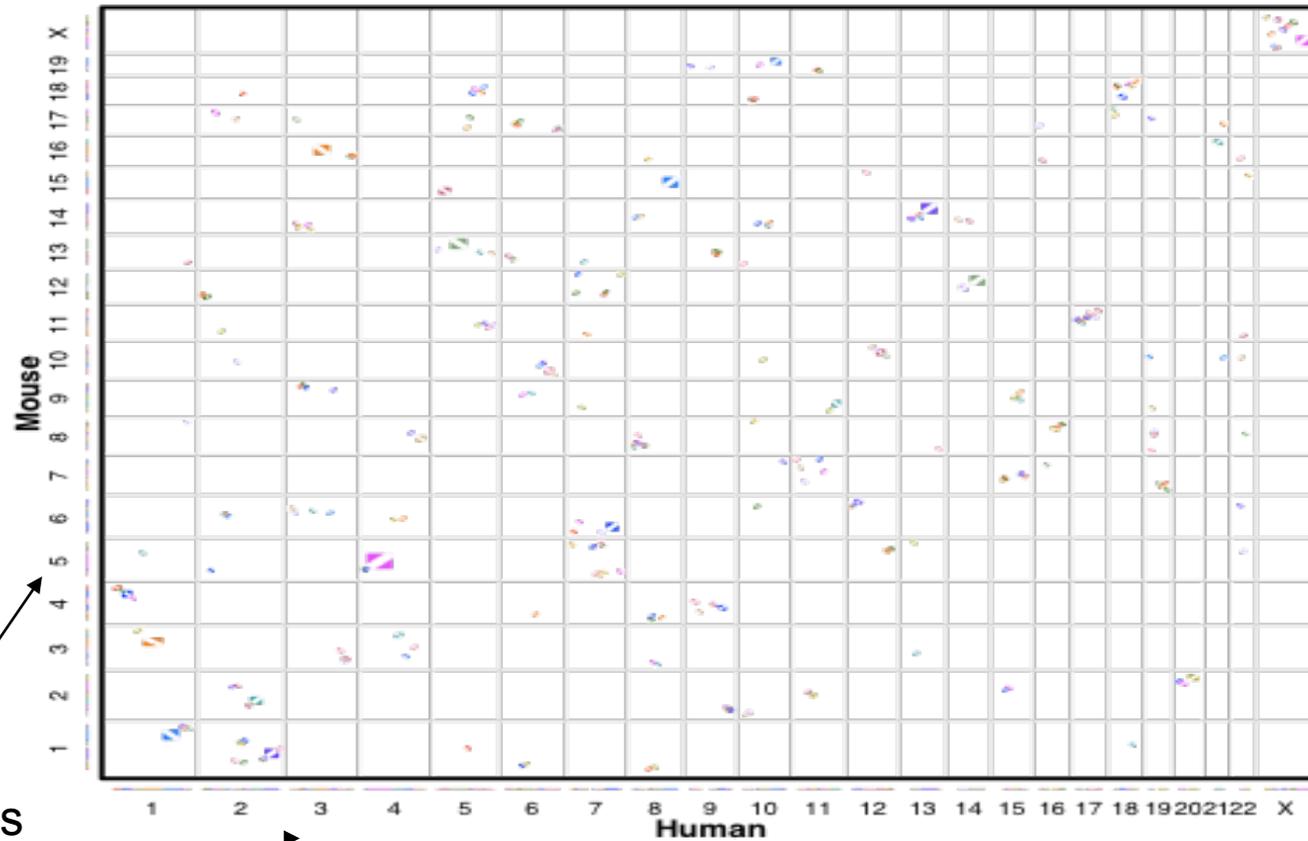


Chromosome X



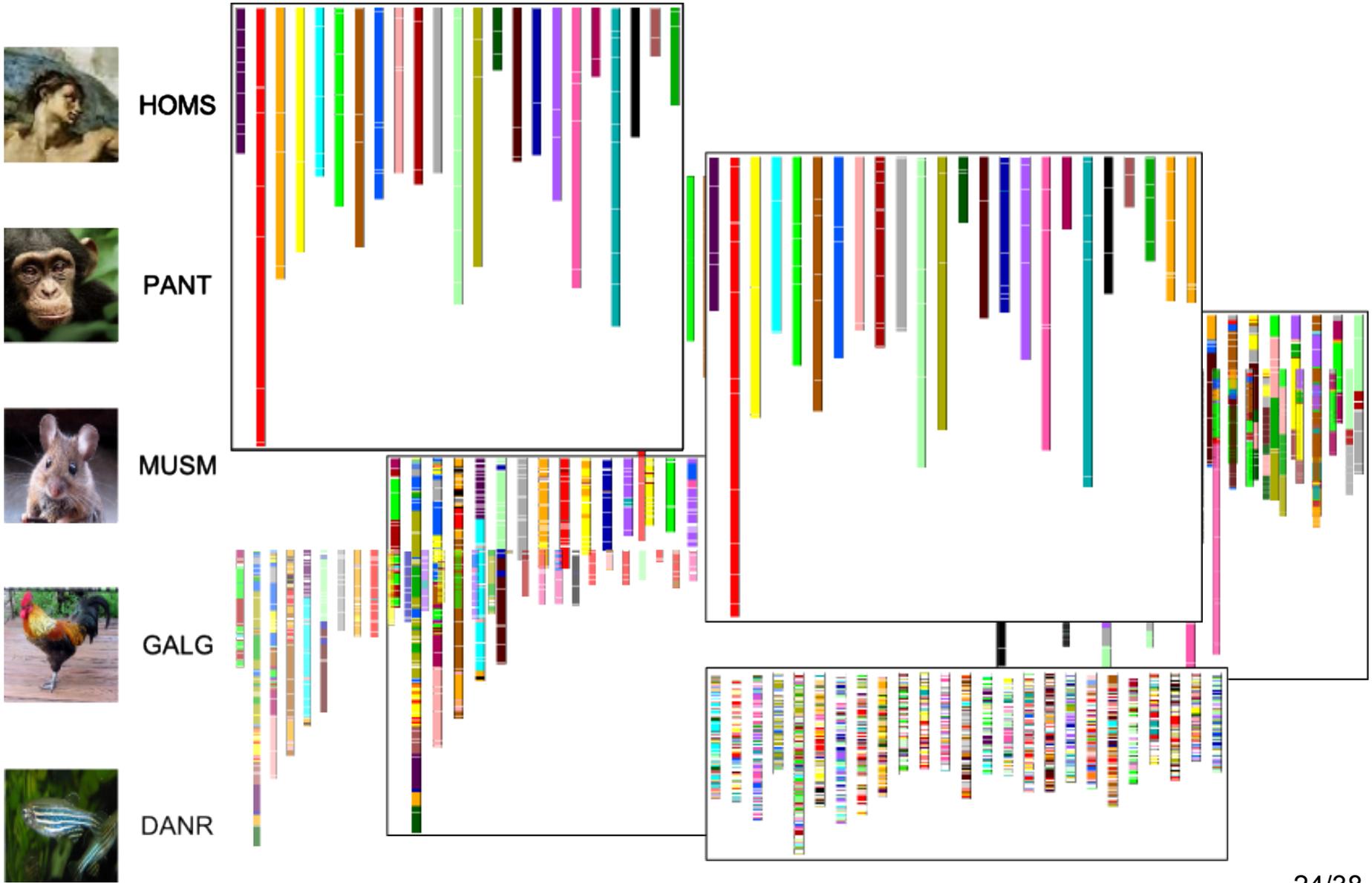
Comparaison de l'architecture du génome de l'homme et de la souris par dotplots

(b) Arrangement of human and mouse synteny blocks



chromosomes
A.Carbone | UPMC

Species evolution and chromosomal rearrangements



Evolution after Whole Genome Duplication

A



X



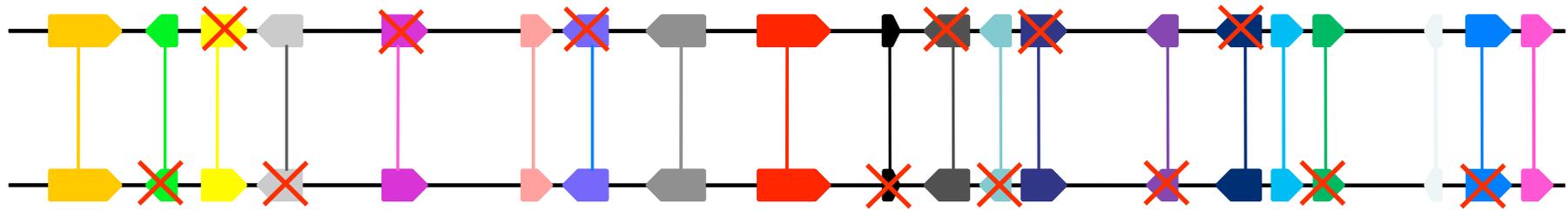
B



Speciation

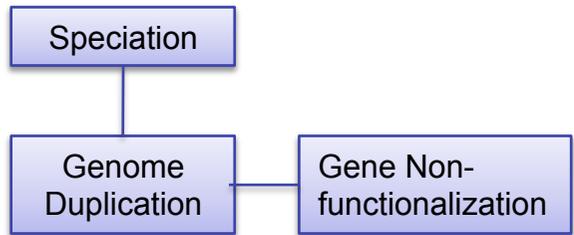
Evolution after Whole Genome Duplication

A1

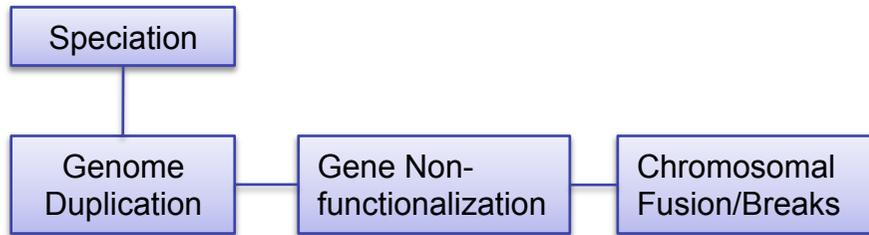
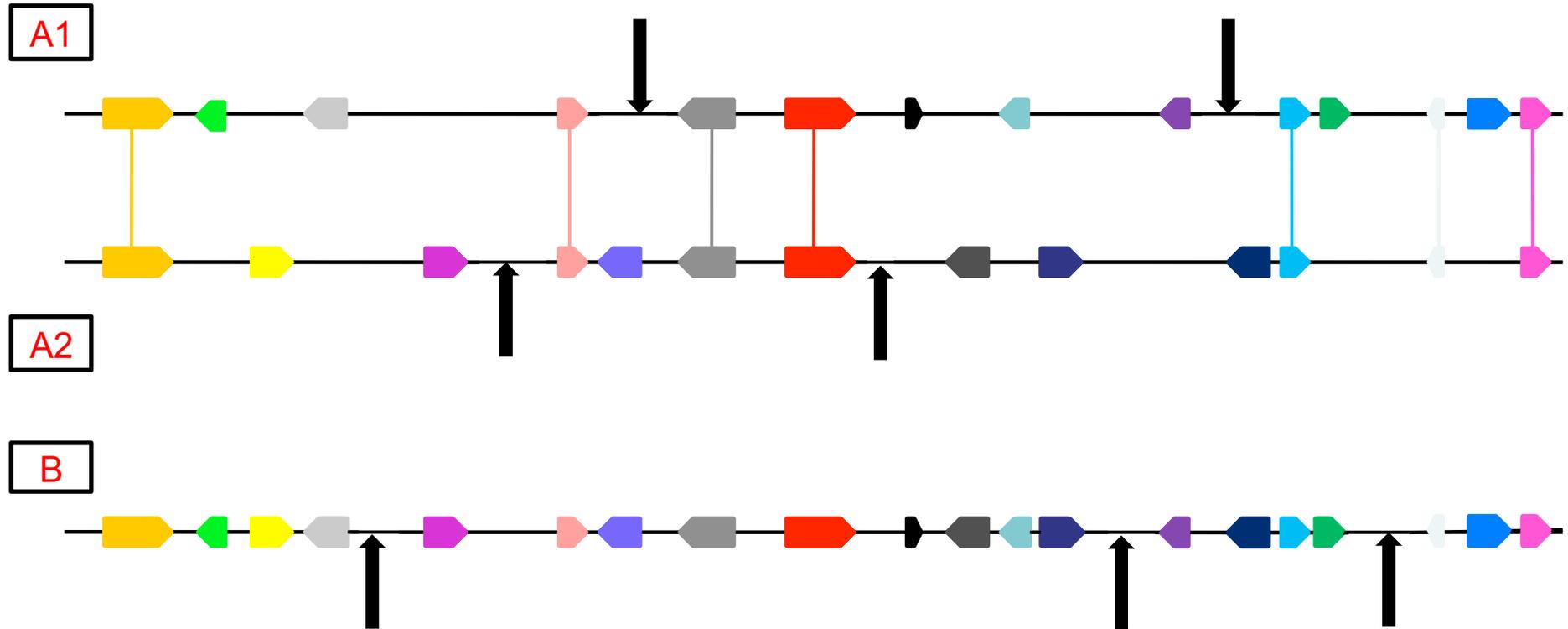


A2

B

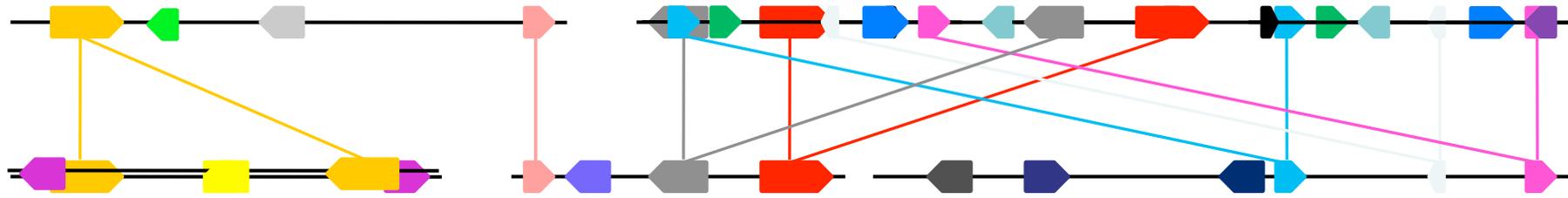


Evolution after Whole Genome Duplication



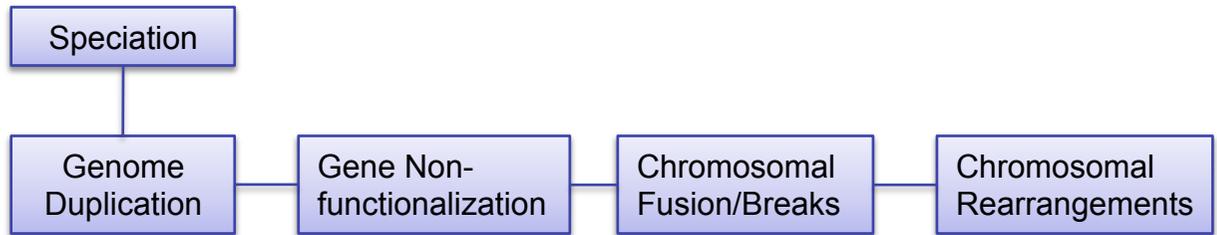
Evolution after Whole Genome Duplication

A1

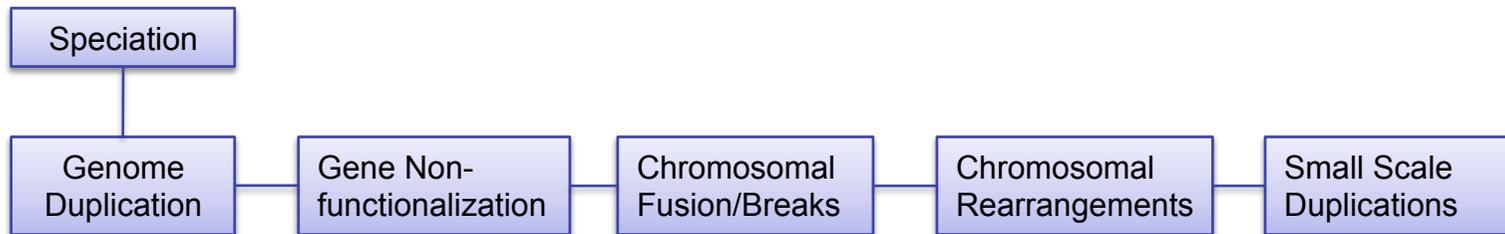
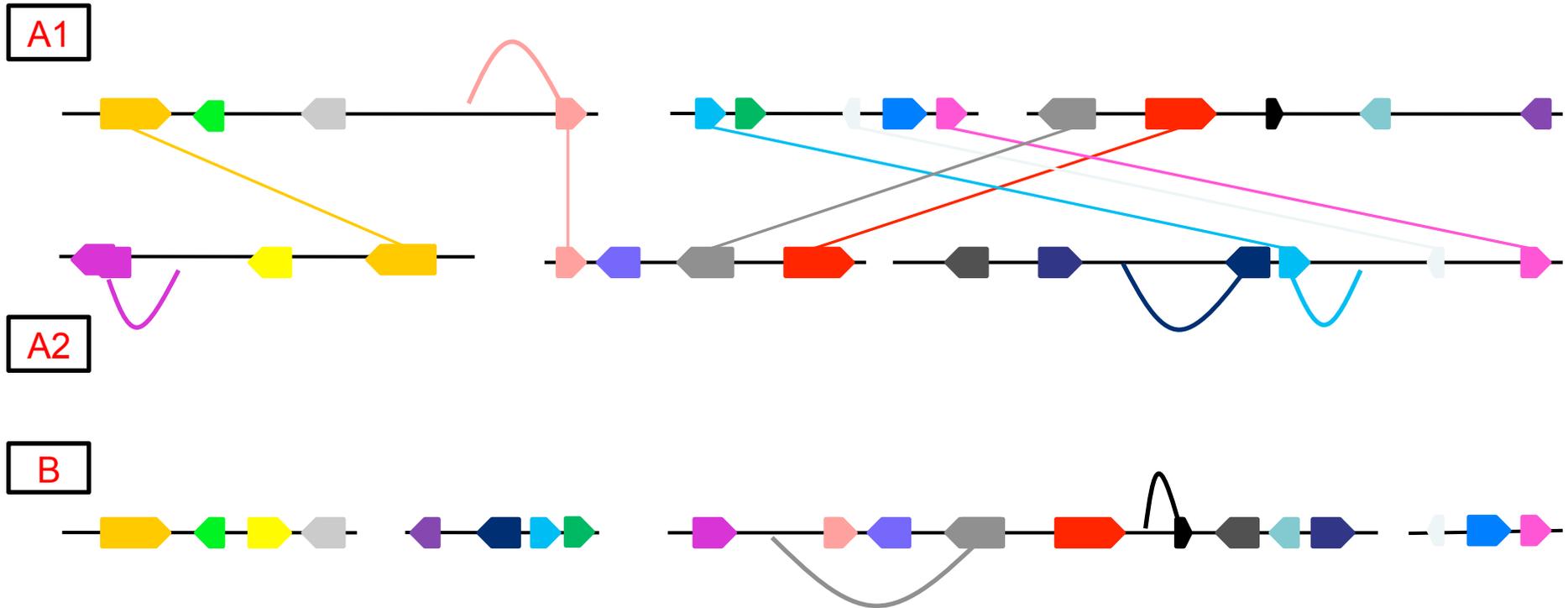


