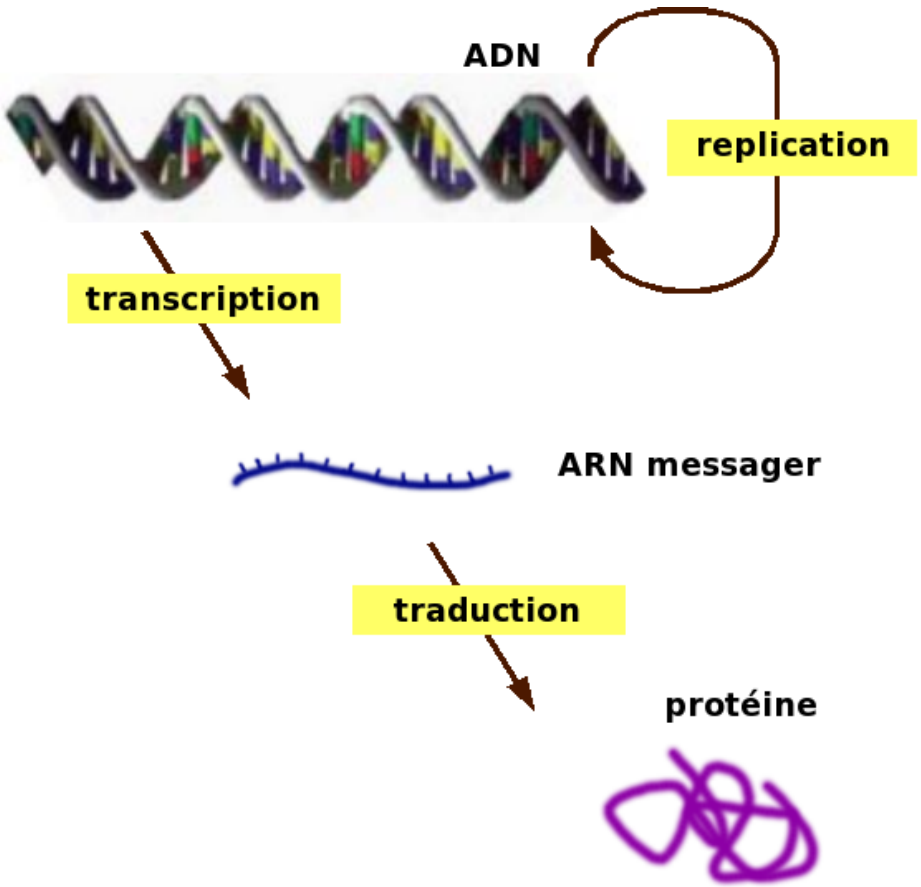
Architecture et Génomique
Comparaison rapide de séquences

Mathieu Giraud, Dominique Lavenier
mgiraud@irisa.fr

Projet Symbiose, IRISA / Université de Rennes 1

École chercheurs IRISA
Liffré, le 3 novembre 2005

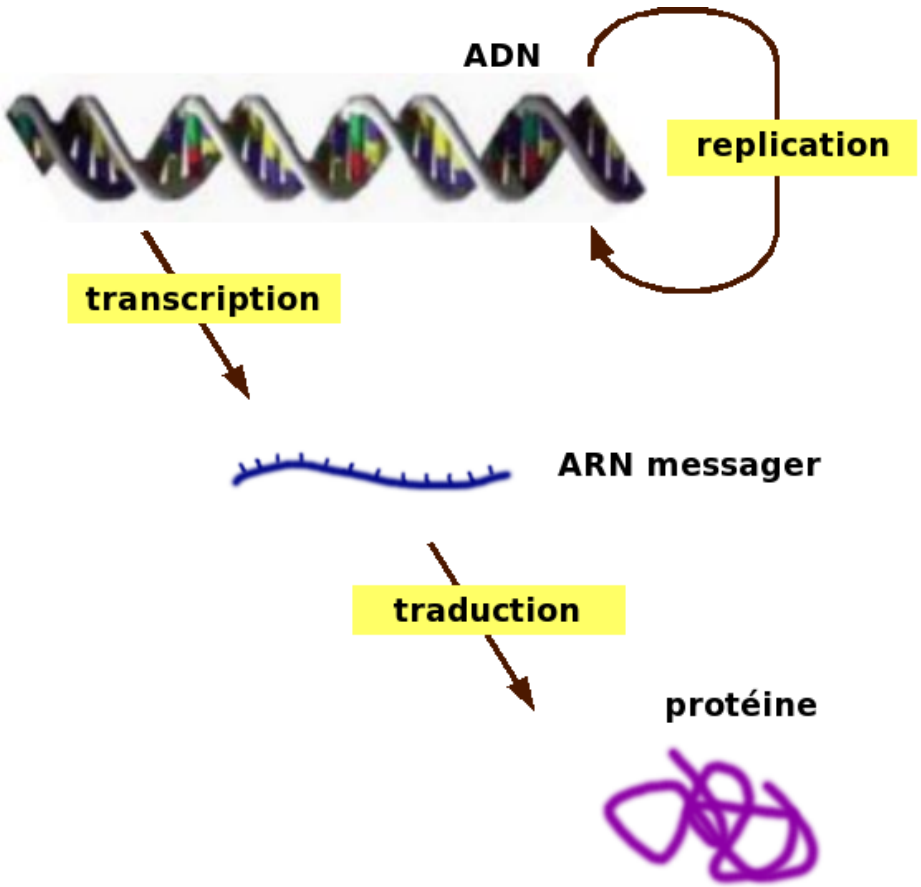
Similarités dans les chaînes d'ADN



- ADN : $\Sigma_4 = \{A, C, G, T\}$
- Bases de données : EMBL

$107 \cdot 10^9$ bases
dans $58 \cdot 10^6$ séquences
(septembre 2005)

Similarités dans les chaînes d'ADN

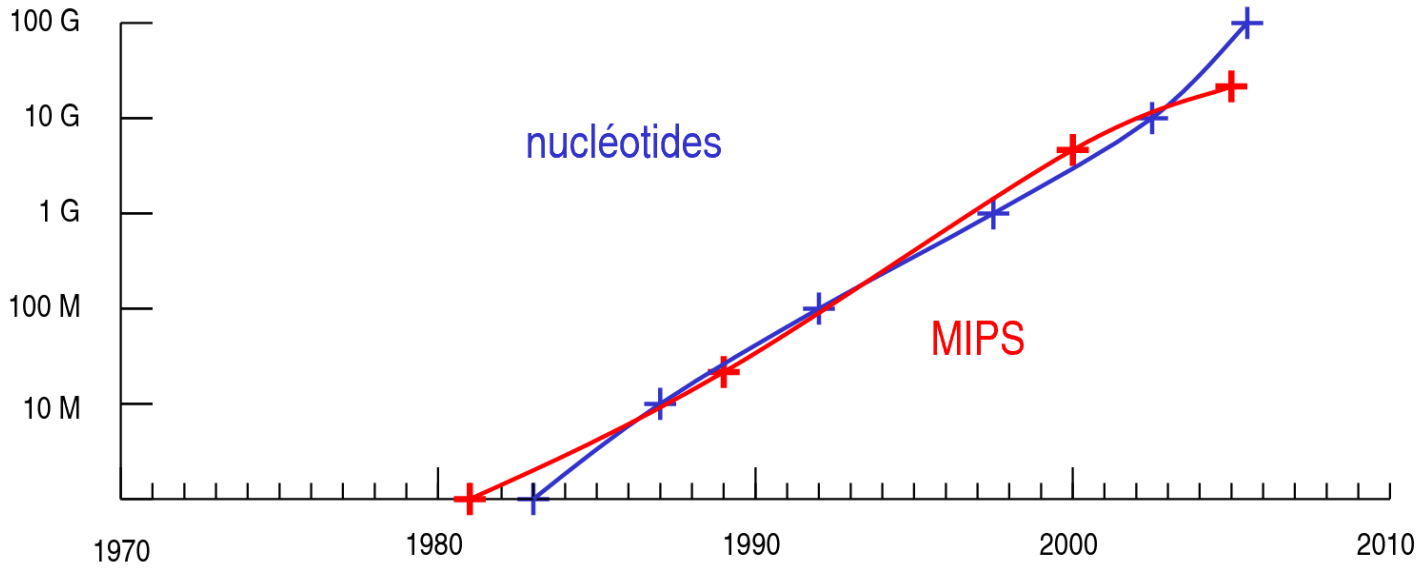


- ADN : $\Sigma_4 = \{A, C, G, T\}$
- Bases de données : EMBL

$107 \cdot 10^9$ bases
dans $58 \cdot 10^6$ séquences
(septembre 2005)

- Séquences similaires (> 30 %)
  $\longrightarrow$ structures 3D similaires
  $\longrightarrow$ fonctions pouvant être similaires
  $\longrightarrow$ recherche de similarités

Taille des banques et vitesse des processeurs

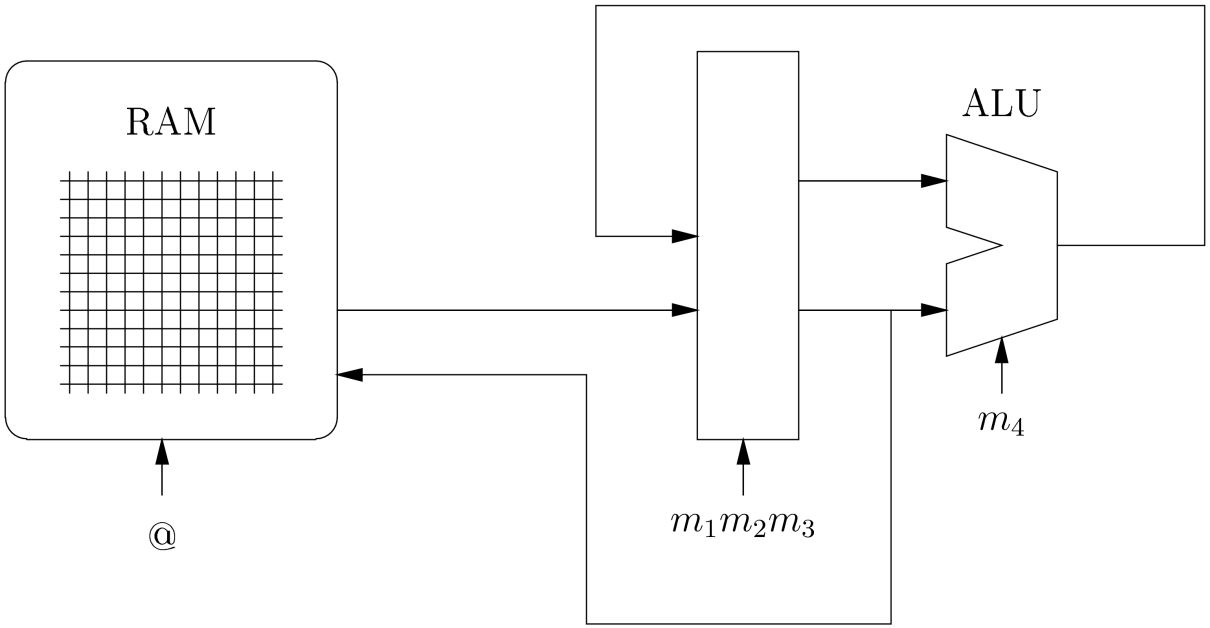


$\longrightarrow$ Il faut aller “plus vite”

Accéder à de grosses masses de données

Arbre des suffixes avec liens suffixes

$\longrightarrow$ parcours aléatoire de la mémoire

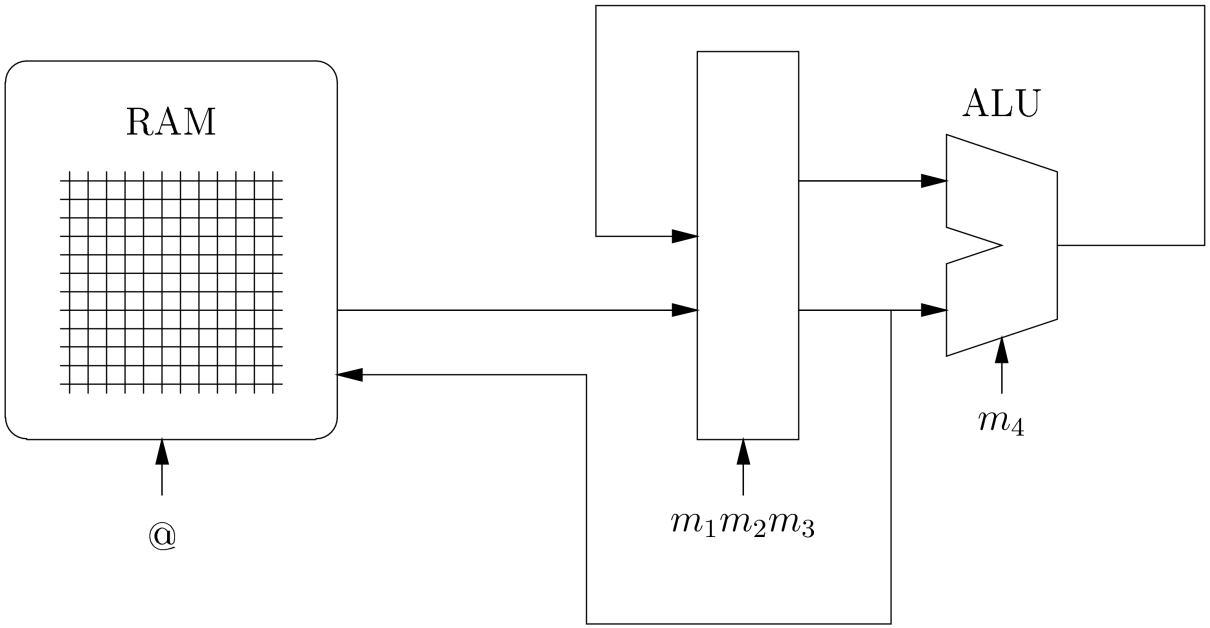


- L'accès de données en mémoire est en $\mathcal{O}(1)$.

Accéder à de grosses masses de données

Arbre des suffixes avec liens suffixes

$\longrightarrow$ parcours aléatoire de la mémoire

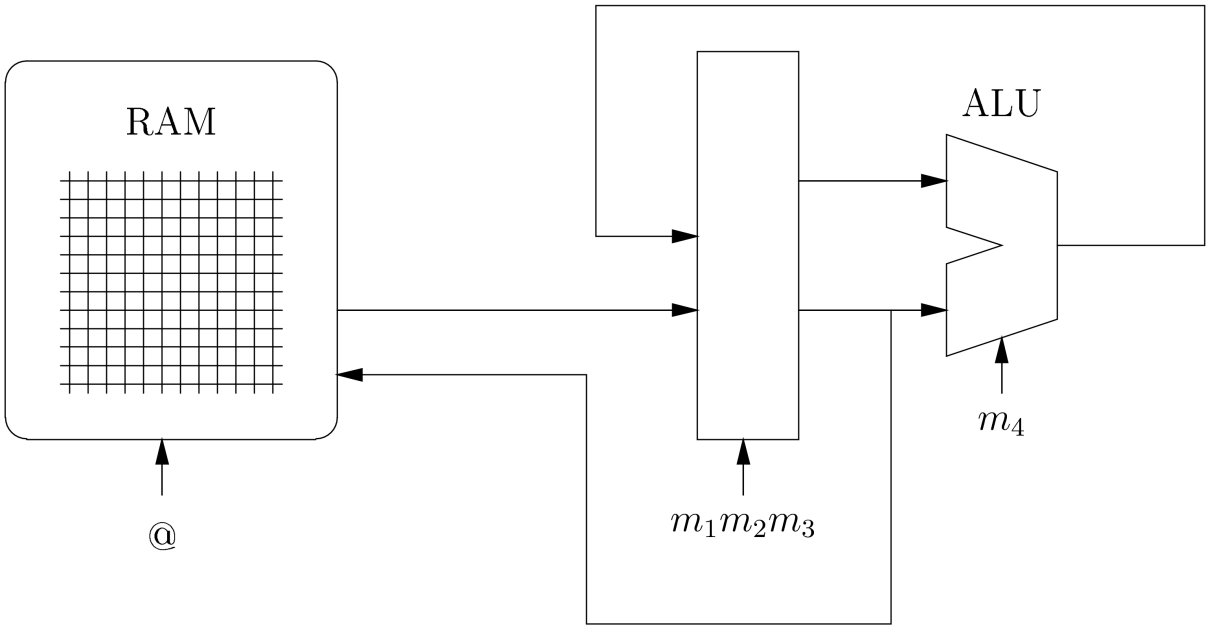


- L'accès de données en mémoire est en $\mathcal{O}(1)$. NON...
- L'accès de données dans le cache est en $\mathcal{O}(1)$.

Accéder à de grosses masses de données

Arbre des suffixes avec liens suffixes

$\longrightarrow$ parcours aléatoire de la mémoire



- L'accès de données en mémoire est en $\mathcal{O}(1)$. NON...
- L'accès de données dans le cache est en $\mathcal{O}(1)$. NON...
- L'accès de données dans le cache L1 est en $\mathcal{O}(1)$. (?)

Accéder à de grosses masses de données

$\longrightarrow$ hiérarchie mémoire progressive.

Registres	≤ 1 cycle	1 ko
Cache L1	1 – 10 cycles	128 ko
Cache L2	5 – 50 cycles	2 Mo
Mémoire externe	10 – 1000 cycles	1 Go
Stockage de masse	$\times 10^6$ cycles	200 Go

2 GHz $\longrightarrow$ un cycle = 0,5 ns

$\longrightarrow$ Il faut aller “plus vite”

Comment accélérer ?

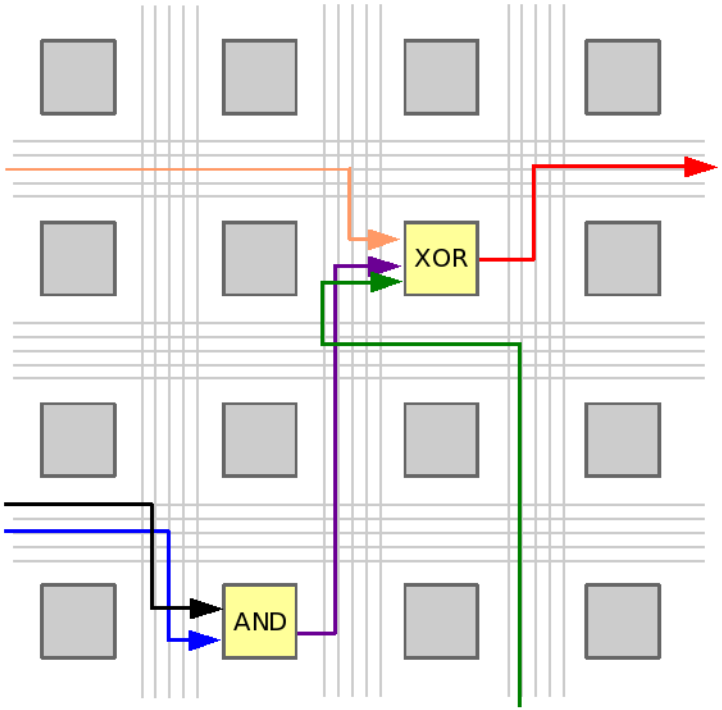
- Améliorer les algos, leurs constantes
- Paralléliser
  - Cluster de PC
  - Threads, multi-cores

Comment accélérer ?

- Améliorer les algos, leurs constantes
- Paralléliser
  - Cluster de PC
  - Threads, multi-cores
  - Architectures spécialisées : ASIC, ..., FPGA

Coût / puissance ?

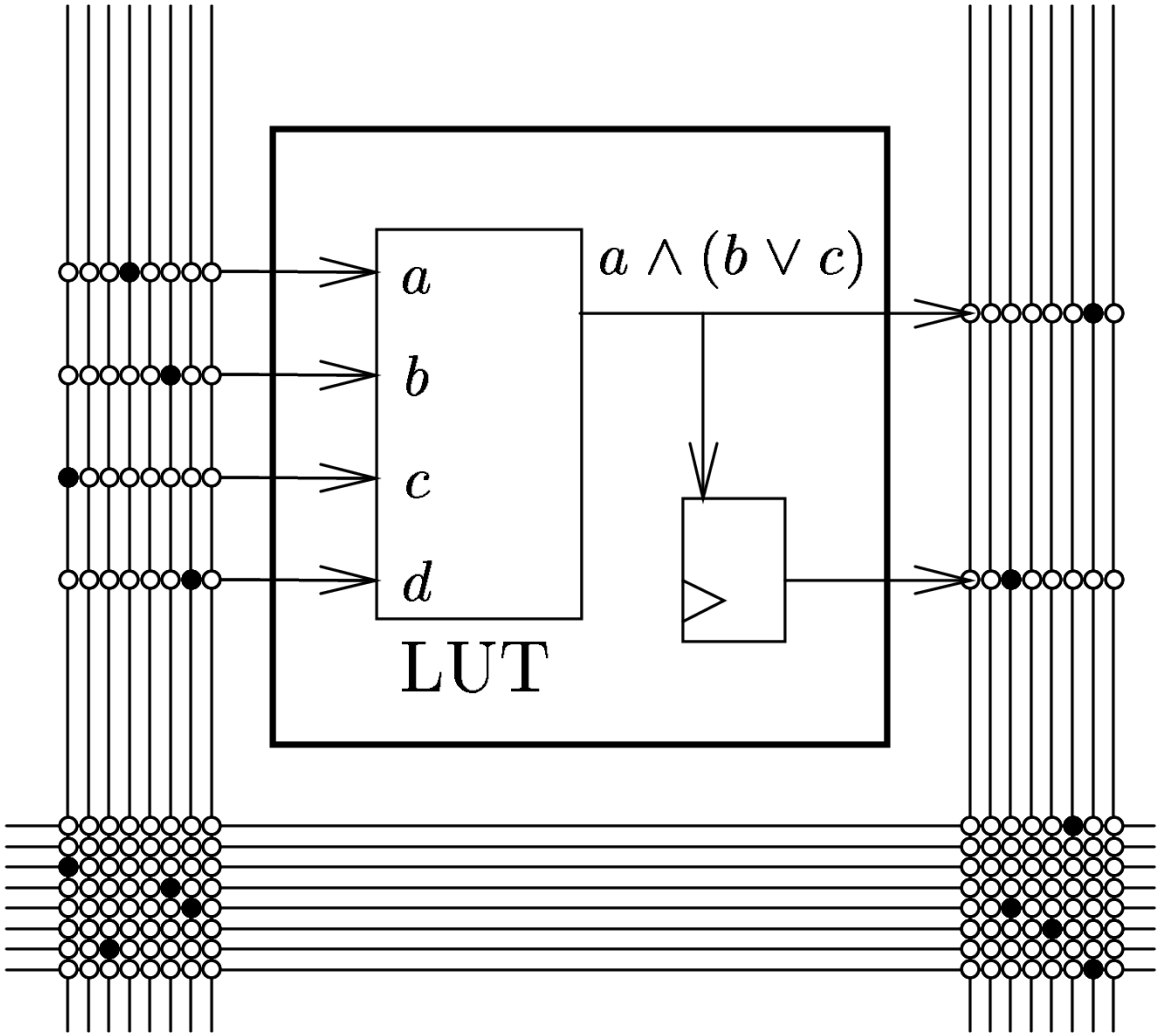
Les FPGAs, une puissance de calcul reconfigurable



- Grille de cellules logiques
- Interconnexion (routage)
- Reconfigurable
- Prototypage / circuits économiques

2000	Spartan II	$28 \times 42 = 1176$ CLB	(200 K portes)
2004	Spartan 3	$104 \times 80 = 8300$ CLB	(5 M portes)

Détail d'une LUT (look-up table) et du routage



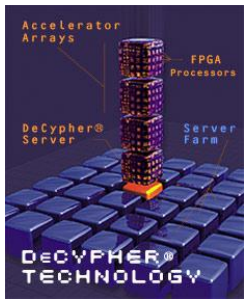
Marchands du temple

BioXL/H

Smith-Waterman, Blast, Profile, HMM...