A2

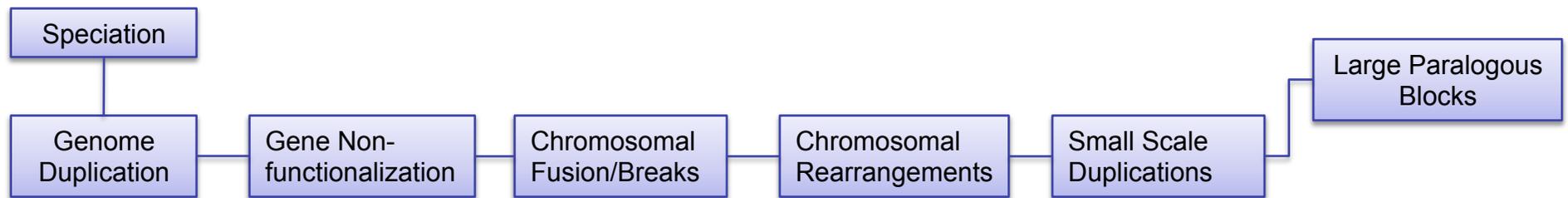
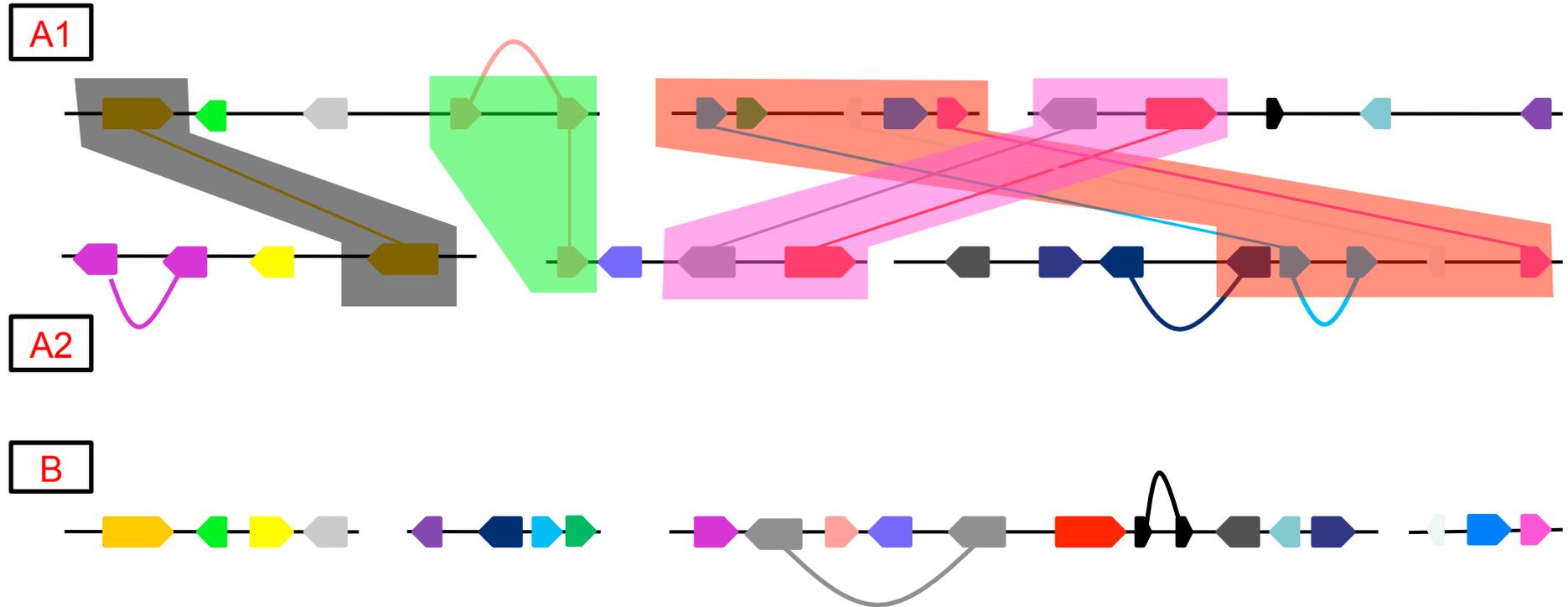
B



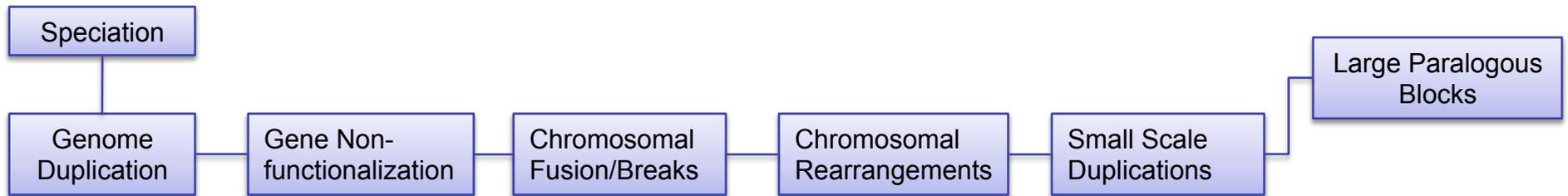
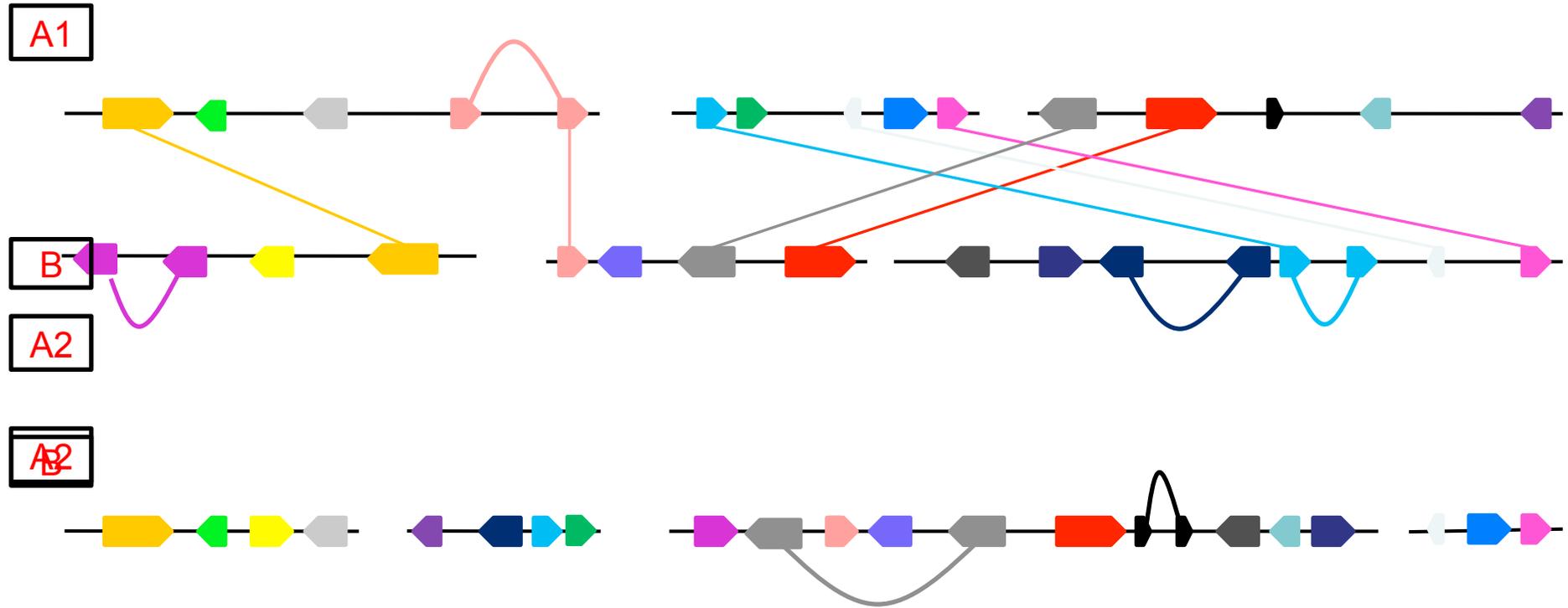
Evolution after Whole Genome Duplication



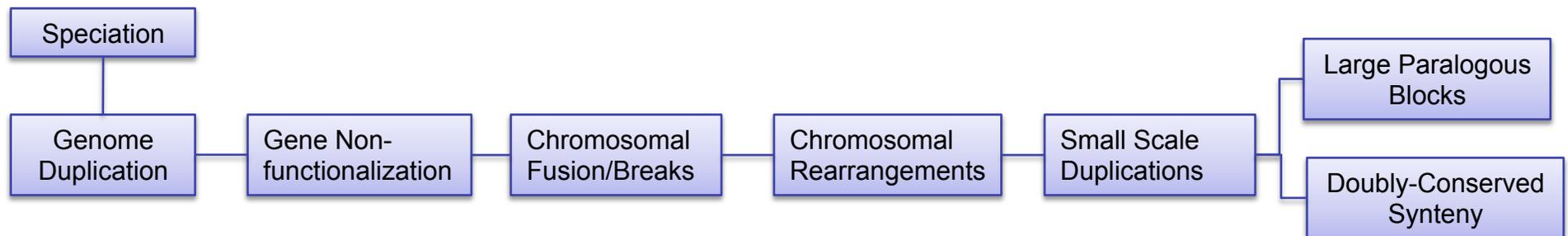
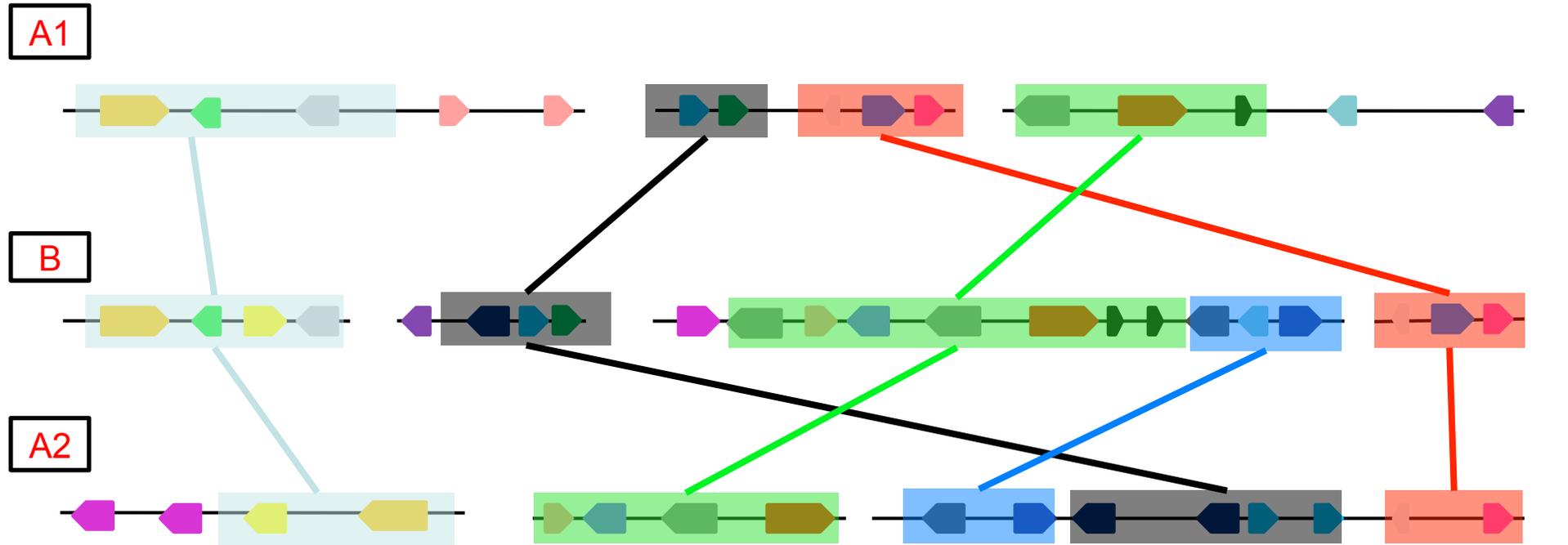
Evolution after Whole Genome Duplication



Evolution after Whole Genome Duplication



Evolution after Whole Genome Duplication



Courtesy of Param-Priva Singh

Renversement : un exemple

$\pi = 1\ 2\ \underline{3\ 4\ 5}\ 6\ 7\ 8$



$\rho(3,5)$

positions

1 2 5 4 3 6 7 8

Renversement : un exemple

$\pi = 1\ 2\ \underline{3\ 4\ 5}\ 6\ 7\ 8$



$\rho(3,5)$

positions

1 2 5 4 3 6 7 8



$\rho(5,6)$

1 2 5 4 6 3 7 8

Renversement et ordre des gènes

En générale, l'ordre des gènes est représenté par une permutation π :

$$\pi = \pi_1 \text{ ----- } \pi_{i-1} \underline{\pi_i \pi_{i+1} \text{ ----- } \pi_{j-1} \pi_j \pi_{j+1}} \text{ ----- } \pi_n$$

$\rho(i,j)$


$$\pi_1 \text{ ----- } \pi_{i-1} \pi_j \pi_{j-1} \text{ ----- } \pi_{i+1} \pi_i \pi_{j+1} \text{ ----- } \pi_n$$

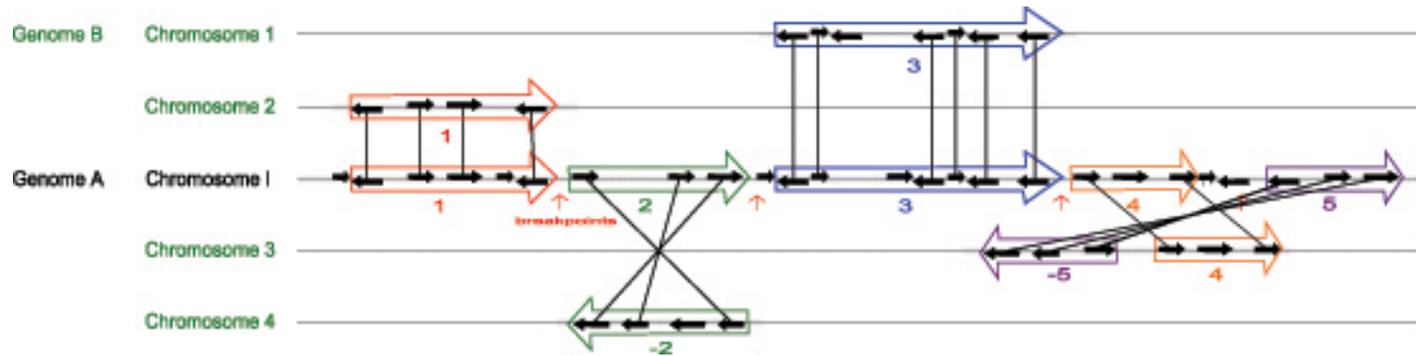
Le renversement $\rho(i, j)$ renverse les éléments de i à j dans π

Permutations sans signe

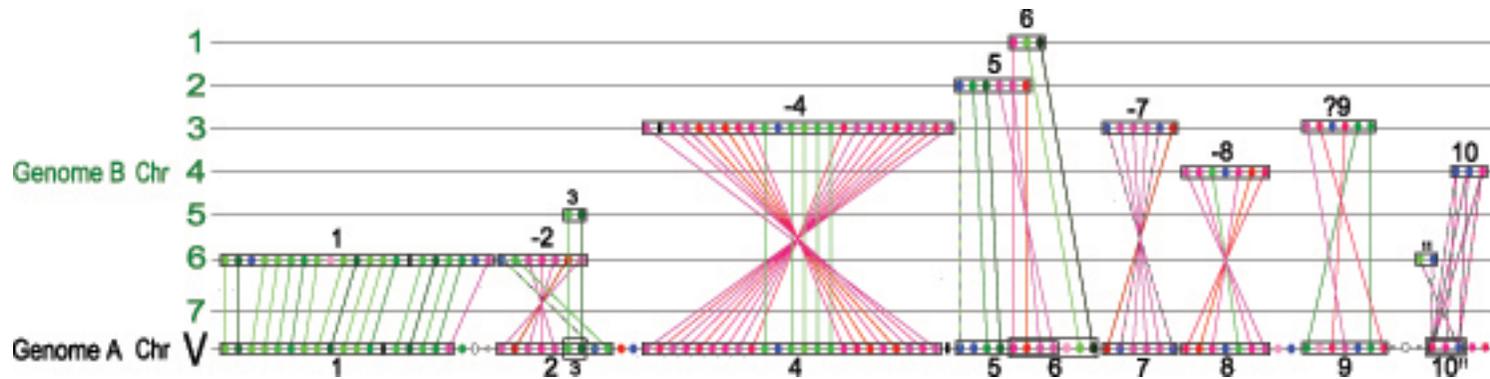
Etude des inversions sur les permutations sans signe.

Hypothèses:

1. On considérera un seul chromosome et on travaillera sous deux hypothèses:
 - Il est possible d'**identifier les gènes sans ambiguïté** sur les chromosomes et d'identifier leurs homologues dans d'autres espèces.
 - **Tous les gènes sont différents**
2. Les **gènes** d'une séquence seront représentés par les éléments de la permutation et la **direction de chaque gène est oubliée**.
3. **L'ordre des gènes** (ou de leurs homologues), possiblement différent dans des espèces différentes, est une permutation de ces gènes (ou de leurs homologues):
une permutation $\pi = (\pi_1, \dots, \pi_n)$ de séquences sans signe d'entiers différents représentera des ordres de gènes différents



Cas idéal (à plusieurs chromosomes)



Cas réel chez les levures

Définition : une **transformation de renversement sur une séquence** est l'opération qui prend une sous-séquence et renverse l'ordre des éléments. Par exemple $1\underline{234}5 \rightarrow 14325$.

Définition : Une **distance de renversement** entre deux séquences est le nombre **minimale** de renversements nécessaires pour transformer l'une dans l'autre.

Exemple:

$$\pi_1 = (1 \underline{234} 5 6)$$

$$\pi_2 = (1 4 \underline{325} 6)$$

$$\pi_3 = (\underline{146} 5 2 3)$$

$$\pi_4 = (6 4 1 5 2 3)$$

Distance de renversement = 3

Les sous-mots soulignés correspondent au lieu de renversement

Problème (tri par renversions)

Entrée: une permutation π sur n éléments

Sortie: la distance de renversement entre π et la permutation identité ($\text{id} = (1 2 \dots n)$).

Problème (équivalent au précédent) :

Etant données deux permutations, trouver la suite de renversements la plus courte qui transforme l'une dans l'autre.

Entrée: Permutations π et σ

Sortie: Une suite de renversements r_1, \dots, r_t transformant π en σ , tel que t (distance de renversement entre π et σ) soit minimum.

Tri par renversement: 4 étapes

Step 0: π	2	4	3	5	8	7	6	1
Step 1:	2	3	4	5	8	7	6	1
Step 2:	5	4	3	2	8	7	6	1
Step 3:	5	4	3	2	1	6	7	8
Step 4: γ	1	2	3	4	5	6	7	8

Quelle est la distance de renversement pour cette permutation ? Peut-on la trier en 3 étapes ?

Plusieurs solutions ont été proposées:

Algorithme de 2-approximation	(Kececioglu and Sankoff, 1995)
Algorithme de 1.75-approximation	(Bafna and Pevzner, 1996)
Algorithme de 1.5-approximation	(Christie, 1998)
Algorithme de 1.375-approximation	(Hannenhalli, 1998)
Démonstration de NP-complétude	(Caprara, 1999)

On commence par des solutions qui utilisent des algorithmes gourmands

Problème du renversement des crêpes (Pancake Flipping Problem)

- Le chef prépare une pile de crêpes pas triées en taille
- Le serveur veut les réarranger (de telle façon que la plus petite soit en haut et la plus large en bas)
- Il le fait en renversant plusieurs crêpes du haut, et en répétant cette idée le nombre nécessaire de fois.



Christos Papadimitriou et Bill Gates
renversent les crêpes

Problème (Renversement des crêpes) : Etant donnée une pile de n crêpes, quel est le nombre minimum de renversement pour obtenir une pile parfaite?

Entrée: Permutation π

Sortie: Une suite de **renversements préfixes** r_1, \dots, r_t transformant π dans la permutation identité tel que t est **minimum**.

Approche gourmand : nous avons besoin de 2 renversements préfixes au plus pour placer une crêpe à sa position correcte. Donc on a besoin de $2n - 2$ étapes au total au plus.

Bill Gates et Christos Papadimitriou ont montré dans les années 70s que toute permutation peut être triée en **au plus $5/3 (n + 1)$ renversements préfixes**. Par contre le problème du renversement des crêpes reste non résolu (cad, recherche de la solution ayant le plus petit nombre de renversements).

Tri par Renversement : un algorithme gourmand

- Si on tri $\pi = 1\ 2\ 3\ 6\ 4\ 5$, les premiers trois éléments sont déjà dans le bon ordre, donc il ne faut pas les bouger ni les séparer.
- La longueur du préfixe existant de π est dénoté $préfixe(\pi)$
 - $préfixe(\pi) = 3$
- Cette observation est la base de l'algorithme gourmand : **augmenter $préfixe(\pi)$ à chaque étape.**

Exemple: π sera triée comme suit

1 2 3 6 4 5
↓
1 2 3 4 6 5
↓
1 2 3 4 5 6

Le nombre d'étapes pour trier une permutation de longueur n est au plus $(n - 1)$

Algorithme gourmand : Pseudocode

SimpleReversalSort(π)

```
1 for  $i \leftarrow 1$  to  $n - 1$ 
2    $j \leftarrow$  position of element  $i$  in  $\pi$  (i.e.,  $\pi_j = i$ )
3   if  $j \neq i$ 
4      $\pi \leftarrow \pi \cdot \rho(i, j)$ 
5   output  $\pi$ 
6   if  $\pi$  is the identity permutation
7     return
```

Analyse de SimpleReversalSort

SimpleReversalSort ne garantit pas le plus petit nombre de renversements et prend 5 étapes sur $\pi = 6\ 1\ 2\ 3\ 4\ 5$:

- Etape 1: 1 6 2 3 4 5
- Etape 2: 1 2 6 3 4 5
- Etape 3: 1 2 3 6 4 5
- Etape 4: 1 2 3 4 6 5
- Etape 5: 1 2 3 4 5 6

La permutation de l'exemple ($\pi = 6\ 1\ 2\ 3\ 4\ 5$) peut être triée en 2 étapes

Step 1: 5 4 3 2 1 6

Step 2: 1 2 3 4 5 6

Donc, SimpleReversalSort(π) n'est pas optimale

Algorithmes optimaux sont inconnus; algorithmes **approximant** la solution optimale sont utilisés.

Ces algorithmes trouvent des **solutions approximées** plutôt que des solutions optimales. Le **ratio d'approximation** d'un algorithme A **sur l'entrée π** est:

$$A(\pi) / \text{OPT}(\pi)$$

ou

$A(\pi)$ – solution produite par l'algorithme A

$\text{OPT}(\pi)$ – solution optimale du problème

Le **ratio d'approximation** (**garantie de performance**) **de l'algorithme A**:

max/min du ratio d'approximation sur toutes les entrées de taille n

A.Carbone - UPMC

$$\max_{|\pi|=n} A(\pi) / \text{OPT}(\pi) \quad \text{pour les algorithmes A de minimization}$$
$$\min_{|\pi|=n} A(\pi) / \text{OPT}(\pi) \quad \text{pour les algorithmes A de maximization} \quad 47$$

Adjacences

$$\pi = \pi_1 \pi_2 \pi_3 \dots \pi_{n-1} \pi_n$$

Une paire d'éléments π_i et π_{i+1} est **adjacente** si

$$\pi_{i+1} = \pi_i \pm 1$$

Par exemple:

$$\pi = 1 \quad 9 \quad \underline{3} \quad \underline{4} \quad \underline{7} \quad \underline{8} \quad 2 \quad \underline{6} \quad 5$$

(3, 4), (7, 8) et (6,5) sont des paires adjacentes.

Points de rupture

Définition : un **point de rupture** est n'importe quel point dans la séquence où deux nombres adjacents ne sont pas consécutifs ($|\pi_i - \pi_{i+1}| \neq 1$).

Exemple : dans la séquence 123|654 il y a un point de rupture entre 3 et 6.

Exemple:

$$\pi = 1 \mid 9 \mid 3 \mid 4 \mid 7 \mid 8 \mid 2 \mid 6 \mid 5$$

Paires (1,9), (9,3), (4,7), (8,2) et (2,6) forment les points de rupture de la permutation π

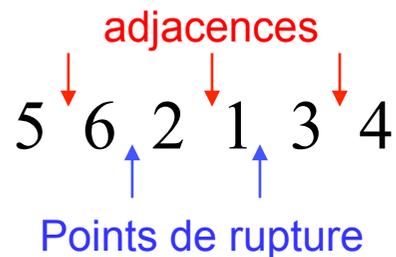
$b(\pi)$ - # points de rupture dans la permutation π

Adjacences et points de rupture

- Une **adjacence** - paire d'éléments adjacents **consécutifs**
- Un **point de rupture** - paire d'éléments adjacents **non consécutifs**

Exemple:

$$\pi = 5 \ 6 \ 2 \ 1 \ 3 \ 4$$



Attention: extension des permutations

On insère deux éléments $\pi_0 = 0$ et $\pi_{n+1} = n+1$ aux extrémités de π

Exemple:

$$\pi = 1 \mid 9 \mid 3 \mid 4 \mid 7 \mid 8 \mid 2 \mid 6 \mid 5$$



Etendre avec 0 et 10

$$\pi = 0 \mid 1 \mid 9 \mid 3 \mid 4 \mid 7 \mid 8 \mid 2 \mid 6 \mid 5 \mid 10$$

Noter: Un nouveau point de rupture a été introduit

Lemme (distance de renversement et points de rupture) : chaque renversement élimine au plus 2 points de rupture.

$$\pi = 2 \ 3 \ 1 \ 4 \ 6 \ 5$$

$$0 \ | \ \underline{2 \ 3} \ | \ 1 \ | \ 4 \ | \ 6 \ 5 \ | \ 7$$

$$b(\pi) = 5$$

$$0 \ 1 \ | \ \underline{3 \ 2} \ | \ 4 \ | \ 6 \ 5 \ | \ 7$$

$$b(\pi) = 4$$

$$0 \ 1 \ 2 \ 3 \ 4 \ | \ \underline{6 \ 5} \ | \ 7$$

$$b(\pi) = 2$$

$$0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7$$

$$b(\pi) = 0$$

Cela implique que:

$$\text{distance de renversement} \geq \# \text{points de rupture} / 2$$

On a dénoté $b(\pi)$, le nombre de points de rupture dans la permutation π .
Quand on réalise un renversement, en transformant π dans π' , on dénote $b(\pi')-b(\pi)$ par Δb .

Théorème (Kececioglu et Sankoff, 1995): $b(\pi)/2 \leq \lceil b(\pi)/2 \rceil \leq d(\pi) \leq n-1$

Preuve: Borne inférieure : un renversement peut effacer au plus deux points de rupture, et $d(\pi)$ est un entier.

Borne supérieure : on pourra utiliser au plus $n-1$ renversement pour arriver à créer une séquence souhaitée. Par exemple, considérer la séquence de transformations appliquée aux positions $(i \dots \pi^{-1}(i))$ pour reconstruire les préfixes.

Tri par renversement : un algorithme gourmand plus efficace

BreakPointReversalSort(π)

1 **while** $b(\pi) > 0$

2 Entre tous les possibles renversements, choisi le renversement r minimisant $b(\pi \cdot r)$

3 $\pi \leftarrow \pi \cdot r(i, j)$

4 **output** π

5 **return**

Problème: peut cet algorithme rentrer dans une boucle infinie? en éliminant des points de rupture on peut, en principe, en trouver d'autres

0 1 2 | 8 9 | 5 6 7 | 3 4

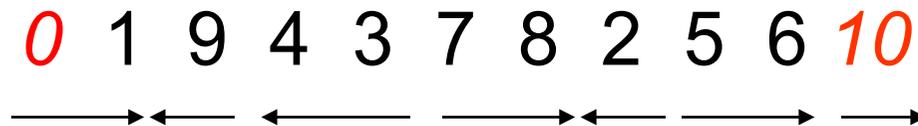
0 1 2 3 | 7 6 5 | 9 8 | 4

Bandes

Définition : Une **bande** est un intervalle entre deux points de rupture consécutifs dans une permutation.

Exemple : considérer la séquence 0 7 6 4 1 9 8 3 2 5 10; « 7 6 » est une bande.

- Bande décroissante: bande d'éléments en ordre décroissant (7 6, 9 8 et 3 2).
- Bande croissante: bande d'éléments en ordre croissant (7 8 et 5 6)



Une bande constituée par un seul élément peut être déclarée soit croissante soit décroissante. On choisit de la déclarer toujours décroissante à l'exception des bandes avec 0 et $n+1$.

Lemme (reduction du nombre des points de rupture): si $\pi \neq \text{id}$ contient une bande décroissante, alors il y a un renversement qui decremente $b(\pi)$ pour une valeur k , ou $k \geq 1$. Un tel renversement est appelé “bon renversement”.

Preuve:

1. Trouver une bande décroissante avec nombre minimale, disons K . K se trouve à la droite de la bande.
2. Trouver $(K-1)$ dans π ; il sera dans une bande croissante de longueur ≥ 1 , et en particulier il se situera à la droite ou à la gauche de la bande.
3. Renverser la séquence entière entre ces deux valeurs, K et $K-1$. On trouvera ces deux valeurs adjacentes. En ayant recombinié ces deux valeurs un point de rupture sera été éliminé.

Cas 1	<u>7 6 5 4</u> <u>2 3</u>	→	7 6 5 4 3 2
Cas 2	2 3 <u>7 6 5 4</u>	→	2 3 4 5 6 7 ...

QED

Ce lemme permet de définir l’**algorithme d’approximation suivant**: s’il existe une bande décroissante, trouver et réaliser un bon renversement ($\Delta b = -1$). Si non, renverser une bande croissante, en construisant alors une bande décroissante ($\Delta b = 0$). Cet algorithme 4-approxime l’algorithme optimale, et cela parce qu’il y a au plus $2b(\pi)$ renversements.

Exemple : 1ere étape algorithme

- For $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$

$0\ 1\ |4|\ 6\ 5|\ 7\ 8|\ 3\ 2|\ 9$ $b(\pi) = 5$

- Choisir une bande décroissante avec plus petit élément k dans π ($k = 2$ dans ce cas)

Exemple : 2eme étape algorithme

- For $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$

$0\ 1\ |4\ |6\ 5\ |7\ 8\ |3\ 2\ |9$ $b(\pi) = 5$

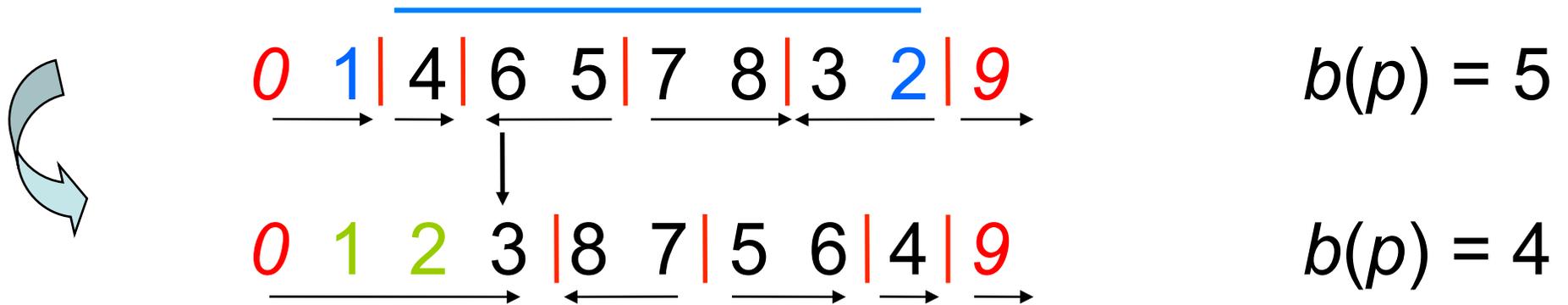
- Choisir une bande décroissante avec plus petit élément k dans π ($k = 2$ dans ce cas)
- Trouver $k - 1$ dans la permutation

Exemple : 3eme étape algorithmme

- For $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$

$0\ 1\ |4|\ 6\ 5\ |7\ 8|\ 3\ 2|\ 9$ $b(\pi) = 5$

- Choisir une bande décroissante avec plus petit élément k dans π ($k = 2$ dans ce cas)
- Trouver $k - 1$ dans la permutation
- Renverser le segment entre k et $k-1$



Réduction ultérieure du nombre de points de rupture

- S'il n'y a pas de bande décroissante, il n'y a peut être pas de renversements ρ qui réduisent le nombre de points de rupture (cad $b(\pi \cdot \rho) \geq b(\pi)$ pour tout renversement ρ).
- On a dit: en renversant une bande croissante (le # de points de rupture reste inchangé), on créera une bande décroissante à l'itération suivante, et le nombre de points de rupture sera réduit à l'itération suivante (théorème 1).

Exemple, continuation

Il n'y a pas de bande décroissante dans π , pour:

$$\begin{aligned} \pi &= \underline{0} \ 1 \ 2 \mid 5 \ 6 \ 7 \mid 3 \ 4 \mid \underline{8} \rightarrow & b(\pi) &= 3 \\ \pi \cdot \rho(6,7) &= \underline{0} \ 1 \ 2 \mid 5 \ 6 \ 7 \mid \underline{4} \ 3 \mid \underline{8} \rightarrow & b(\pi) &= 3 \end{aligned}$$

- ✓ $\rho(6,7)$ ne change pas le # de points de rupture
- ✓ $\rho(6,7)$ crée une bande décroissante et garantie que l'étape successive décrémentera le # de points de rupture.

ImprovedBreakpointReversalSort

ImprovedBreakpointReversalSort(π)

```
1 while  $b(\pi) > 0$ 
2   if  $\pi$  has a decreasing strip
6     Among all possible reversals, choose reversal  $\rho$ 
        that minimizes  $b(\pi \cdot \rho)$ 
4   else
11  Choose a reversal  $\rho$  that flips an increasing
    strip in  $\pi$ 
6    $\pi \leftarrow \pi \cdot \rho$ 
7   output  $\pi$ 
8  return
```

ImprovedBreakpointReversalSort: garantie de performance

ImprovedBreakPointReversalSort est un **algorithme d'approximation** avec une **garantie de performance d'au plus 4**

- Il élimine au moins un point de rupture chaque deux étapes; au plus $2b(\pi)$ étapes
- Ratio d'approximation: $2b(\pi) / d(\pi)$
- Algorithme optimal élimine au plus 2 points de rupture à chaque étape : $d(\pi) \geq b(\pi) / 2$
- Garantie de performance:

$$(2b(\pi) / d(\pi)) \geq [2b(\pi) / (b(\pi) / 2)] = 4$$

Quand on réalise un renversement, en transformant π dans π' , on dénote $b(\pi')-b(\pi)$ par Δb .