Paracel GeneMatcher

Réseau 1D de quelques milliers de cellules



TimeLogic DeCypher

FPGAs, algorithmes usuels

Objectifs

- Partie I : Programmation dynamique
  SW, NW, architectures systoliques
- Partie II : Heuristiques à base de graines
  BLAST, graines, filtres Bloom
- Bonus : Les gènes olfactifs du chien

Partie I
Programmation dynamique

1. Alignements
2. Relations de programmation dynamique
3. Architectures systoliques
4. Automates pondérés avec ε-transitions
5. R-disk

Comparaison de séquences et évolution

1.	T	T	G	A	A	A	T	G	C	G	A	G	T
2.	T	T	C	A	T	A	T	C	G	T	A	G	T

- La recopie de l'ADN n'est pas une opération exacte
- Erreurs $(10^{-8} - 10^{-5}) \longrightarrow$ mutations et évolution
- *La chaîne 1 a muté vers la chaîne 2*

Comparaison de séquences et évolution

1.	T	T	G	A	A	A	T	G	C	G	A	G	T
2.	T	T	C	A	T	A	T	C	G	T	A	G	T

- La recopie de l'ADN n'est pas une opération exacte
- Erreurs $(10^{-8} - 10^{-5}) \longrightarrow$ mutations et évolution
- *La chaîne 1 a muté vers la chaîne 2*, ou réciproquement, ou il existe un ancêtre commun aux deux chaînes

Alignement

```
1.  T T G A A A T G C G – A G T
    | |   | |   | |   | |   | |
2.  T T C A T A T – C G T A G T
```

Alignement

```
1.  T T G A A A T G C G – A G T
    | |   | |   | |   | |   | |
2.  T T C A T A T – C G T A G T
```

À chaque position, 3 possibilités :

- un *match* : même caractère α
- un *mismatch* ou une *substitution* : deux caractères $\alpha \neq \beta$
- un *gap* : *insertion* d'un caractère dans une chaîne, ou symétriquement *délétion*.

Alignement

```
1.  T T G A A A T G C G – A G T
    | |   | |   | |   | |   | | |
2.  T T C A T A T – C G T A G T
```

À chaque position, 3 possibilités :

- un *match* : même caractère α
- un *mismatch* ou une *substitution* : deux caractères $\alpha \neq \beta$
- un *gap* : *insertion* d'un caractère dans une chaîne, ou symétriquement *délétion*.

-1 par erreur $\longrightarrow$ score de -4

Scores

N A K N N
| | | |
N A – C N

Scores

N	A	K	N	N
|	|			|
N	A	–	C	N

Matrice BLOSUM62

$\longrightarrow$ probabilités de substitution $d(\alpha, \beta)$ (Henikoff 1992).

	A	R	**N**	**D**	**C**	Q	
A	4	-1	-2	-2	0	-1	A
R		5	0	-2	-3	1	R
N			**6**	**1**	**-3**	0	**N**
D				6	-3	0	D
C					9	-3	C
Q						5	Q

Scores

N	A	K	N	N
|	|			|
N	A	–	C	N

	A	R	N	D	C	Q	
A	4	-1	-2	-2	0	-1	A
R		5	0	-2	-3	1	R
N			**6**	**1**	**-3**	0	**N**
D				6	-3	0	D
C					9	-3	C
Q						5	Q

Matrice BLOSUM62

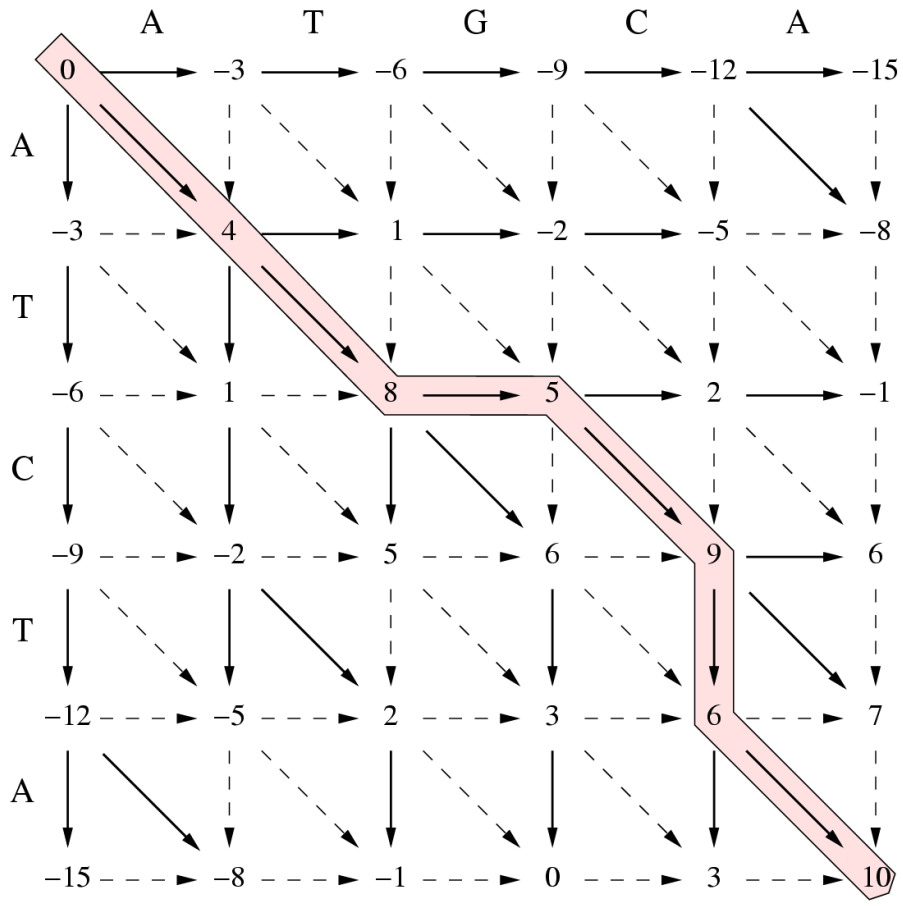
$\longrightarrow$ probabilités de substitution $d(\alpha, \beta)$ (Henikoff 1992).

gap $\longrightarrow$ pénalité $g = -5$ $\quad d(\alpha, -) = g$ or $d(-, \beta) = g$

Score de $(+6) + (+4) + (-5) + (-3) + (+6) = (+8)$

Aligner ATGCA et ATCTA

$$\begin{aligned} \text{match} &\longrightarrow +4 \\ \text{substitution} &\longrightarrow -2 \\ \text{gap} &\longrightarrow -3 \end{aligned}$$


A T G C A
A
T
C
T
A
0 −3 −6 −9 −12 −15
−3 4 1 −2 −5 −8
−6 1 8 5 2 −1
−9 −2 5 6 9 6
−12 −5 2 3 6 7
−15 −8 −1 0 3 10

A T G C – A
| | | |
A T – C T A

Alignement global : Needleman-Wunsch

- $X = (x_1, x_2 \dots x_m)$ et $Y = (y_1, y_2 \dots y_n)$
- $H(i,j)$ similarité entre $x_1 \dots x_i$ et $y_1 \dots y_j$

$$\forall i : \quad H(i,0) = g \times i \qquad \forall j : \quad H(0,j) = g \times j \qquad \forall i,j, ij \neq 0 :$$

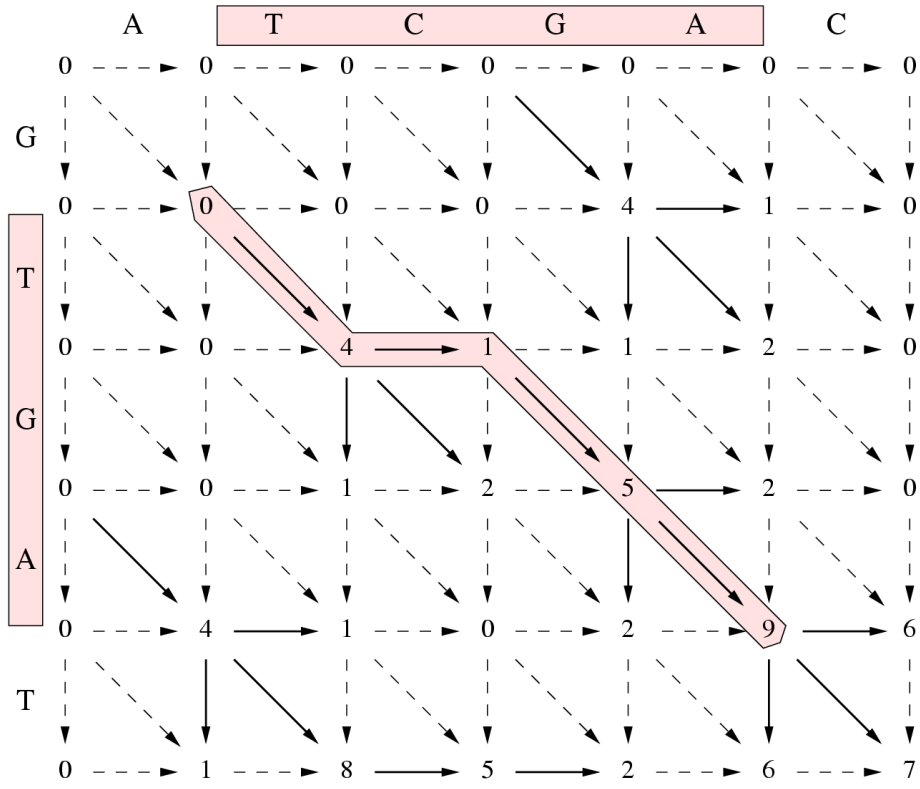
$$H(i,j) = \max \begin{cases} H(i-1,j-1) + d(x_i, y_j) & \text{(match ou substitution)} \\ H(i-1,j) - g & \text{(insertion)} \\ H(i,j-1) - g & \text{(délétion)} \end{cases}$$