Lemme : s'il existe une bande décroissante et si chaque renversement qu'élimine un point de rupture donne une permutation sans bande décroissante, alors il existe un renversement qu'élimine 2 points de rupture.

Ce lemme permet de définir l'algorithme suivant:

Jusqu'à quand cela est possible:

1. Réaliser un bon renversement qui garantie une bande décroissante à l'étape d'après; cette transformation donne lieu à une permutation avec bande décroissante ($\Delta b=-1$).

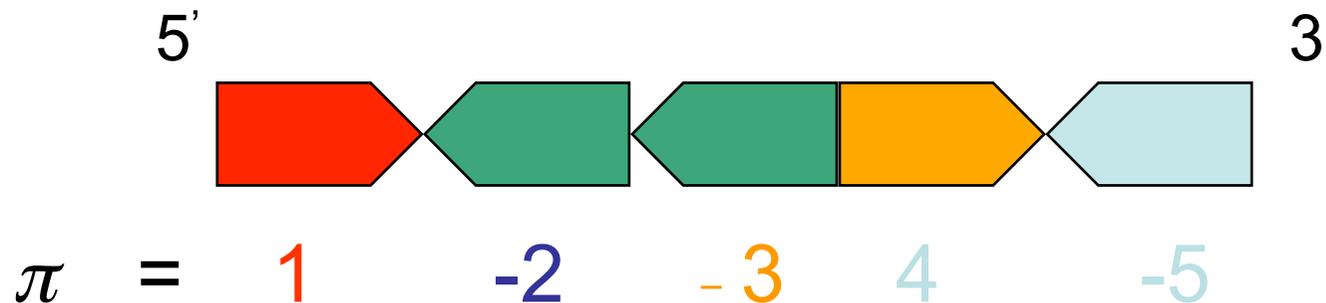
Si un tel renversement n'existe pas:

2. Réaliser un renversement avec $\Delta b=-2$, et en suite renverser une bande quelconque.

Cet algorithme amène à une **2-approximation** de l'ordonnancement avec renversement, car $\Delta b=-1$ en moyenne.

Permutations avec signe

- Jusqu'à maintenant, toutes permutations à trier étaient sans signe
- Mais les gènes ont une direction... donc on doit considérer des permutations avec signe



GRIMM Web Server

Les architectures des génomes réels sont représentées par des permutations avec signe.

Algorithmes efficaces pour trier des permutations signées ont été développés

GRIMM web server calcule les distances de renversement entre permutations avec signe.

GRIMM - Genome rearrangement algorithms

Multiple genome form

Source genome:

Destination genome:

Chromosomes: circular linear (directed) multichromosomal or undirected

Signs: signed unsigned

Or,

Formatting options

Report Style: One line per genome (chromosomes concatenated) One column (chromosomes separated) Two column before & after (chromosomes separated)

Horizontal Vertical Yes Show all chromosomes Only affected chromosomes

Highlighting style: Show all possible initial steps of optimal scenarios

Should operations (reversal, translocation, fission, fusion) be highlighted, and when?
 before after between/both no highlighting

Chromosome end for mat: numeric (10) subscripts (C₁₀) omit

Color coding: Genes should be colored according to their chromosome in which genome:
 source destination

[Click here or scroll up to enter new data or change options.](#)

3 chromosomes, 12 genes, 6 caps Multichromosomal Distance: 6

One optimal rearrangement scenario

Step	Description	Permutation
0	[Source]	-12 -7 -6 -5 -4 1 8 -11 -9 -10 -3 -2
1	Fusion	-12 -7 -6 -5 -4 1 2 3 10 9 11 8
2	Translocation	-12 -11 -9 -10 -3 -2 -1 4 5 6 7 8
3	Reversal	-12 -11 9 -10 -3 -2 -1 4 5 6 7 8
4	Reversal	-12 -11 10 -9 -3 -2 -1 4 5 6 7 8
5	Reversal	-12 -11 -10 -9 -3 -2 -1 4 5 6 7 8
6	Reversal [Destination]	-12 -11 -10 -9 1 2 3 4 5 6 7 8

GRIMM 1.04 by [Glenn Tesler](#), University of California, San Diego.
Copyright © 2001-2002, The University of California.
Contains code from [GRAPPA](#), © 2000-2001, The University of New Mexico and The University of Texas at Austin.

MGR 1.0 by [Guillaume Bourque](#), University of Southern California.
Copyright © 2001, University of Southern California.
Contains code from [Phylip](#) 3.5, Copyright © 1986-1995 by Joseph Felsenstein and the University of Washington.

[Click here for details on how to cite this in your work.](#)

Tri par renversement avec signe : 5 étapes

Step 0:	π	2	-4	-3	5	-8	-7	-6	1
Step 1:		2	3	4	5	-8	-7	-6	1
Step 2:		2	3	4	5	6	7	8	1
Step 3:		2	3	4	5	6	7	8	-1
Step 4:		-8	-7	-6	-5	-4	-3	-2	-1
Step 5:	γ	1	2	3	4	5	6	7	8

Tri par renversement avec signe : 4 étapes

Step 0: π	2	-4	-3	5	-8	-7	-6	1
Step 1:	2	3	4	5	-8	-7	-6	1
Step 2:	-5	-4	-3	-2	-8	-7	-6	1
Step 3:	-5	-4	-3	-2	-1	6	7	8
Step 4: γ	1	2	3	4	5	6	7	8

Tri par renversement avec signe : 4 étapes

Step 0:	π	2	-4	-3	5	-8	-7	-6	1
Step 1:		2	3	4	5	-8	-7	-6	1
Step 2:		-5	-4	-3	-2	-8	-7	-6	1
Step 3:		-5	-4	-3	-2	-1	6	7	8
Step 4:	γ	1	2	3	4	5	6	7	8

Quelle est la distance de renversement pour cette permutation ? Peut-on la trier en 3 étapes ?

Un algorithme pour ordonner les permutations avec signe

Une permutation avec signe est une permutation $\pi = (\pi_1, \dots, \pi_n)$ d'entiers $\{1, 2, \dots, n\}$, ou on a associé à chaque nombre un signe. Un renversement $\rho(i, j)$ de π transforme π dans

$$\pi' = \pi \cdot \rho(i, j) = (\pi_1, \dots, \pi_{i-1}, -\pi_j, -\pi_{j-1}, \dots, -\pi_i, \pi_{j+1}, \dots, \pi_n)$$

Cette définition est conforme à la définition usuelle du produit (composition) de permutations, définit comme suit:

$$\rho(i, j) = (1, \dots, i-1, -j, -j-1, \dots, -i, j+1, \dots, n)$$

Définition : la **distance de renversement** entre permutations est le nombre minimale de renversements nécessaires pour transformer l'une des permutations dans l'autre.

Problème (tri d'une permutation avec signe par renversement)

Entrée : Une permutation avec signe π .

Sortie : la distance de renversement entre π et la permutation avec signe identité
 $\text{id} = (+1 +2 \dots +n)$.

Une **borne supérieure** simple pour la distance de renversement est $2n$. Cela est dû au fait que l'on peut créer la séquence correcte (négligeant les signes) avec n renversements et on aura besoin de n renversement de plus pour changer les signes (renversements de longueur 1).

Application de ce problème à la comparaison à large échelle de génomes d'espèces différentes par blocs de synténie

Pour tester à large échelle la similarité entre génomes, une approche possible est la comparaison de l'ordre d'occurrence de gènes homologues dans les deux espèces.

Dobzhansky et Sturtevant (1935) ont montré les inversions chez la *Drosophila*. Palmer (1987, 1988) a démontré que des espèces différentes peuvent avoir (essentiellement) les mêmes gènes mais que l'ordre entre les gènes peut varier de façon importante entre espèces.

Représentation mathématique du problème:

- Gènes peuvent être pensés comme des points sur une droite étiquetés avec des entiers
- Les signes correspondent à la direction des gènes sur le génome
- La différence dans l'ordre des gènes entre génomes est étudiée par les renversements
- Le nombre de renversements correspond à la distance évolutive entre espèces.

Hannenhalli et Pevzner (1995) ont montré que le problème a une solution polynomiale, en $O(n^4)$. Berman et Hannenhalli (1996) ont trouvé un algorithme en $O(n^2\alpha(n))$, où $\alpha(n)$ est l'inverse de la fonction de Ackermann.

Kaplan, Shamir et Tarjan (1999) ont trouvé un algorithme en $O(n^2)$.

Application de ce problème à la comparaison à large échelle de génomes d'espèces différentes.

Pour tester à large échelle la similarité entre génomes, une approche possible est la comparaison de l'ordre d'occurrence de gènes identiques dans deux espèces.

Dobzhansky et Sturtevant (1935) ont montré les inversions dans la drosophile.
Palmer (1987, 1988) a démontré que des espèces différentes peuvent avoir (essentiellement) les mêmes gènes mais que l'ordre entre les gènes peut varier de façon importante entre espèces.

Représentation mathématique du problème:

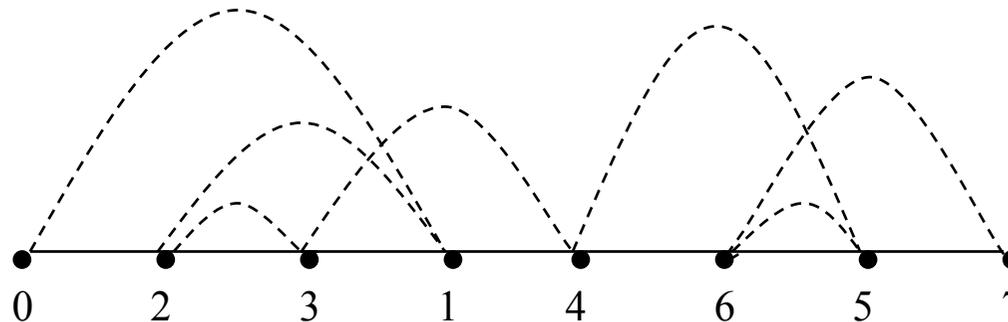
- Gènes peuvent être pensés comme des points sur une droite étiquetés avec des entiers
- Les signes correspondent à la direction des gènes sur le génome
- La différence dans l'ordre des gènes entre génomes est étudiée par les renversements
- Le nombre de renversements correspondent à la distance évolutive entre espèces.

Hannenhalli et Pevzner (1995) ont montré que le problème a une solution polynomiale en $O(n^4)$.
Berman et Hannenhalli (1996) ont trouvé un algorithme en $O(n^2\alpha(n))$, où $\alpha(n)$ est l'inverse de la fonction de Ackermann.

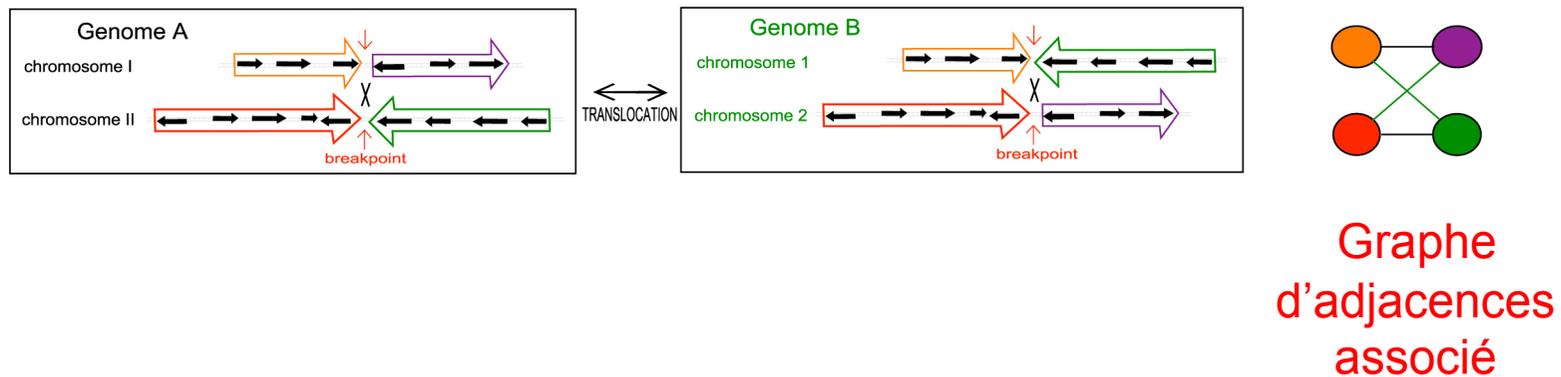
Kaplan, Shamir et Tarjan (1999) ont trouvé un algorithme en $O(n^2)$.

Graphe des points de rupture

- 1) Représenter les éléments de la permutation $\pi = 2\ 3\ 1\ 4\ 6\ 5$ comme noeuds dans le graphe (de gauche à droite)
- 2) Connecter les noeuds dans l'ordre donné par π avec des arêtes noir (chemin noir)
- 3) Connecter les noeuds dans l'ordre donné par $\text{id} = 1\ 2\ 3\ 4\ 5\ 6$ avec des arêtes grises (chemin gris)
- 4) Chevaucher les chemins noir et gris

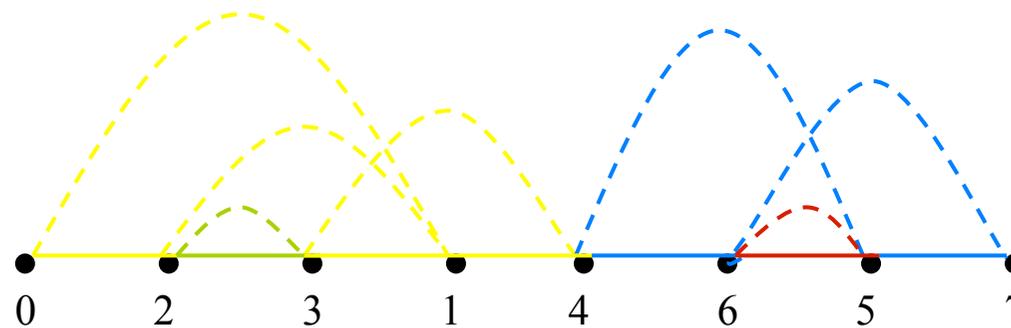


Transformation entre deux génomes qui défini les blocs



Une opération de translocation est associée à un cycle dans le graphe d'adjacences. On utilisera cette formalisation pour seules les opérations de renversement.

Décomposition maximale en cycles, où chaque cycle est une suite d'arcs différents et alternés (traits et droits).



Bafna et Pevzner (1996) ont démontré que

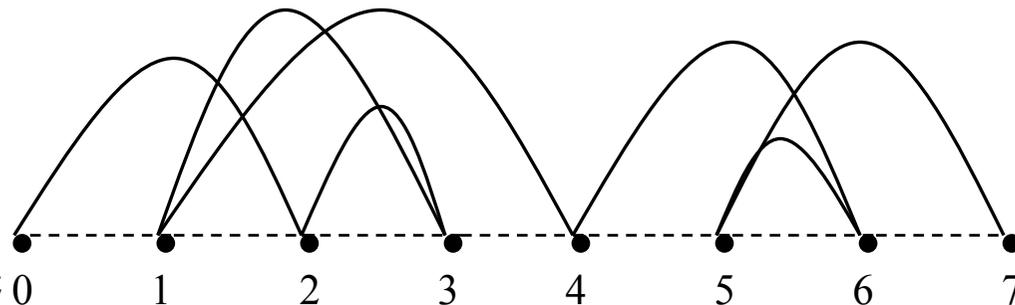
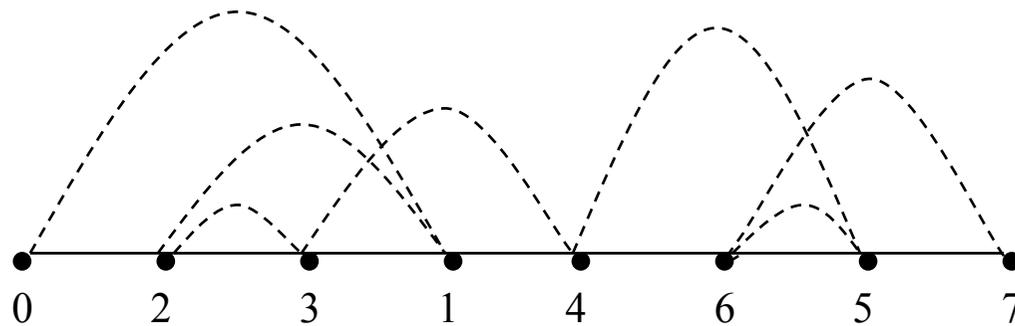
$$d(\pi) \geq n+1 - c(\pi)$$

Ce résultat constitue une amélioration de la borne

$$d(\pi) \geq b(\pi)/2$$

Deux représentations équivalentes du graphe des points de rupture

- Considérer le graphe des points de rupture suivant
- si on aligne le chemin gris (a la place du chemin noir) sur une ligne horizontale, alors on obtiendra le graphe suivant
- même s'ils semblent différents, il s'agit du même graphe

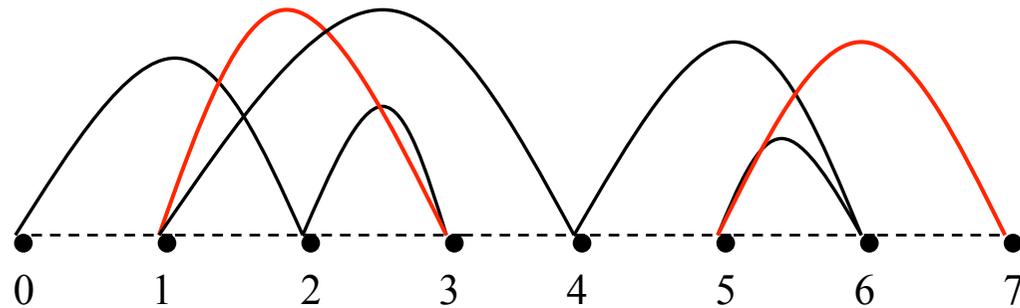


Qu'est-ce que l'effet d'un renversement?

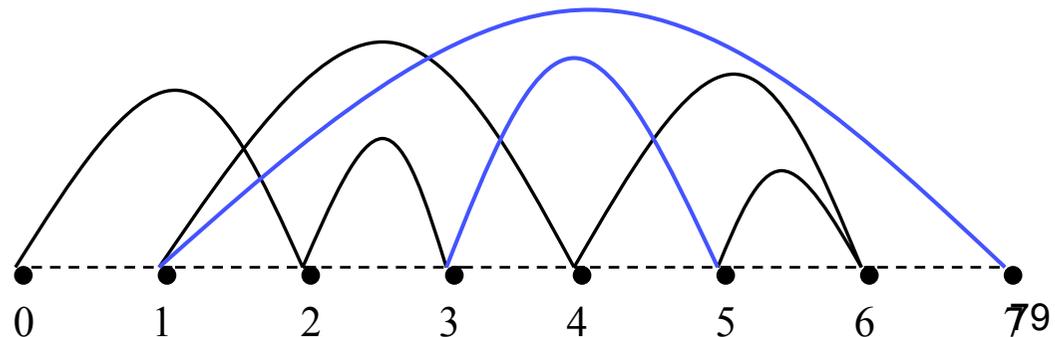
Comment un renversement change un graphe de points de rupture?

- les chemins gris restent les mêmes pour les deux graphes
- il y a un changement dans le graphe a ce point
- et un autre changement a ce point
- les arêtes noir ne sont pas affectées par le renversement et elle restent les mêmes dans les deux graphes

Avant: 0 2 3 1 4 6 5 7

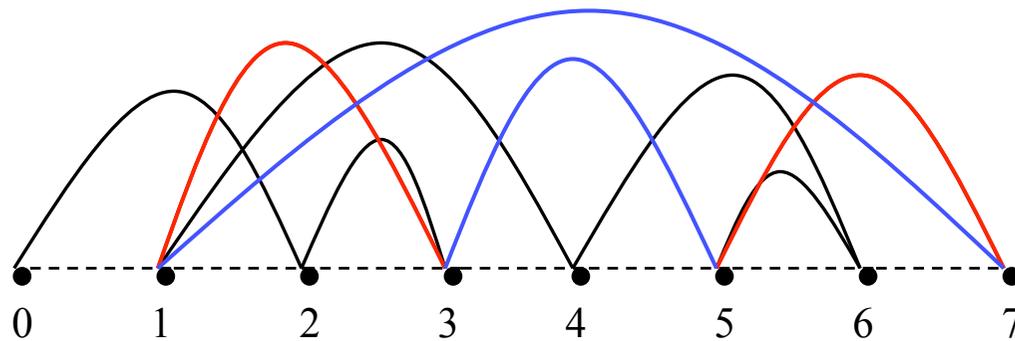


Après: 0 2 3 5 6 4 1 7



Un renversement affecte 4 arêtes dans le graphe de points de rupture

- Un renversement élimine 2 arêtes (rouge) et les remplace avec 2 nouvelles arêtes (bleue)



Effets d'un renversement

Case 1:

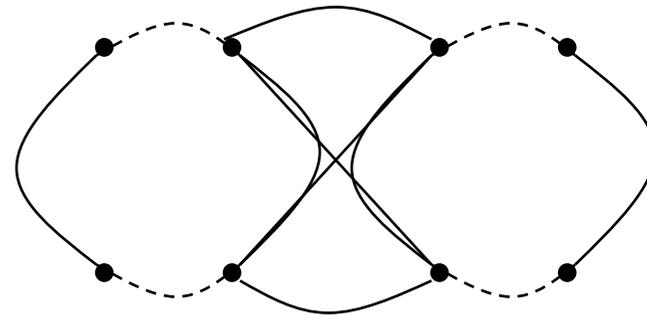
Les deux arêtes appartiennent au même cycle

- Eliminer les arcs noir centraux et les remplacer avec des nouveaux arcs noir (il y a deux manières pour les remplacer).
- (a) Après ce remplacement, ils existent maintenant 2 cycles a la place que 1 cycle
- (b) Ou après ce remplacement, il existe encore 1 cycle

Donc après le renversement $\rightarrow c(\pi p) - c(\pi) = 0$

Ceci est appelle *renversement propre* parce qu'il y a une *incrémentat* d'un cycle après le

A.Carbord *renversement.*



Effets de renversement (continuation)

Cas 2:

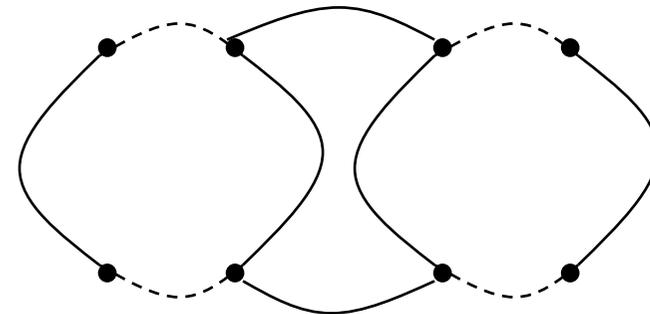
Les deux arêtes appartiennent a deux cycles différents

- Eliminer les arêtes noir centrales et les remplacer avec des nouvelles arêtes noir.
- Apres le remplacement, il existe 1 cycle a la place que 2 cycles

➔ $c(\pi\rho) - c(\pi) = -1$

De cette façon, pour chaque permutation π et renversement ρ ,

$$c(\pi\rho) - c(\pi) \leq 1$$



Distance de renversement et décomposition des cycles maximale

- Comme la permutation identité de taille n est une décomposition de $n+1$ cycles maximales, $c(\text{identity}) = n+1$
- $c(\text{identity}) - c(\pi)$ est le nombre de cycles que doit être “ajouté” à $c(\pi)$ quand on transforme π dans l’identité
- Basé sur le théorème précédent, au mieux après chaque renversement, la décomposition de cycles peut incrémenter de 1, alors:

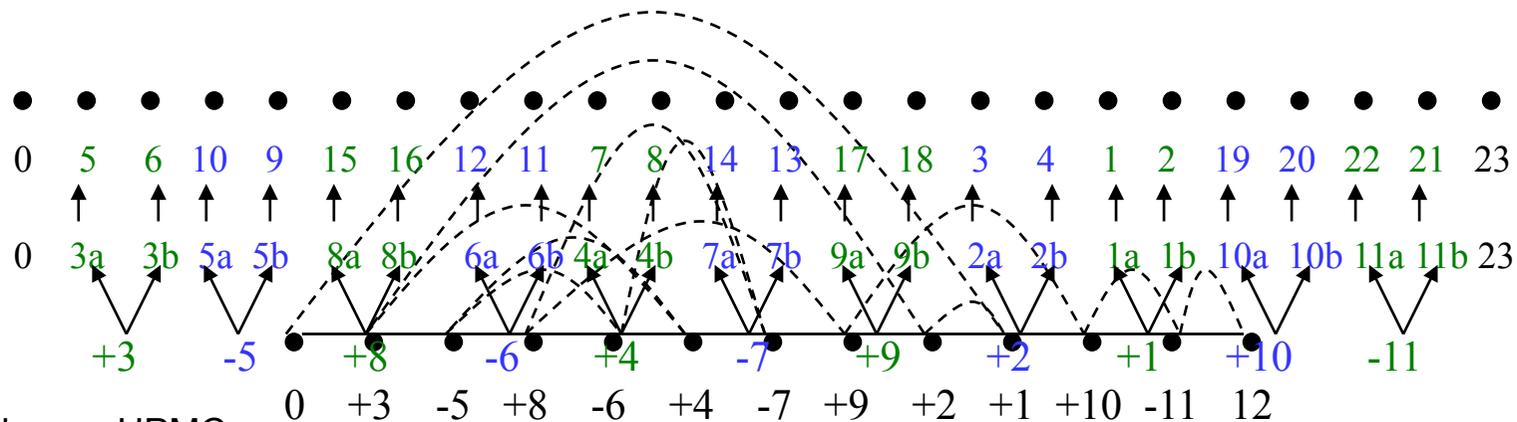
$$d(\pi) = c(\text{identity}) - c(\pi) = n+1 - c(\pi)$$

- Par contre, pas tous les renversements peuvent incrémenter la décomposition des cycles

➔ donc, $d(\pi) \geq n+1 - c(\pi)$

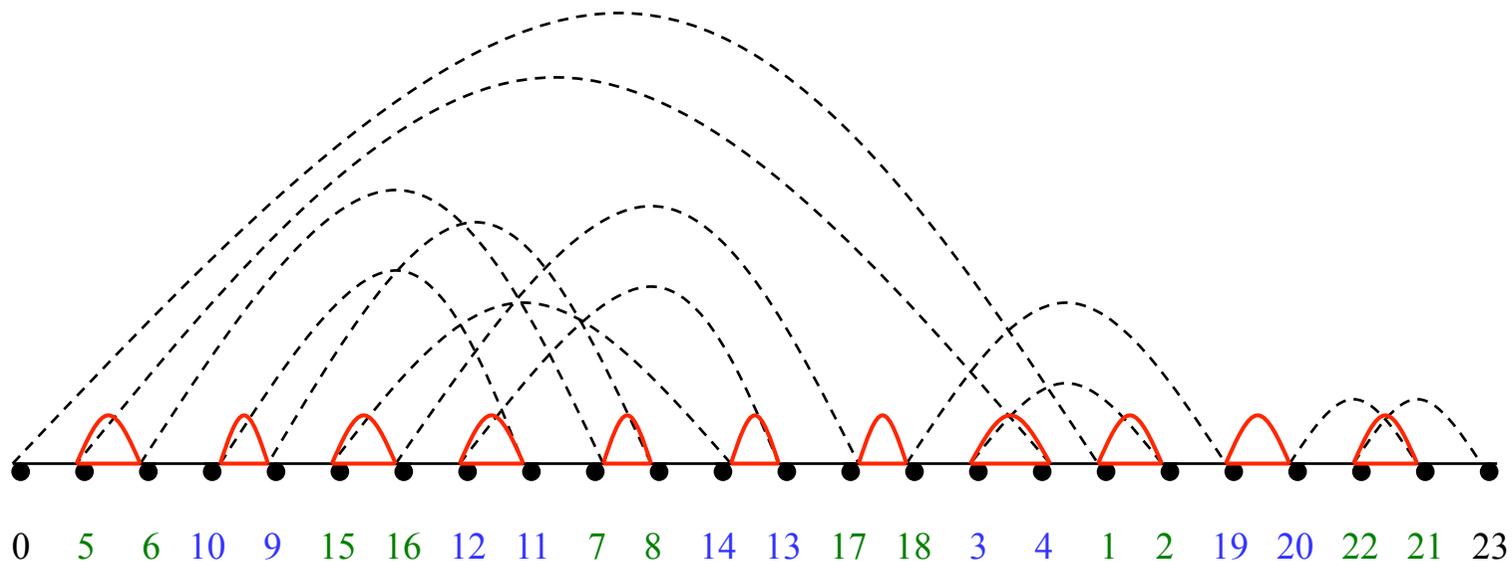
D'une permutation avec signe vers une permutation sans signe

- Commencer par construire un graphe de points de rupture avec signe
- Redéfinir chaque noeud x avec les règles suivantes:
 - Si le noeud x est **positif**, remplacer le noeud x avec le noeud $2x-1$ et le noeud $2x$ dans le même ordre
 - Si le noeud x est **négatif**, remplacer le noeud x avec le noeud $2x$ et le noeud $2x-1$ dans le même ordre
 - Les extensions des noeuds $x = 0$ et $x = n+1$ sont laissées comme avant



D'une permutation avec signe à une sans signe (cont)

- Construire le graphe des points de rupture précédent
- Noter les cycles alternés dans le graphe entre chaque paire de noeuds introduite
- Comme ces cycles proviennent du même noeud avec signe, on ne réalisera pas des renversements sur les deux paires; donc, ces cycles peuvent être éliminés du graphe

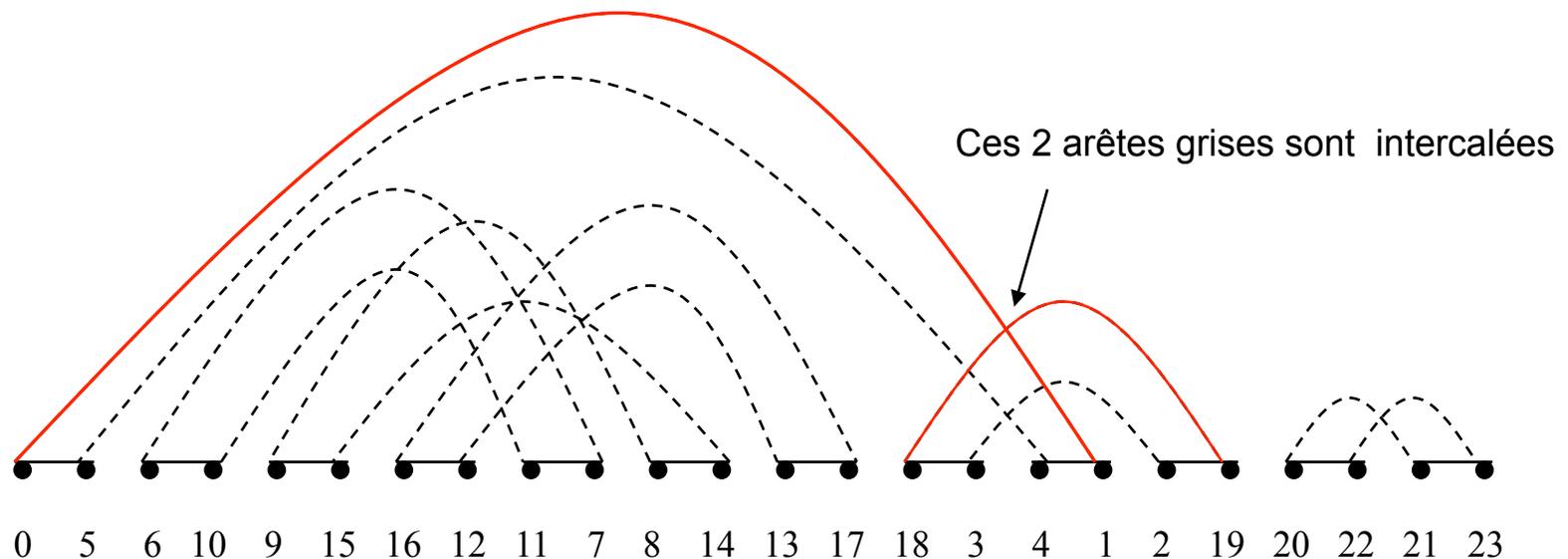


Arêtes intercalées

- Les arêtes intercalées sont des arêtes grises qui croisent l'une l'autre

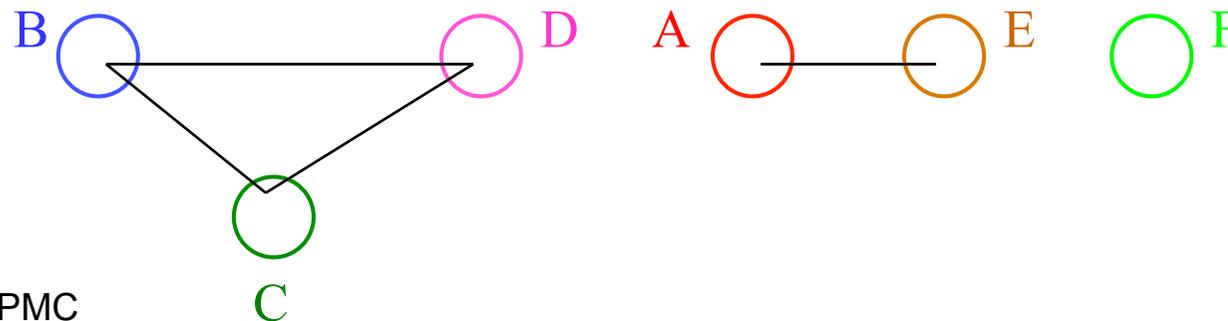
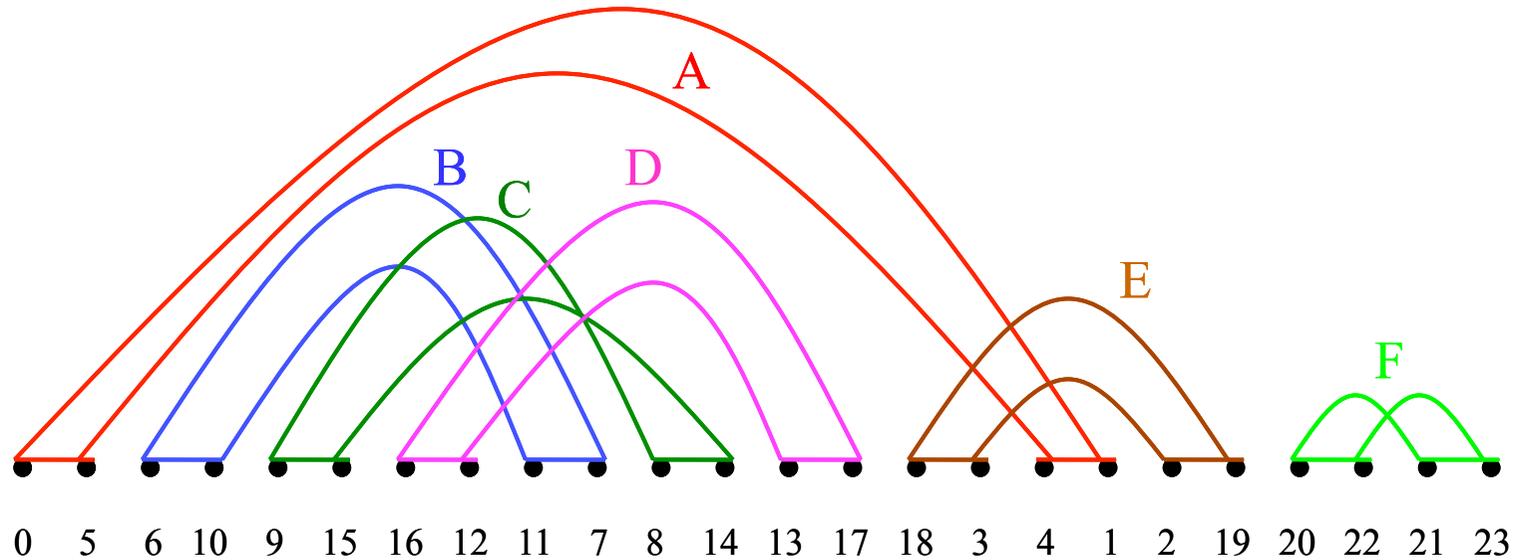
Exemple: Arêtes (0,1) et (18, 19) sont intercalées