$\longrightarrow$ $H(n,m)$ similarité globale entre X et Y

Alignement local : Smith-Waterman

Aligner localement ATGCAC et GTCTAT

$$\begin{aligned} \text{match} &\longrightarrow +4 \\ \text{substitution} &\longrightarrow -2 \\ \text{gap} &\longrightarrow -3 \end{aligned}$$


A T C G A C
G T G A T
0 0 0 0 0 0 0
0 0 0 0 4 1 0
0 0 4 1 1 2 0
0 0 1 2 5 2 0
0 4 1 0 2 9 6
0 1 8 5 2 6 7

T C G A

| | |

T – G A

Alignement local : Smith-Waterman

- $H(i,j)$ score maximum entre toutes les sous-séquences $x_a \dots x_i$ et $y_b \dots y_j$

$$\forall i,j : \quad H(i,0) = H(0,j) = 0 \qquad \forall i,j, ij \neq 0 :$$

$$H(i,j) = \max \begin{cases} 0 & \text{(début d'un nouvel alig.)} \\ H(i-1,j-1) + d(x_i, y_j) & \text{(match ou substitution)} \\ H(i-1,j) - g & \text{(insertion)} \\ H(i,j-1) - g & \text{(délétion)} \end{cases}$$

$\longrightarrow$ $\max_{i,j} H(i,j)$ meilleure similarité entre X et Y

Alignement local : Smith-Waterman

- $H(i,j)$ score maximum entre toutes les sous-séquences $x_a \dots x_i$ et $y_b \dots y_j$

$$\forall i,j : \quad H(i,0) = H(0,j) = 0 \qquad \forall i,j, ij \neq 0 :$$

$$H(i,j) = \max \begin{cases} 0 & \text{(début d'un nouvel alig.)} \\ H(i-1,j-1) + d(x_i, y_j) & \text{(match ou substitution)} \\ H(i-1,j) - g & \text{(insertion)} \\ H(i,j-1) - g & \text{(délétion)} \end{cases}$$

$\longrightarrow$ $\max_{i,j} H(i,j)$ meilleure similarité entre X et Y

Pénalité de gap *linéaire* : $g(\ell) = g \cdot \ell$

Alignement local : Smith-Waterman-Gotoh

Pénalité de gap *affine* : $g(\ell) = g_o + \ell\, g_e$

Alignement local : Smith-Waterman-Gotoh

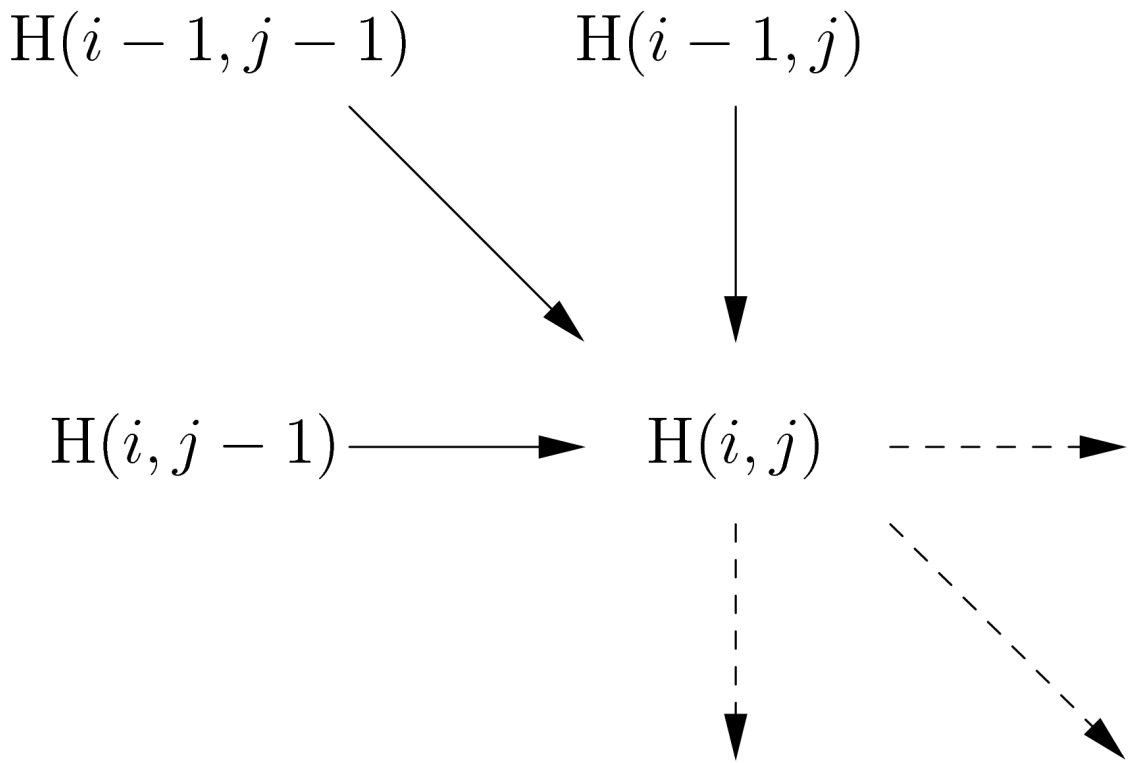
Pénalité de gap *affine* : $g(\ell) = g_o + \ell\, g_e$

$\forall i, j, ij \neq 0$:

$$H(i,j) = \max \begin{cases} 0 & \text{(si alignement local SW)} \\ H(i-1,j-1) + d(x_i, y_j) & \text{(match ou substitution)} \\ I(i,j) & \text{(insertion)} \\ D(i,j) & \text{(délétion)} \end{cases}$$

$$I(i,j) = \max \begin{cases} H(i-1,j) - g_o \\ I(i-1,j) - g_e \end{cases} \qquad D(i,j) = \max \begin{cases} H(i,j-1) - g_o \\ D(i,j-1) - g_e \end{cases}$$

Localité du calcul

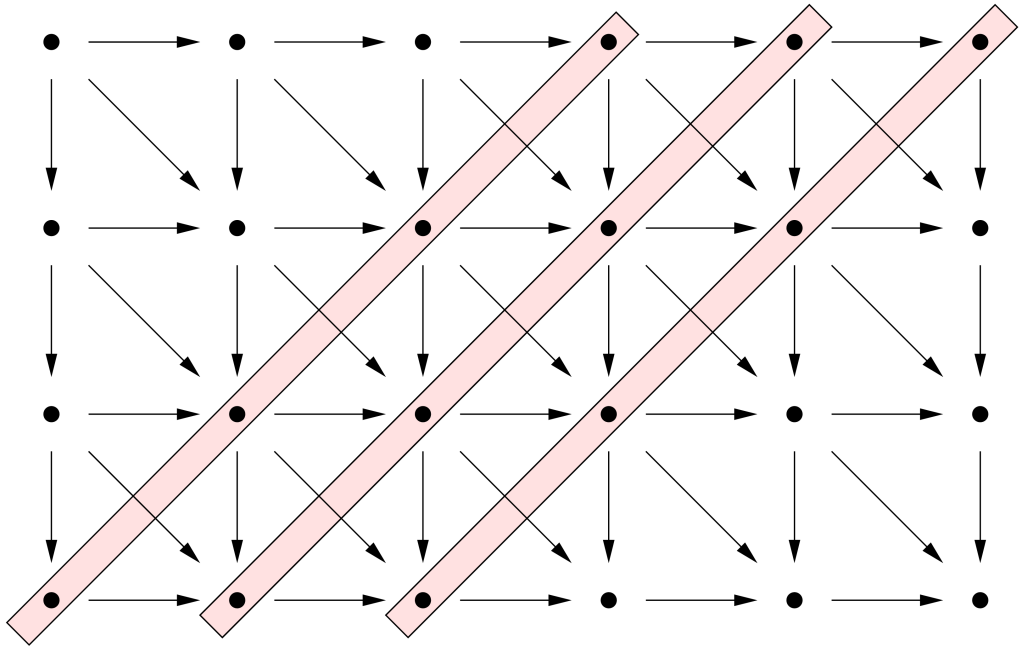


- Dépendance de trois cellules précédentes
- Seule information extérieure : $d(x_i, y_j)$
- Transmission des données vers les trois cellules suivantes

En logiciel

Calcul exhaustif :

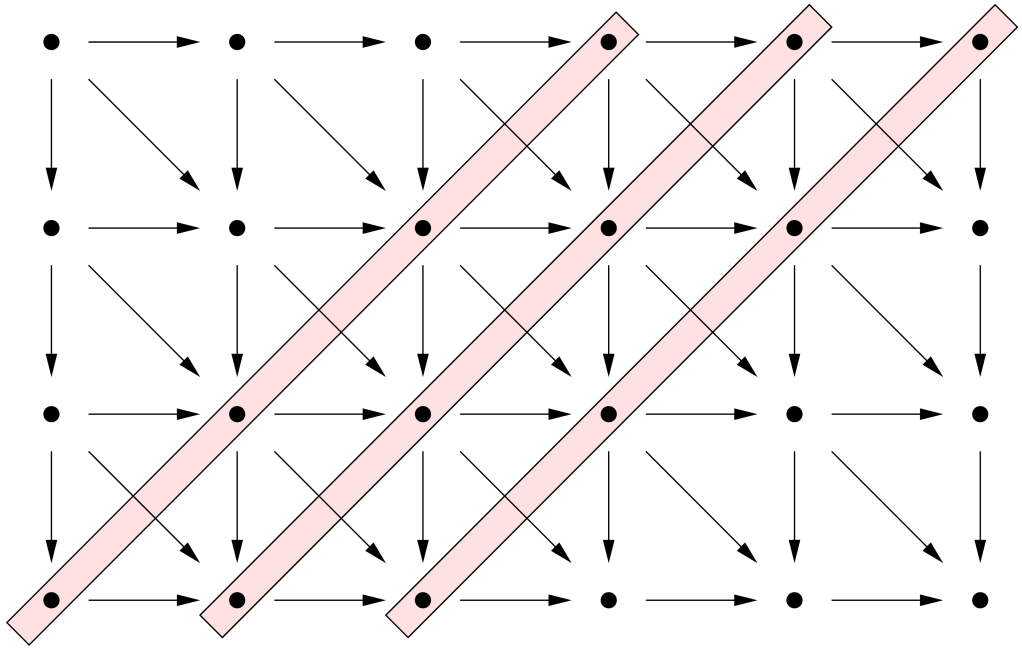
- $\mathcal{O}(mn)$ cellules
- calcul simultané de m cellules
- espace $\mathcal{O}(m)$



En logiciel

Calcul exhaustif :

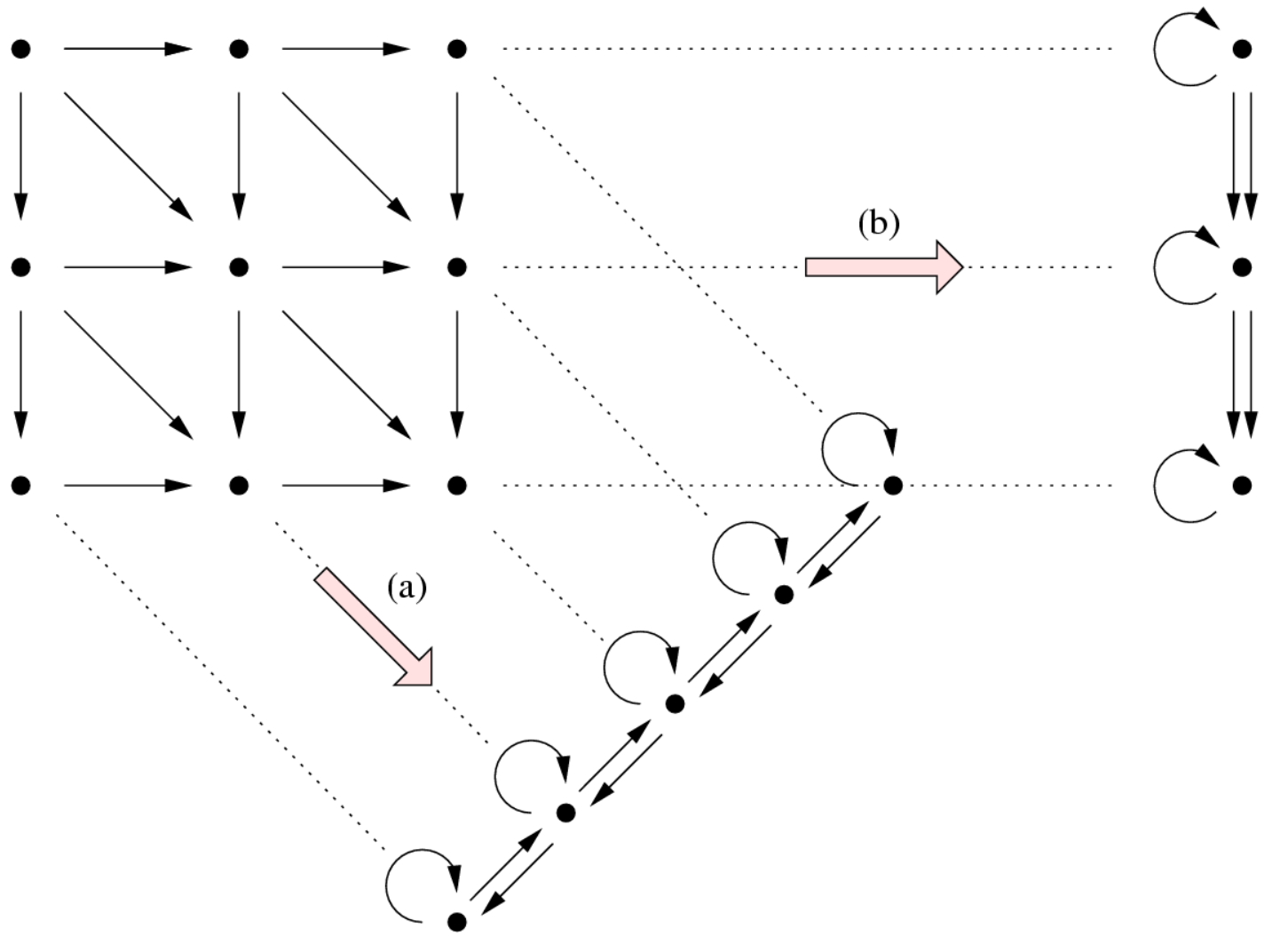
- $\mathcal{O}(mn)$ cellules
- calcul simultané de m cellules
- espace $\mathcal{O}(m)$



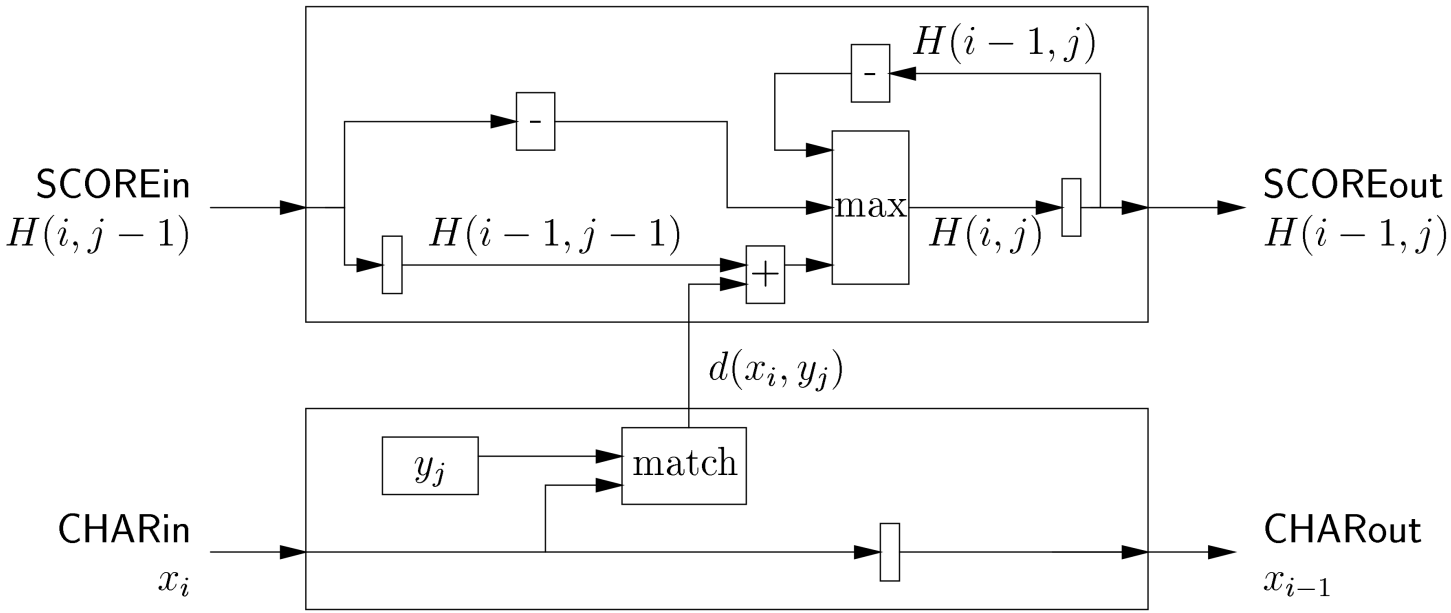
Algorithmes sous-quadratiques ?

- Masek et Paterson (1980) :
  $\mathcal{O}(n^2/\log n)$ pour scores rationnels
- Crochemore, Landau et Ziv-Ukelson (2002) :
  $\mathcal{O}(hn^2/\log n)$ (h entropie de la séquence)

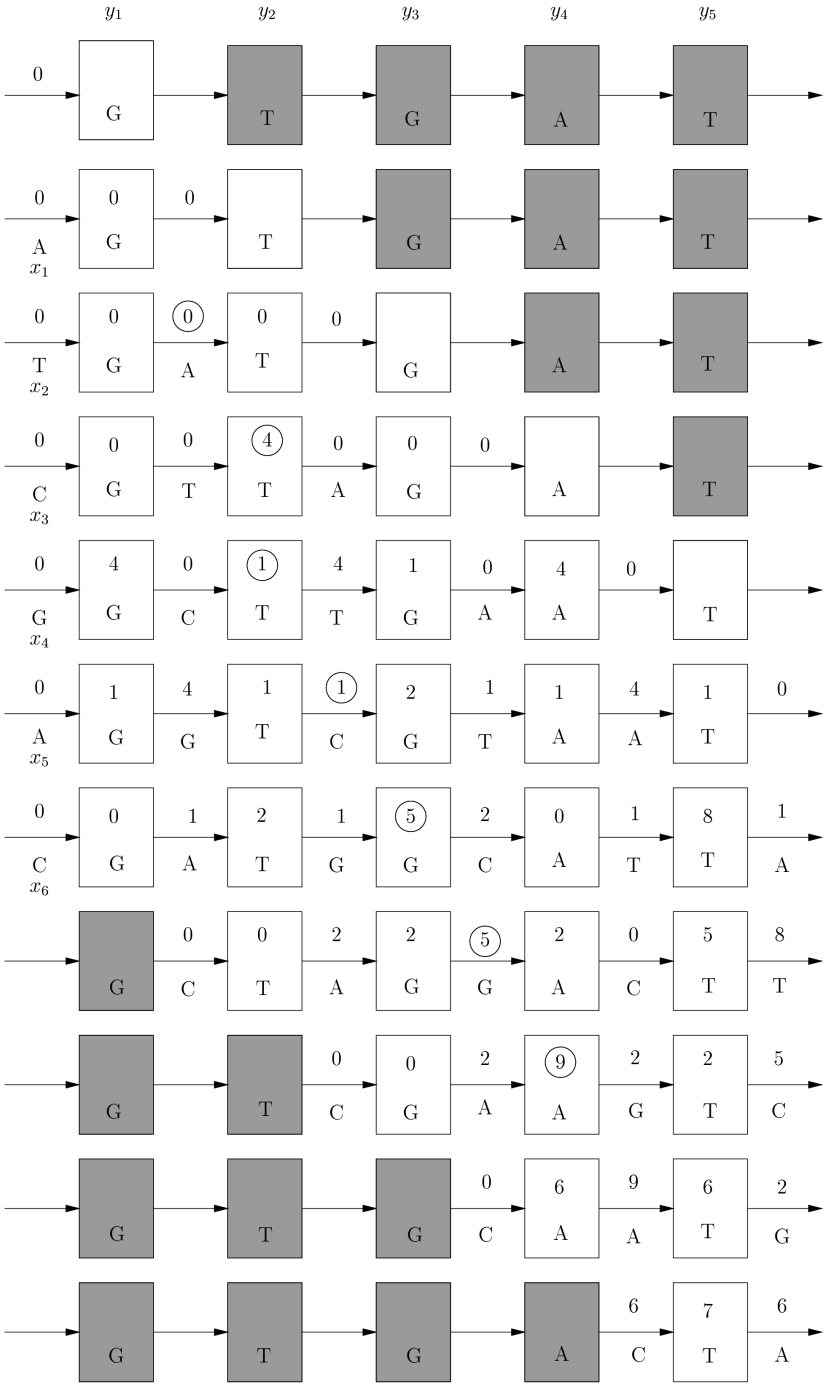
Projection sur une architecture systolique



Architecture systolique unidirectionnelle



Réseau de cellules identiques



(Splash, Hoang et al., 1993)

$$X = \text{ATCGAC}$$

$$Y = \text{GTGAT}$$

La chaîne Y est déjà présente.

Score optimal $\longrightarrow$ 9

Calcul en $\mathcal{O}(m+n)$ cycles
sur $min(m+1, n+1)$ cellules.