- Les cycles sont intercalés s'ils ont une arête intercalée



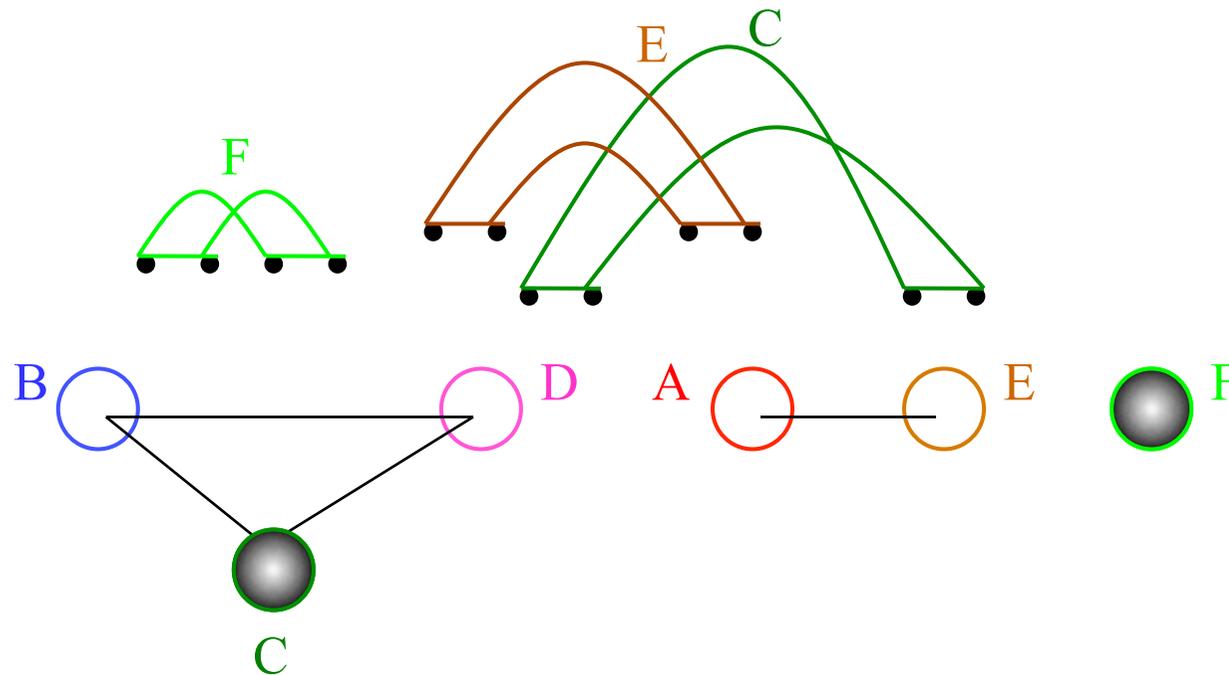
Graphes intercalés

- Un **graphe intercalé** est défini sur l'ensemble des cycles dans le graphe des points de rupture (ces cycles formeront les noeuds) qui seront connectés par des arêtes quand les cycles sont intercalés.



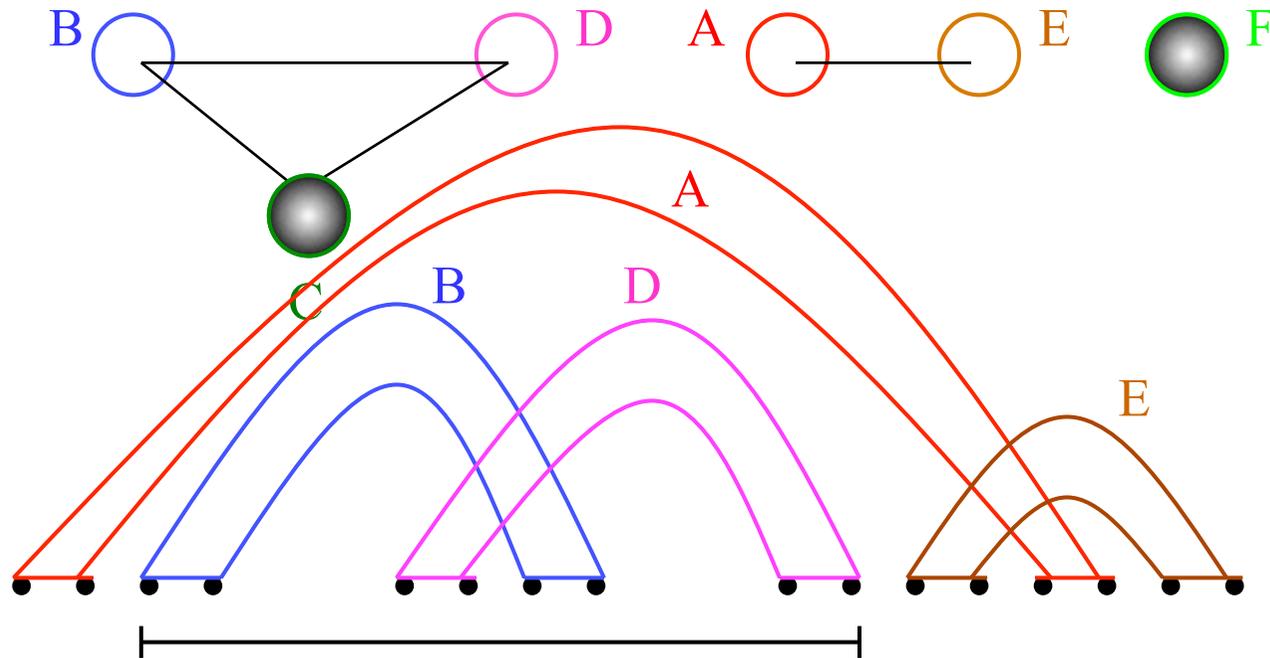
Graphes intercalés (continuation)

- Les cycles orientés sont des cycles ayant la forme suivante
- les marquer dans le graphe intercalé
- les cycles non orientés sont des cycles qui ont la forme suivante
- Dans notre exemple, A, B, D, E sont des cycles non orientés et C, F sont des cycles orientés



Obstacles

- Eliminer les composantes orientées du graphe intercalé
- Le suivant est le graphe de rupture avec ces composantes orientées éliminées
- Les obstacles sont des composantes connexes qui ne «contiennent» pas d'autre composante connexe



Distance de renversement avec obstacles

- Les obstacles constituent un problème pour la reconstruction d'un génome après réarrangement
- Ils demandent un fort nombre de renversements pour que une permutation soit transformée dans la permutation identité.
- Soit $h(\pi)$ le nombre d'obstacles dans la permutation π
- Si on considère les obstacles, la formule suivante donne une borne plus précise à la distance de renversement :

$$\longrightarrow d(\pi) \geq n+1 - c(\pi) + h(\pi)$$

Cette borne est étroite, et $h(\pi)$ est petit pour des permutations aléatoires.

Basé sur cette borne et sur des considérations appropriées sur les obstacles, un algorithme en $O(n^4)$ a pu être conçu par Hannenhalli et Pevzner (1999).

Kaplan, Shamir et Tarjan proposent un algorithme quadratique pour résoudre ce problème.

« Faster and simpler algorithm for sorting signed permutations by reversals »

Kaplan, Shamir, Tarjan

Symposium on Discrete Algorithms

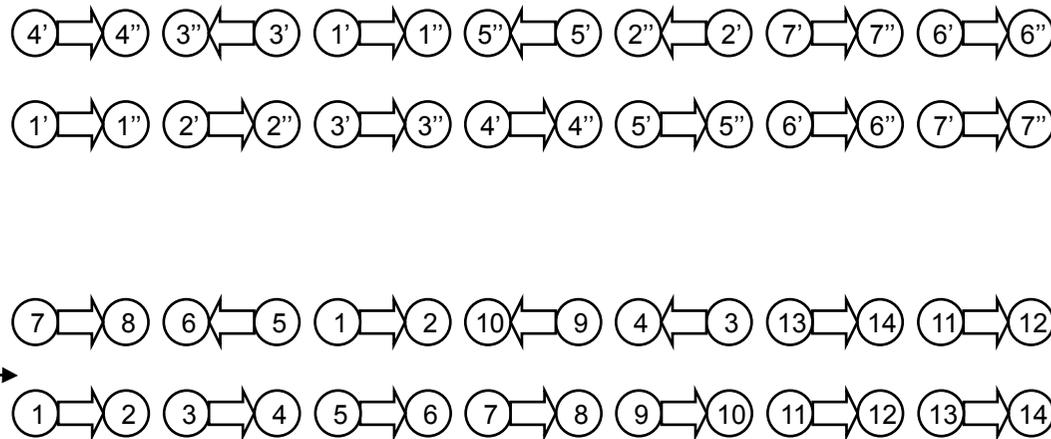
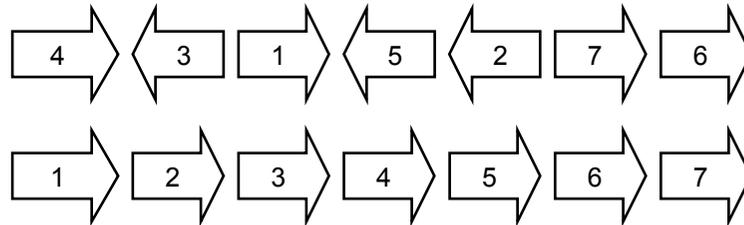
Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms

New Orleans, Louisiana, United States, 344 – 35, 1997

Passage du problème signé à un problème non signé en doublant les sommets

$$\pi = 4 \ -3 \ 1 \ -5 \ -2 \ 7 \ 6$$

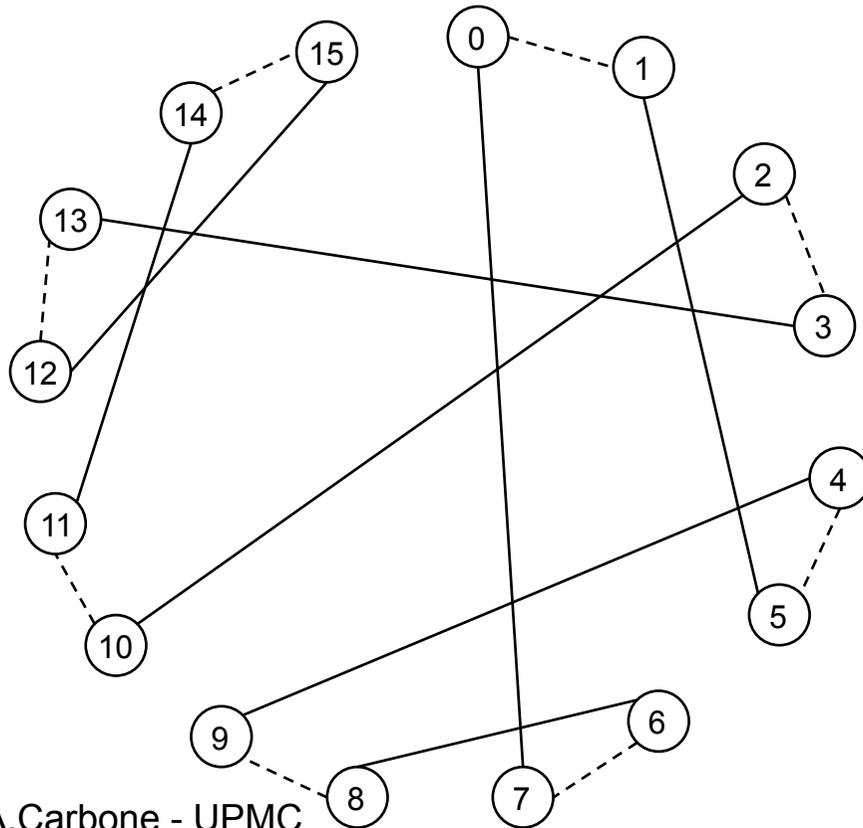
$$\text{id} = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7$$



$$\begin{aligned} x' &= 2x - 1 \\ x'' &= 2x \end{aligned}$$

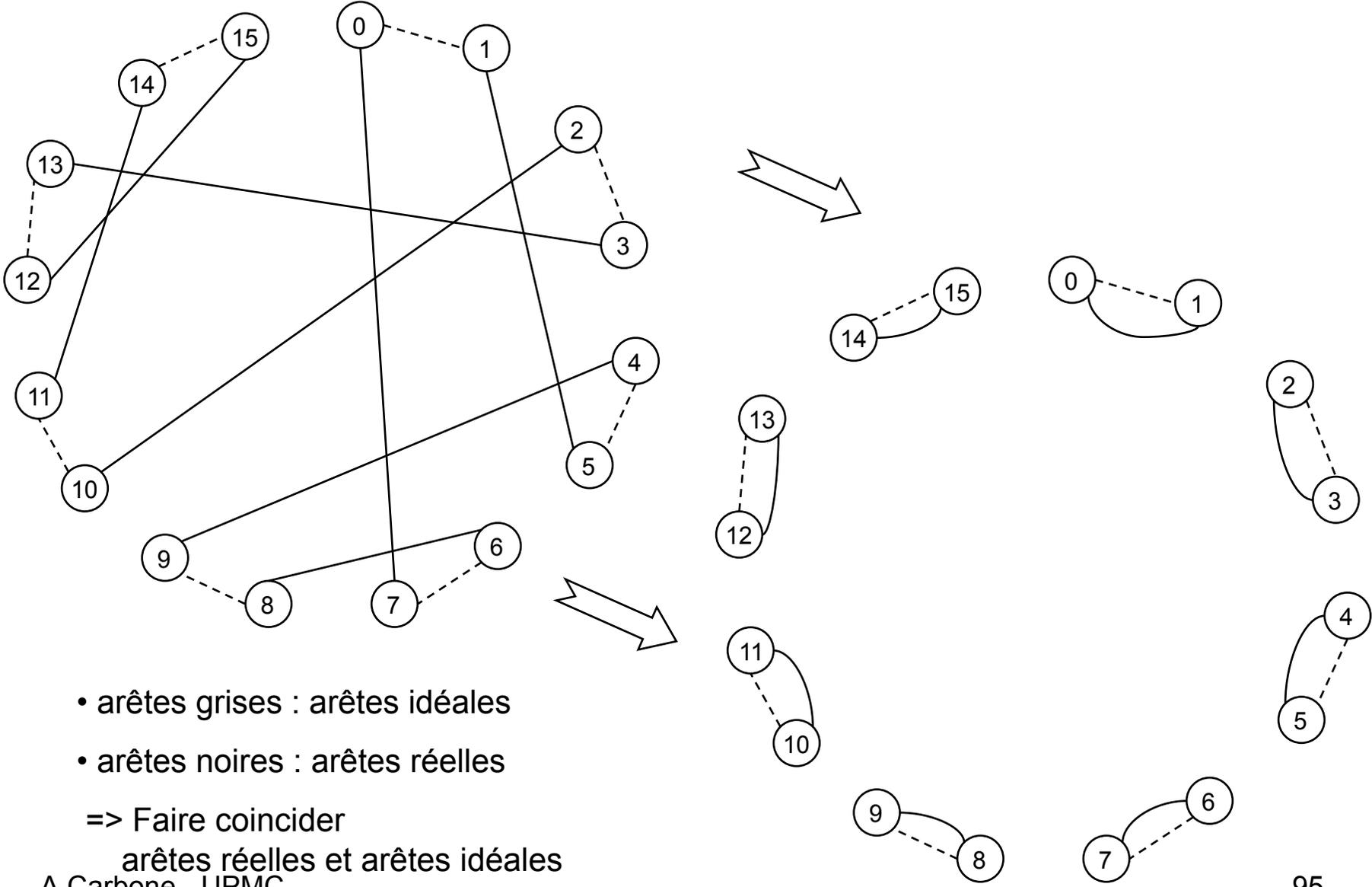
Problème équivalent non signé => graphe des points de rupture

$\pi(i)$ = 0 | 7 8 | 6 5 | 1 2 | 10 9 | 4 3 | 13 14 | 11 12 | 15
i = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



Construction du graphe

$$\pi(i) = 0 | 7 \ 8 | 6 \ 5 | 1 \ 2 | 10 \ 9 | 4 \ 3 | 13 \ 14 | 11 \ 12 | 15$$



- arêtes grises : arêtes idéales
- arêtes noires : arêtes réelles

=> Faire coïncider
arêtes réelles et arêtes idéales

Schéma de l'algorithme de Kaplan, Shamir, Tarjan

Trouver les composantes connexes du graphes de chevauchement

- Définir l'orientation des arêtes grises
- Définir les intervalles des arêtes grises
- Construire le graphe de chevauchement et en trouver les composantes connexes