Encodage modulo

Distance “nombre d’insertions et de délétions” :

$$\begin{cases} g = -1 \\ \forall \alpha,\ d(\alpha, \alpha) = 0 \\ \forall \alpha, \beta,\ \alpha \neq \beta,\ d(\alpha, \beta) = -2 \end{cases}$$

Encodage modulo

Distance “nombre d’insertions et de délétions” :

$$\begin{cases} g = -1 \\ \forall \alpha,\ d(\alpha, \alpha) = 0 \\ \forall \alpha, \beta,\ \alpha \neq \beta,\ d(\alpha, \beta) = -2 \end{cases}$$

Propriété (Lipton and Lopresti, 1985) :

$$H(i,j) = H(i-1,j) \pm 1 \qquad H(i,j) = H(i,j-1) \pm 1$$

$$\longrightarrow \quad H(i,j) - H(i-1,j-1) \in \{-2, 0\}.$$

Encodage modulo

$$H(i,j) = H(i-1,j) \pm 1 \qquad H(i,j) = H(i,j-1) \pm 1$$

$$\longrightarrow \quad H(i,j) - H(i-1,j-1) \in \{-2, 0\}.$$

Réécriture de l'équation Needleman-Wunsch :
$\forall i, j, ij \neq 0$:

$$H(i,j) = \begin{cases} H(i-1,j-1) & \text{si } H(i-1,j) = H(i-1,j-1) + 1 \\ & \text{ou } H(i,j-1) = H(i-1,j-1) + 1 \\ & \text{ou } (x_i = y_j) \\ H(i-1,j-1) - 2 & \text{sinon} \end{cases}$$

$\longrightarrow$ seulement 1 ou 2 bits pour représenter le score

Architectures proposées

Encodage modulo

- ASIC : P-NAC (1987)
- FPGA : Splash (1991), Splash-2 (1993)

Architectures proposées

Encodage modulo

- ASIC : P-NAC (1987)
- FPGA : Splash (1991), Splash-2 (1993), HokieGene (2003), Yu, Kwong, Lee, et Leong (2003), Guccione et Keller (JBits, 2002)

Architectures proposées

Encodage modulo

- ASIC : P-NAC (1987)
- FPGA : Splash (1991), Splash-2 (1993), HokieGene (2003), Yu, Kwong, Lee, et Leong (2003), Guccione et Keller (JBits, 2002)

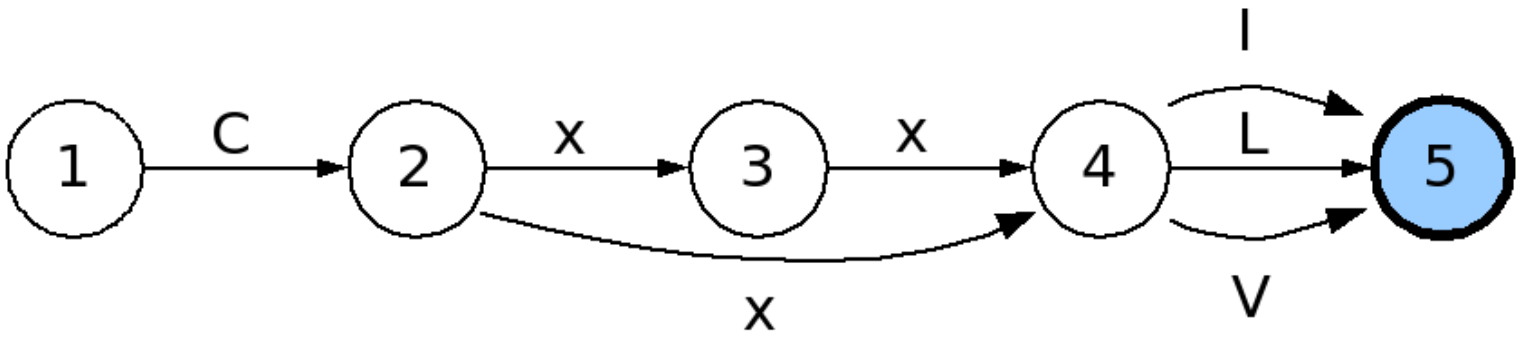
Calcul SW générique

- ASIC : Bisp (1991), Samba, Kestrel, Swasad
- FPGA : Yamaguchi & co (2002), Dydel (2004), Weaver (2003, retiming), Oliver and Schmidt (2004, gaps affines)

- Van Court et Herbordt, 2004

Motifs, expressions régulières et automates

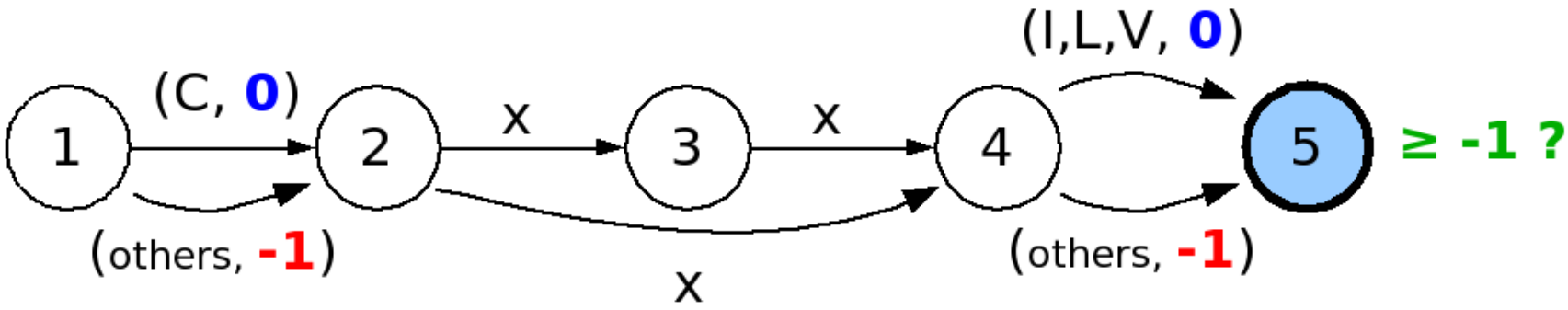
- Syntaxe PROSITE [Bucher, Bairoch 94] :
  C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H
- Expressions régulières, recherche exacte
  ☞ automates non-déterministes (NFA)



C-x(1,2)-[ILV]

Motifs, expressions régulières et automates

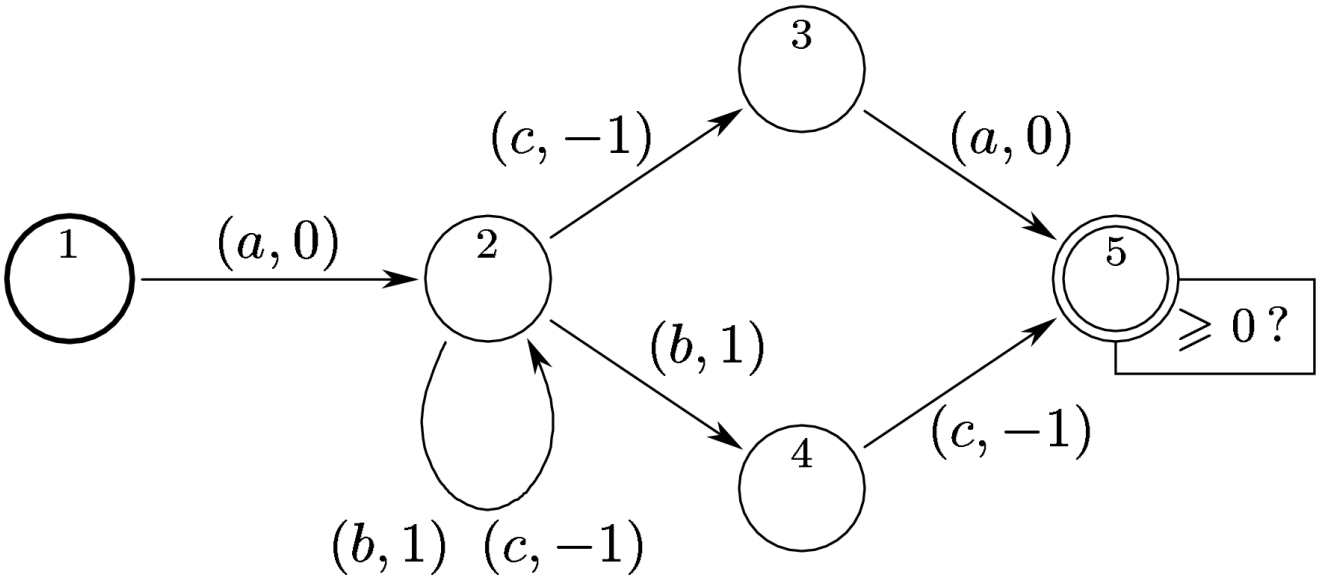
- Syntaxe PROSITE [Bucher, Bairoch 94] :
  C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H
- Expressions régulières, recherche avec erreurs
  ☞ automates pondérés (WFA)



C-x(1,2)-[ILV] avec 1 erreur de substitution

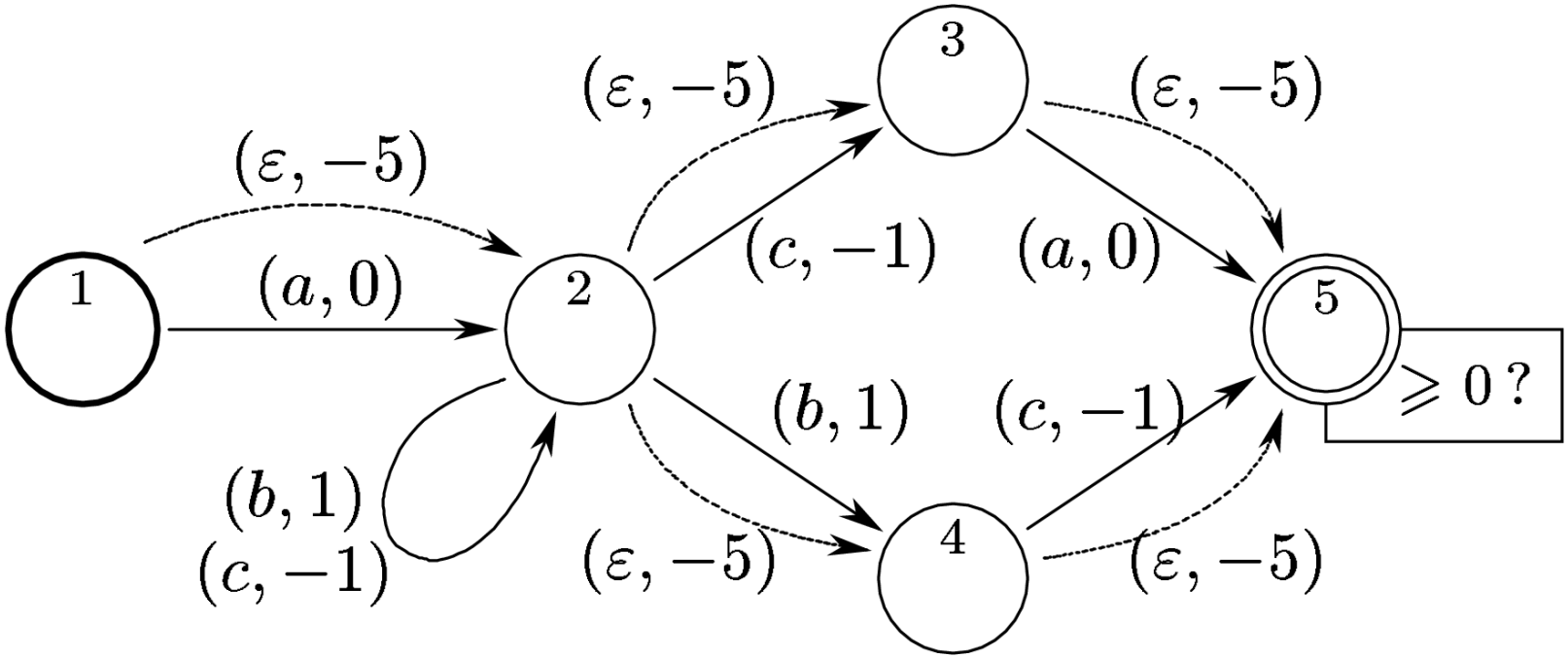
Automates pondérés (WFA)