Supprimer les obstacles

- Trouver les obstacles
- Fusion de 2 obstacles donne 1 composante orientée

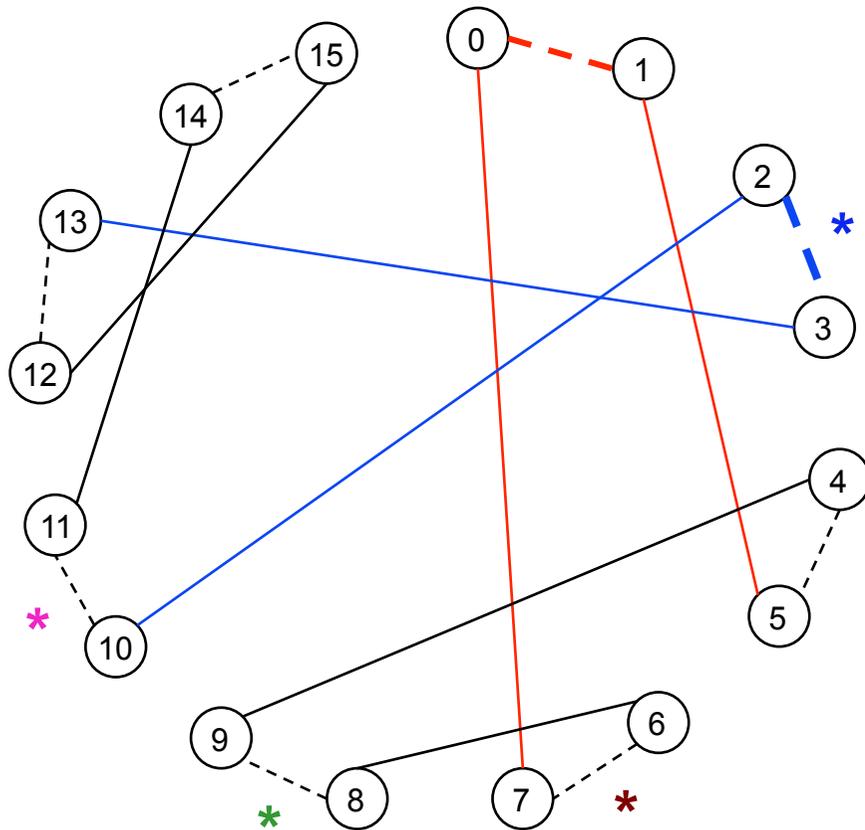
Supprimer les points de ruptures itérativement dans les composantes orientées

A chaque itération :

- Trouver une clique joyeuse
- Trouver un renversement sûr dans la clique joyeuse

Orienter les arêtes grises

$$\pi(i) = 0 \mid \overbrace{7 \ 8}^{\text{red}} \mid \overbrace{6 \ 5}^{\text{green}} \mid 1 \ 2 \mid \overbrace{10 \ 9}^{\text{blue}} \mid \overbrace{4 \ 3}^{\text{blue}} \mid 13 \ 14 \mid 11 \ 12 \mid 15$$



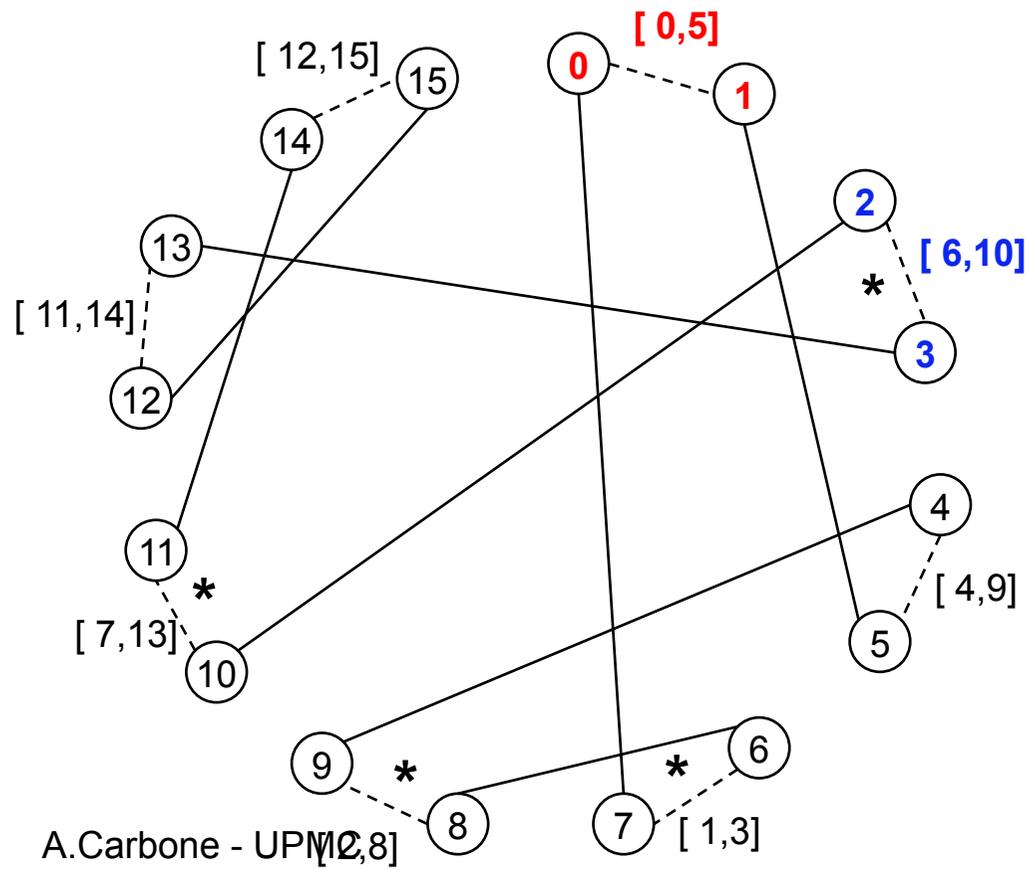
Chaque arête grise induit un renversement.

- Arête orientée (*):
induit un renversement
qui détruit 1 ou 2 points de rupture
- Arête non-orientée :
induit un renversement
qui ne détruit aucun point de rupture

=> Les arêtes orientées nous intéressent
car elles détruisent des points de
rupture

Intervalles des arêtes grises

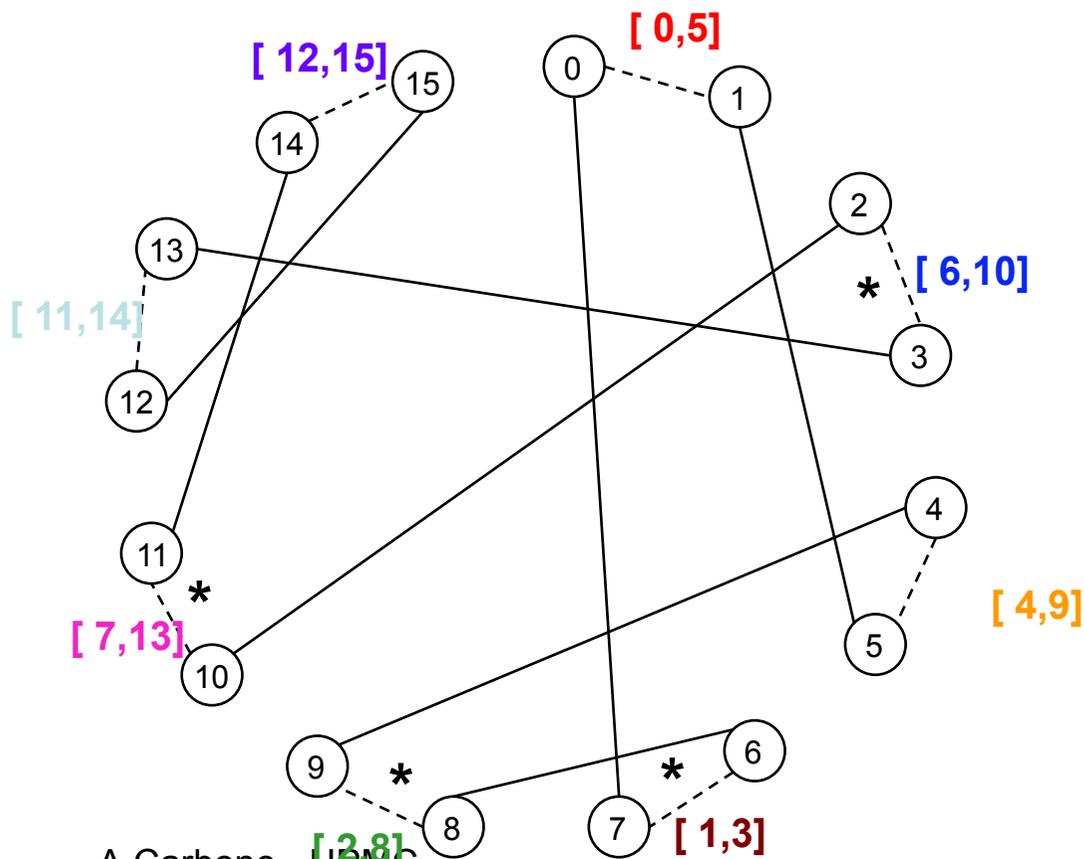
$\pi(i)$	=	0		7		8		6		5		1		2		10		9		4		3		13		14		11		12		15
i	=	0		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15



Chaque arête grise (i,j)
définit un intervalle

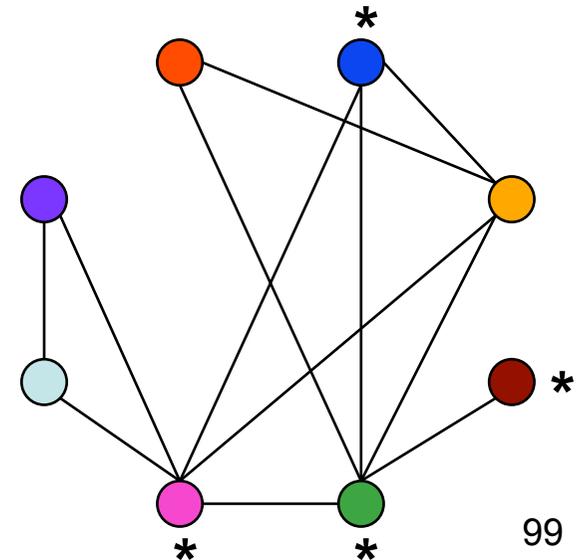
Graphe de chevauchement

$\pi(i)$	=	0		7	8		6	5		1	2		10	9		4	3		13	14		11	12		15							
i	=	0		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15



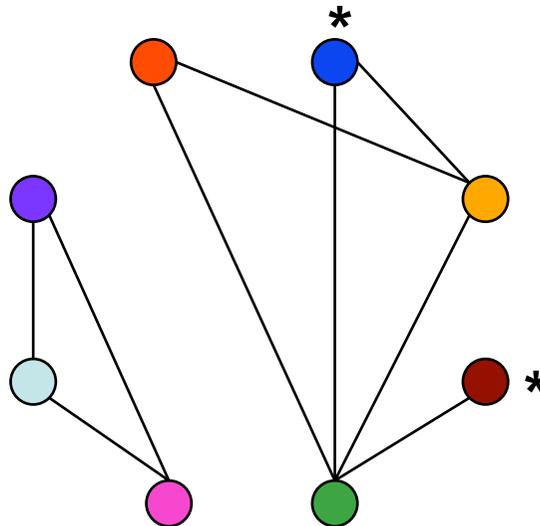
Sommets = arêtes grises

Arêtes = quand 2 arêtes grises ont leurs intervalles qui se chevauchent sans inclusion



Composantes du graphe de chevauchement

- Une composante connexe est une **composante orientée** si elle possède un sommet orienté
- Une composante connexe est une **composante non-orientée** si elle ne possède aucun sommet orienté



Composante non-orientée

A.Carbone - UPMC

Composante orientée

- Dans l'algorithme de Kaplan, Shamir, Tarjan

Composantes orientées :

- on peut trouver une suite de renversements optimale qui supprime tous les points de rupture dans toute composante orientée

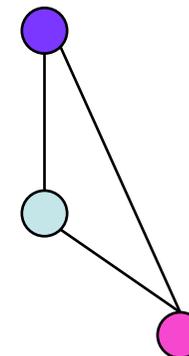
Composantes non-orientées :

- Il faut les transformer en composantes orientées

- Partie 1 de l'algo:
trouver les composantes du graphe de chevauchement
- Partie 2 :
Supprimer les composantes non-orientées

=> Notion d'obstacle : composante difficile à trier

Suppression des composantes non-orientées



- **Idée : fusionner 2 composantes non-orientées en une composante orientée**

10 | 13 14 | 11 12 | 15

Composante non-orientée : aucun point de rupture ne peut-être supprimé car tous les renversements « **admissibles** » sont de la forme $| \{ \} |$ ou $\{ | | \}$

Ex : 10 | 13 14 | 11 est de la forme 10 | | 11

Or on constate que tout **renversement** au sein d'une telle composante va faire passer un renversement de la forme $| \{ \} |$ ou $\{ | | \}$ en $\{ | \} |$ ou $| \{ | \}$

Ex : 10 | **13 14** | 11 12 | 15 \Rightarrow 10 | **14 13** | 11 12 | 15

\Rightarrow 14 | | 15 est devenu | 14 | 15

\Rightarrow il existe donc un renversement qui détruit un point de rupture donc la composante est devenue orientée

N'importe quel renversement «admissible» d'une composante non-orientée la transforme en composante orientée

Fusion de 2 composantes non-orientées

- Faire un renversement qui chevauche 2 composantes non-orientées

Ex : 2 | 6 7 | 3 4 | 8 x x x x 10 | 13 14 | 11 12 | 15

⇒ 2 | 6 7 | 14 13 | 10 x x x x 8 | 4 3 | 11 12 | 15



renversement qui détruit un point de rupture
 => composante devient orientée

- Les 2 composantes ont fusionné en une composante

⇒ 2 | 6 7 | 14 13 | 10 x x x x 8 | 4 3 | 11 12 | 15

Fusion des 2 composantes non-orientées en 1 seule orientée

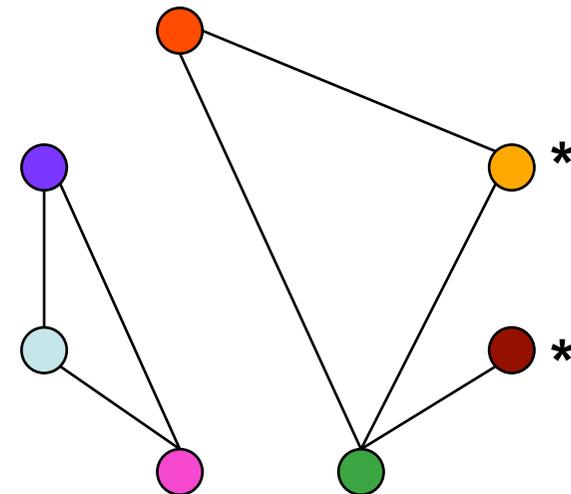
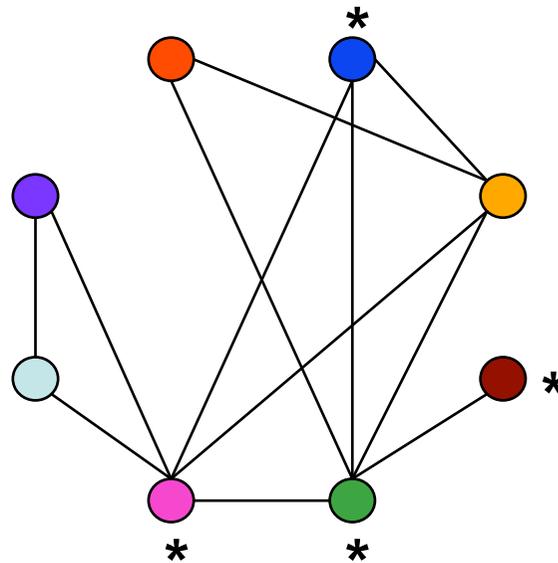
Composante orientée et clique joyeuse

- Partie 3 : toutes les composantes non-orientées ont été supprimées
Il ne reste que des composantes orientées

⇒ Traitement des composantes orientées

⇒ Idée : dans toute composante orientée, il y a des renversements qui détruisent des ruptures.
Il faut en choisir une qui n'introduise pas de nouvelles composantes non-orientées

Choix d'un renversement = choix d'un sommet



Renversement du sommet
bleu

Action d'un renversement d'un sommet :