Weighted Finite Automata ▸ définition



$a\,(b|c)^*\,(ca\,|\,bc)$ avec plus de b que de c

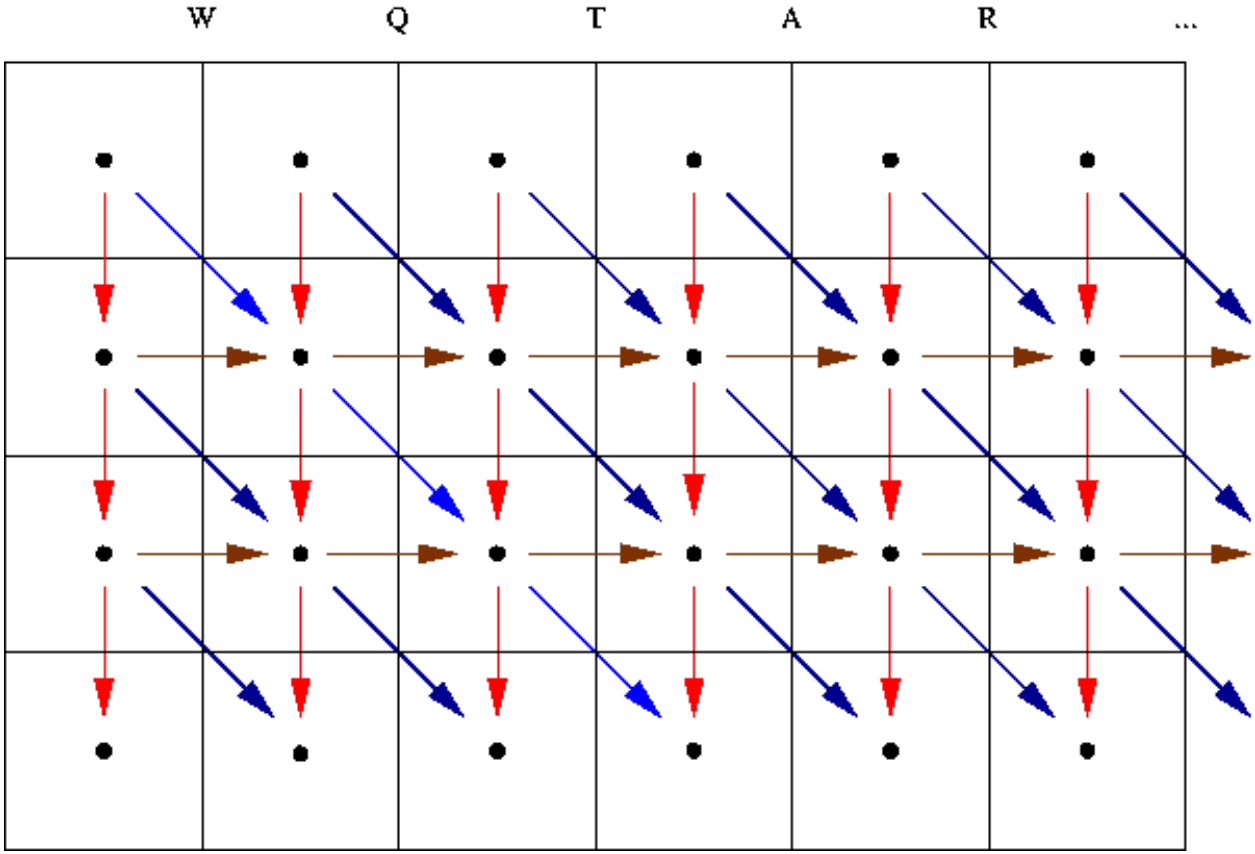
Automates pondérés avec ε-transitions



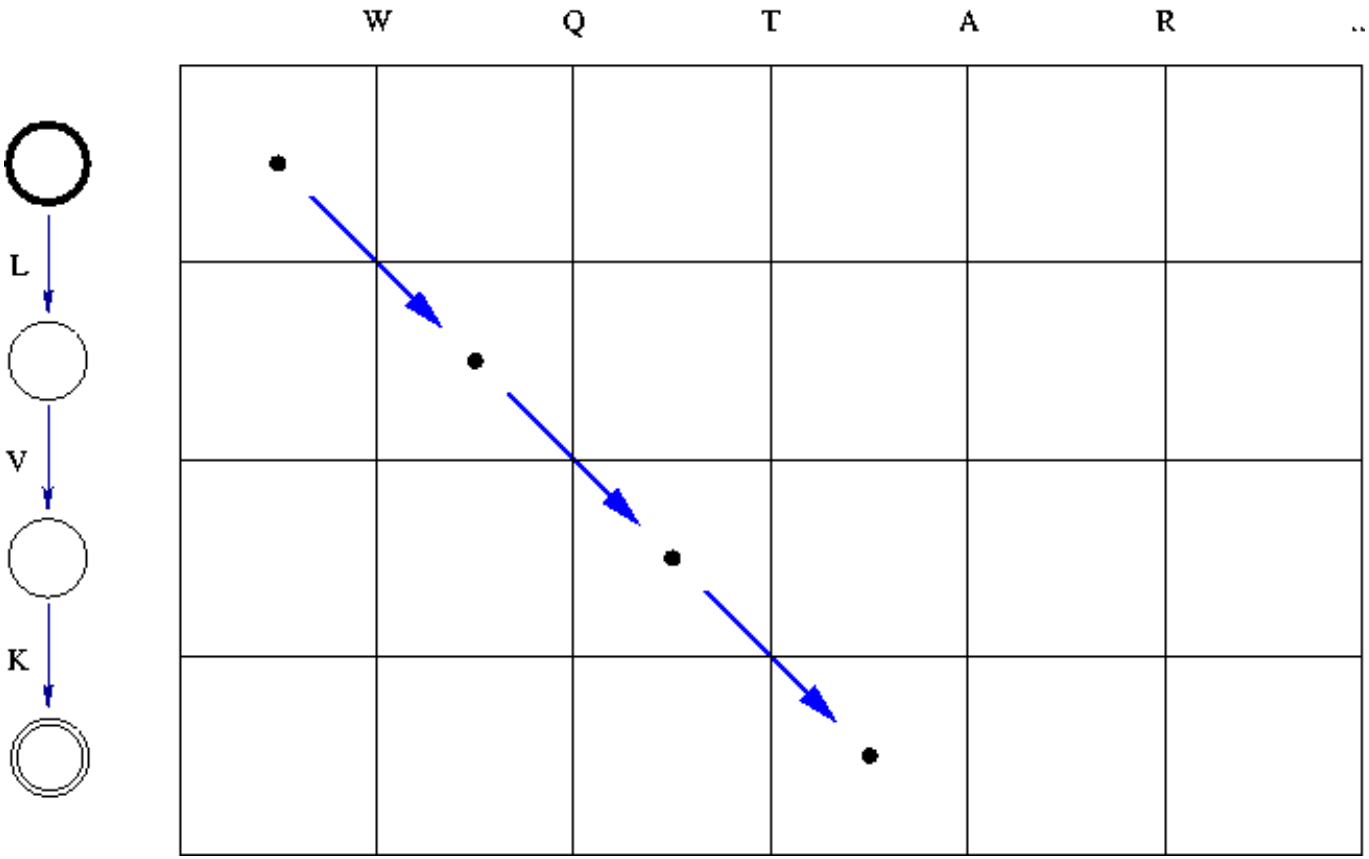
Chaque transition est doublée par une ϵ-transition.

$$c_\epsilon = -5$$

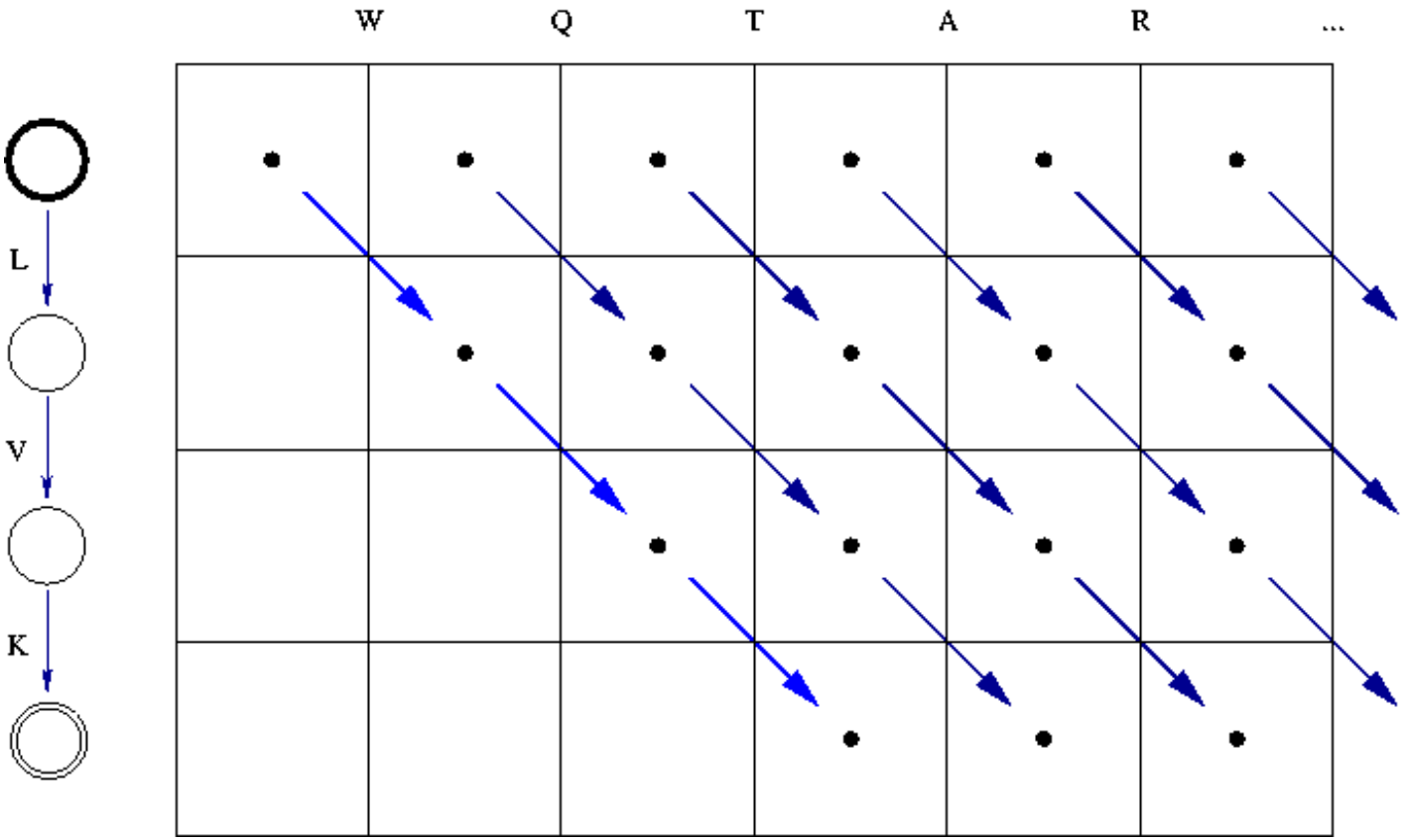
Automates pondérés et programmation dynamique