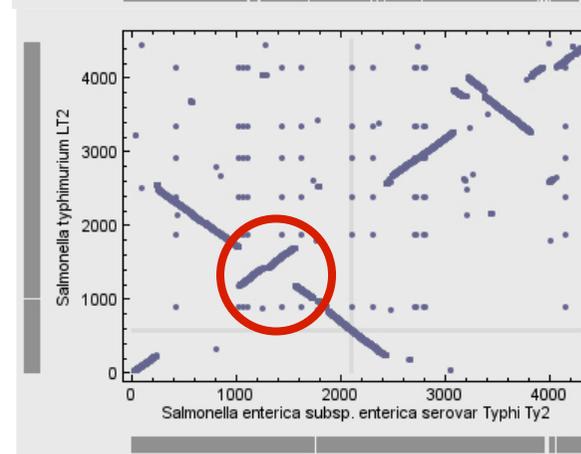
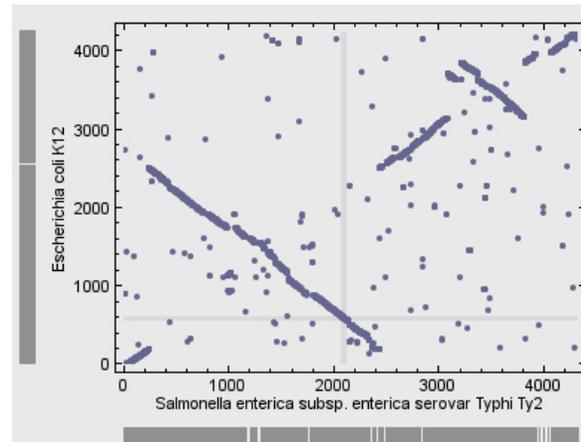
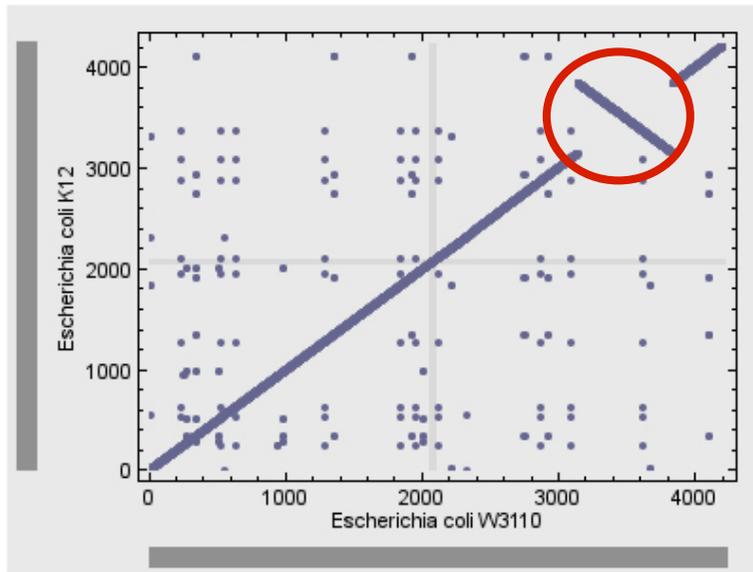
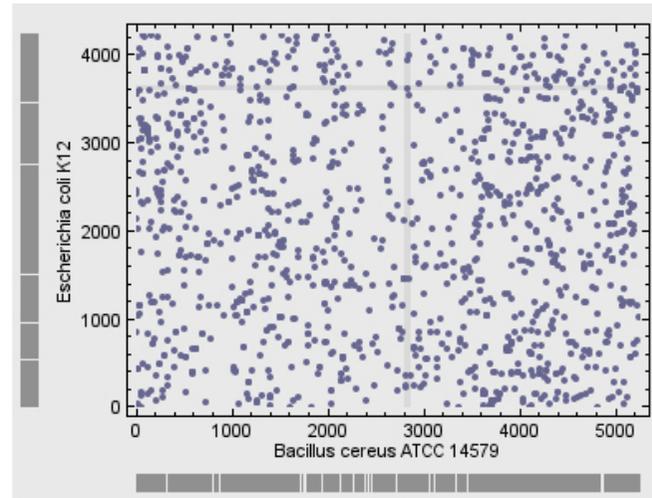
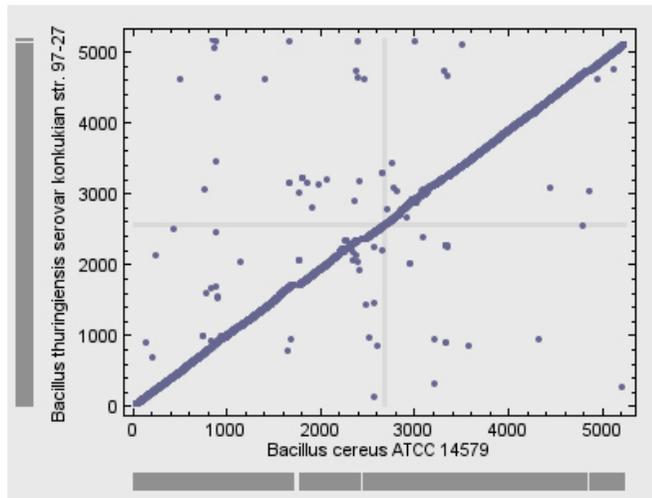
⇒ N'agit que sur la clique des voisins

A. Carbone - UPMC

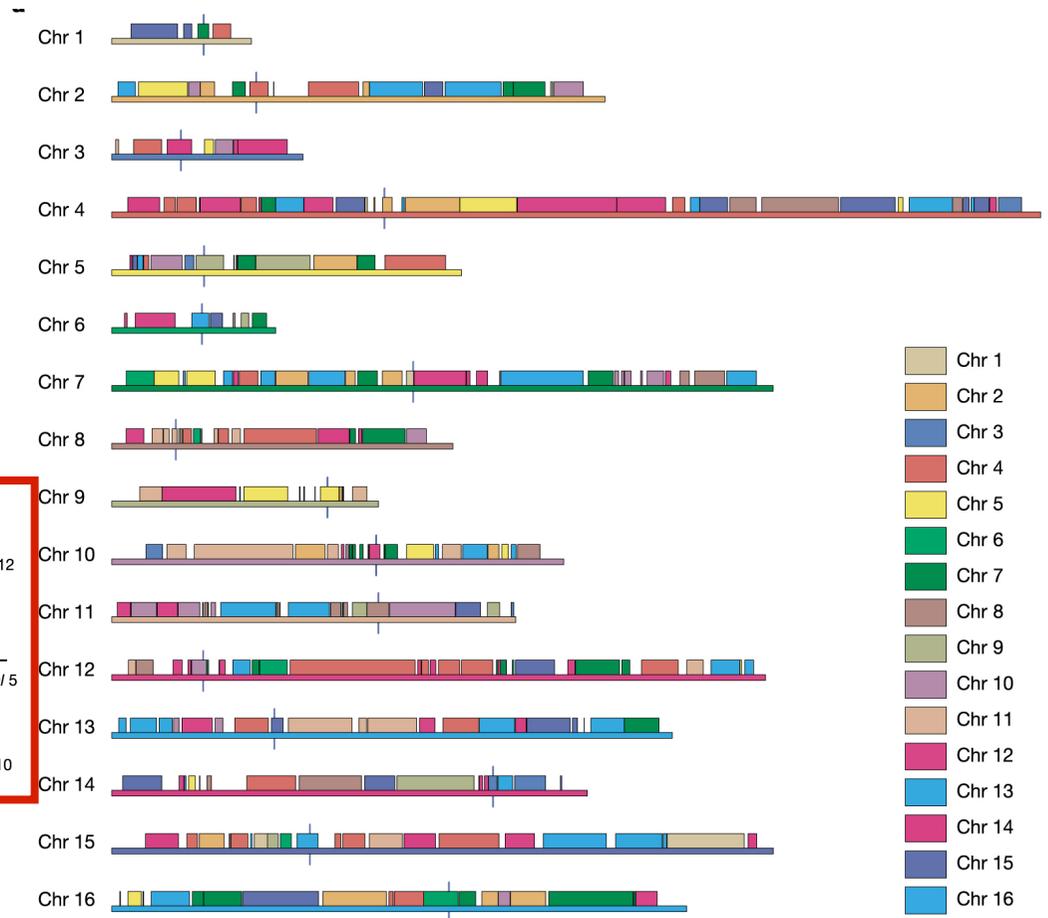
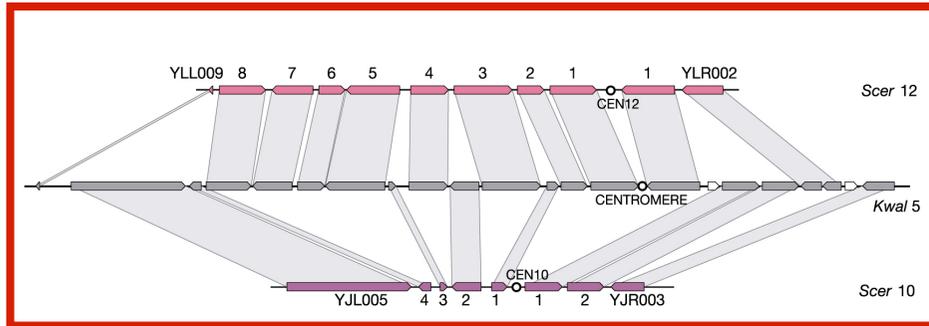
Ex : renversement du sommet bleu

Justifications biologiques:

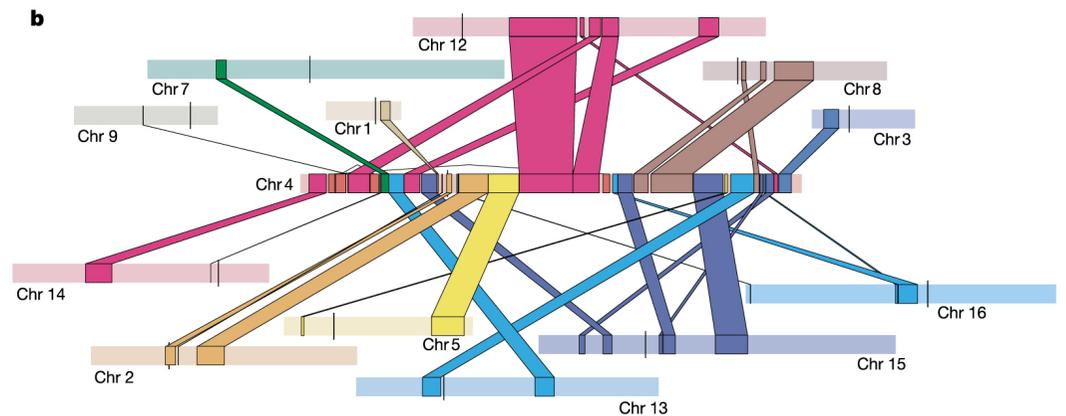
- **rearrangement** des génomes procaryotes et eucaryotes
 - **duplication massive** chez les eucaryotes
- des génomes dans l'évolution.

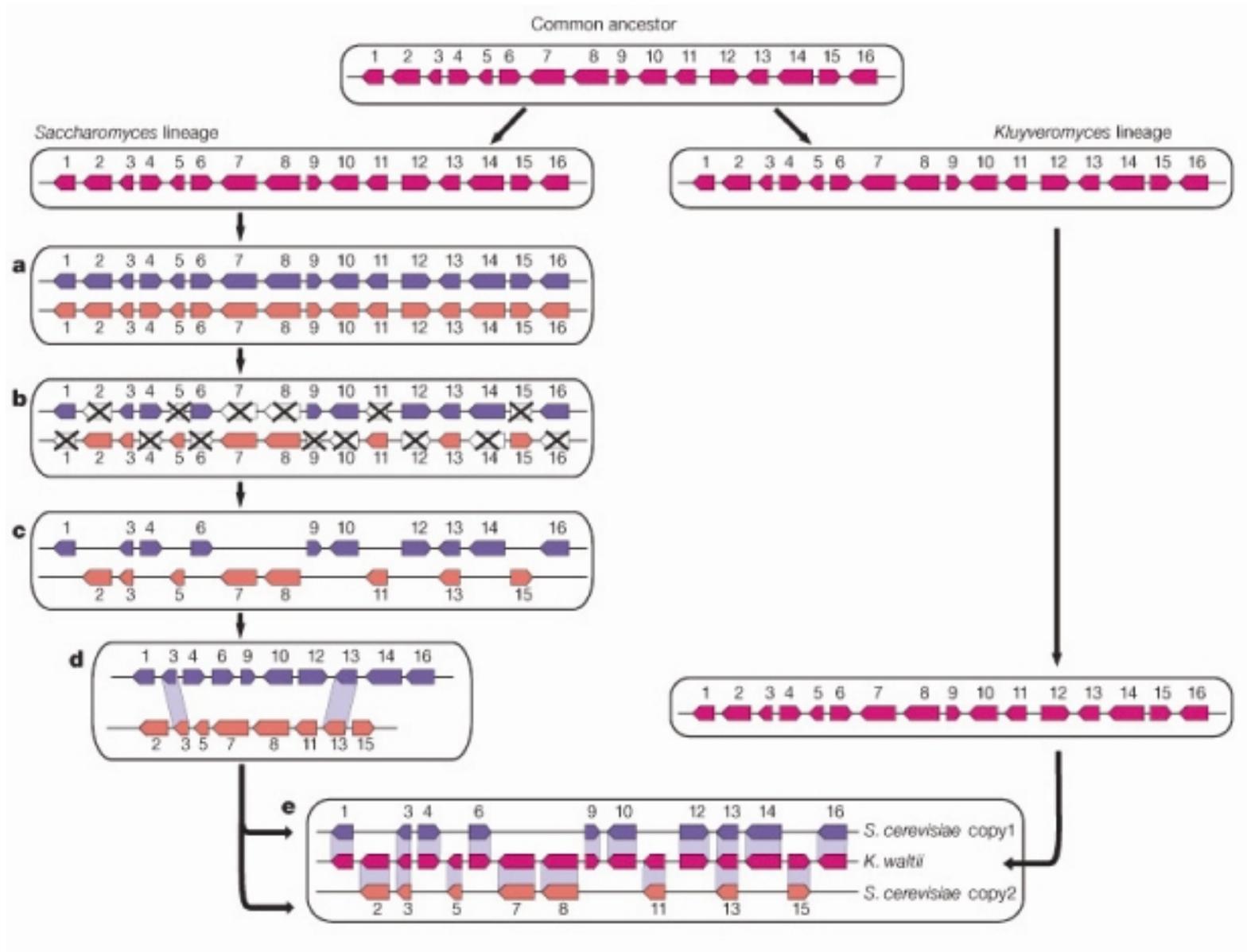


Duplication du génome de la levure

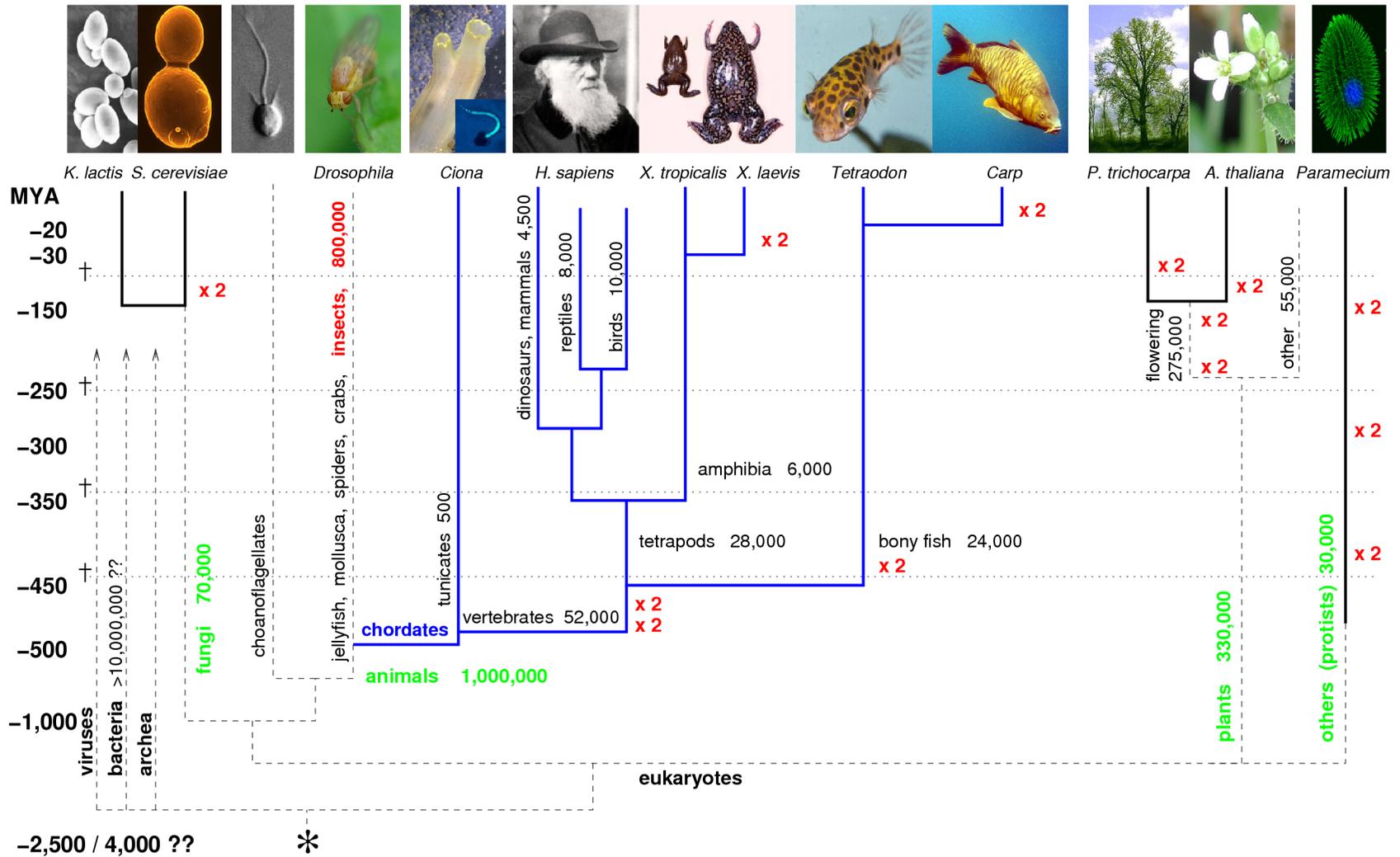


... et réarrangement





Whole Genome Duplications in Evolution



Références bibliographiques

Pevzner, *Computational Molecular Biology, an algorithmic approach*, MIT Press 2000.

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

W.H.Gates, C.H.Papadimitriou, Bound for sorting by prefix reversals, *Discrete Mathematics*, 27, 47-57, 1979.

A.Bergeron, A very elementary presentation of the Hannenhalli-Pevzner theory.
In *Proc. 12th Annual Symposium on Combinatorial Pattern Matching (CPM'01)*, 106-117, 2001.

H.Kaplan, R.Shamir, R.E.Tarjan, Faster and simpler algorithm for sorting signed permutations by reversals, in *SIAM J. of Computing*, 29(3), 880-892, 1999.

Kecegiouglu, Sankoff, Exact and approximation algorithms for sorting by reversals, with application to genome arrangement, *Algorithmica*, 13, 180-210, 1995.

Bader, D.A., Moret, B.M.E., Yan, M. (2001) A linear-time algorithm for computing inversion distances between signed permutations with an experimental study, *J. Comput. Biol.* 8(5), 483-491.

Caprara, A. (1999) Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM J. Discrete Math.* **12**(1), 91-110 (electronic).

Hannenhalli, S., Pevzner, P.A. (1995) Transforming men into mice (polynomial algorithm for genomic distance problem). In *36th Annual Symposium on Foundations of Computer Science (Milwaukee, WI, 1995)*, IEEE Comput. Soc. Press, Los Alamitos, CA, pp. 581-592.

Hannenhalli, S., Pevzner, P. (1996) To cut ... or not to cut (applications of comparative physical maps in molecular evolution). In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (Atlanta, GA, 1996)*, ACM, New York, pp. 304-313.

Hannenhalli, S., Pevzner, P.A. (1999) Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM* **46**(1), 1-27.

Remarque sur les permutations avec signe

- Les gènes sont des fragments **dirigés** d'ADN et on représente un génome par une permutation avec signe
- Si les gènes sont dans la même position mais les orientations sont différentes, les **génomés** ne sont pas **équivalents**.
- Par exemple, ces deux permutations ont le même ordre, mais certains gènes ont une orientation renversée; donc, il ne s'agit pas de deux séquences de gènes équivalents.