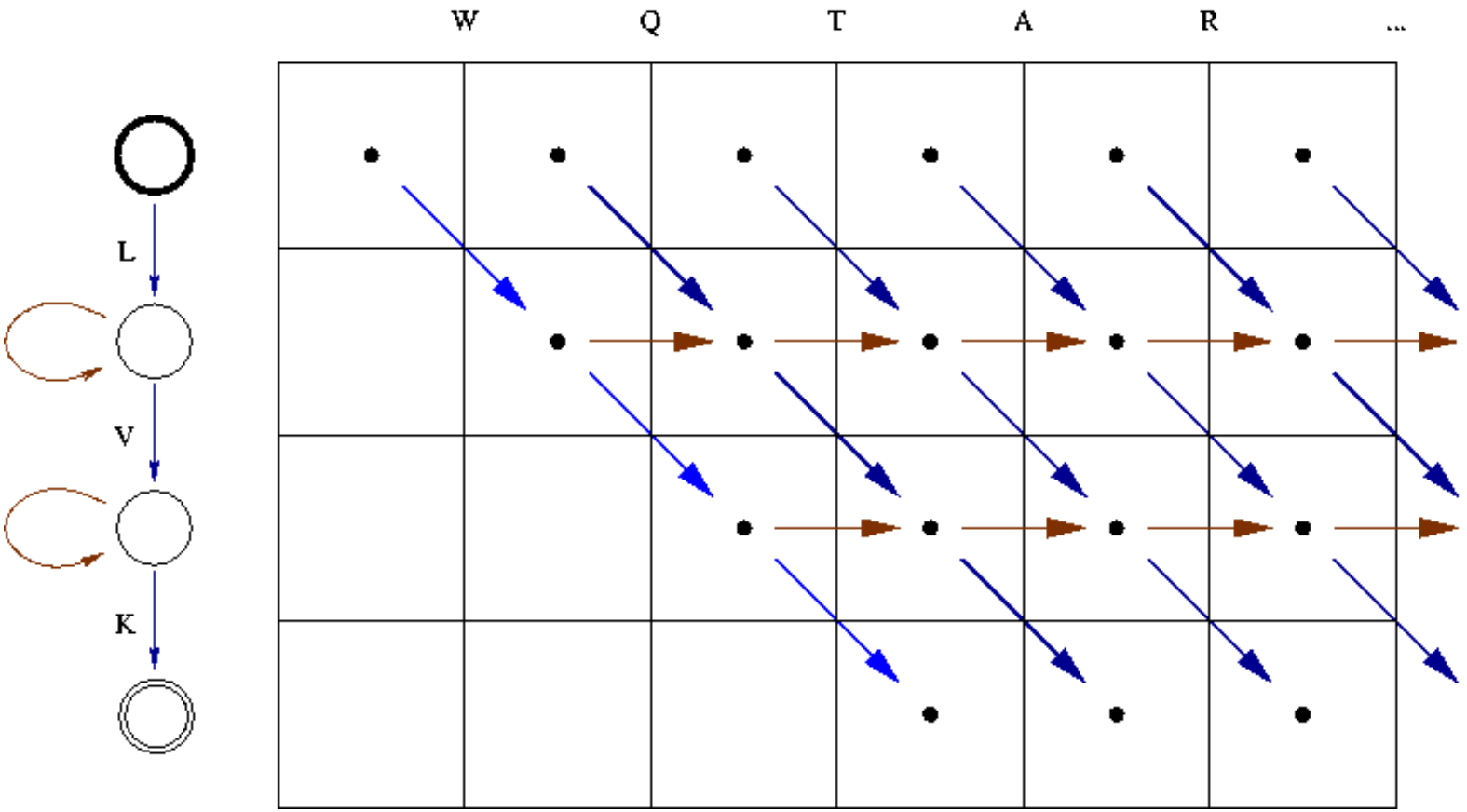
Automates pondérés et programmation dynamique



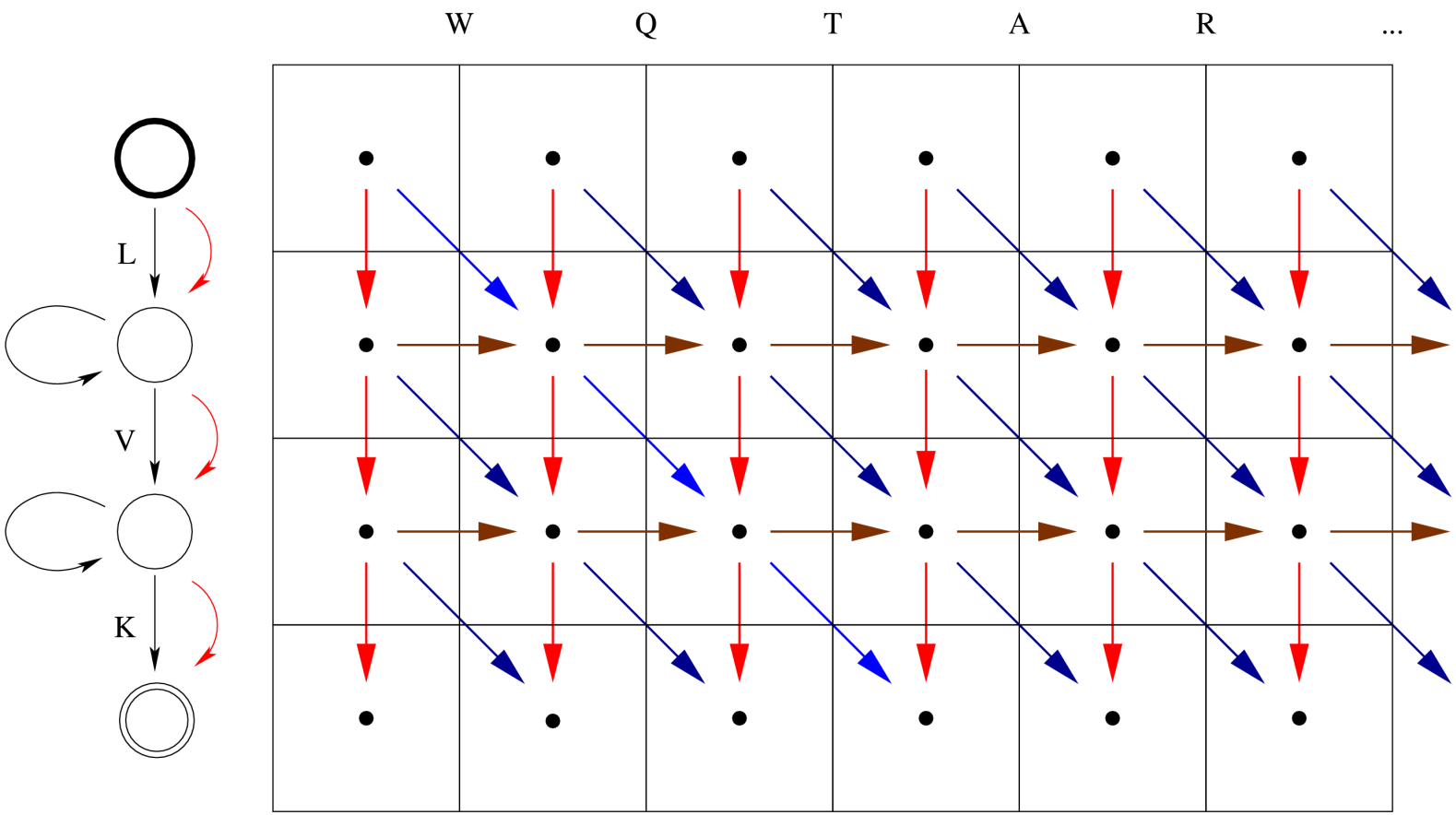
Automates pondérés et programmation dynamique



Automates pondérés et programmation dynamique



Automates pondérés et programmation dynamique



Automates pondérés

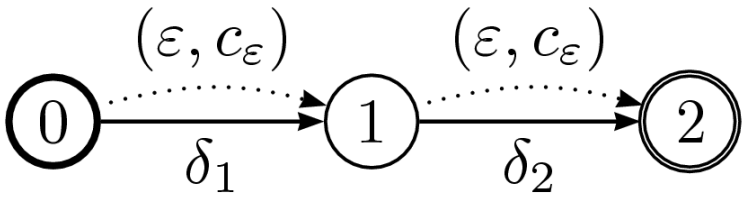
- motifs, matrices de fréquences ou de poids
- HMM
- programmation dynamique
- quelques grammaires

Suppression des ε-transitions

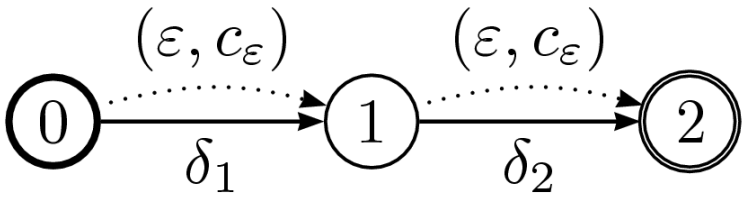
Enlever les ε-transitions par un algorithme usuel [Mohri] ?

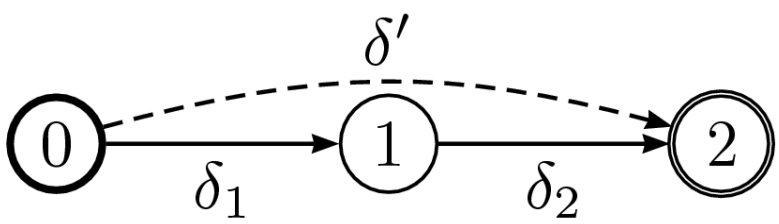
- jusqu'à $\mathcal{O}(t^2)$ nouvelles transitions
- et la place sur le FPGA est limitée (75-100)...

Création de nouvelles transitions



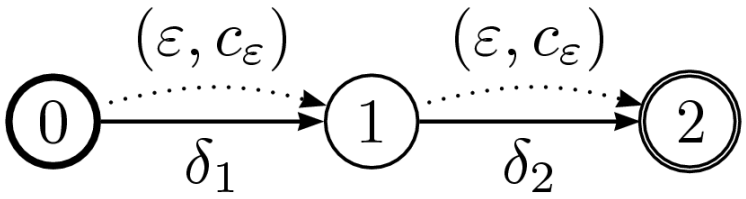
Création de nouvelles transitions

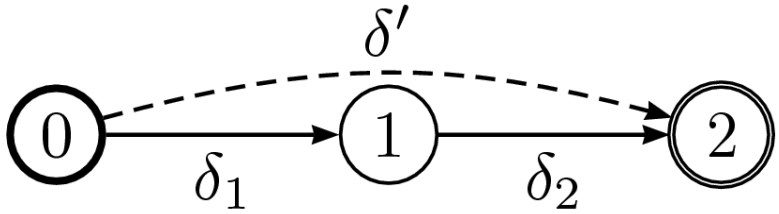




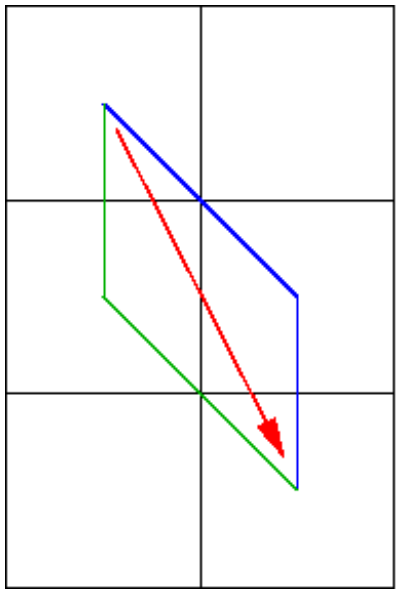
$$\delta'(\alpha) = \max\left(\delta_1(\alpha) + c_\varepsilon,\ c_\varepsilon + \delta_2(\alpha)\right)$$

Création de nouvelles transitions

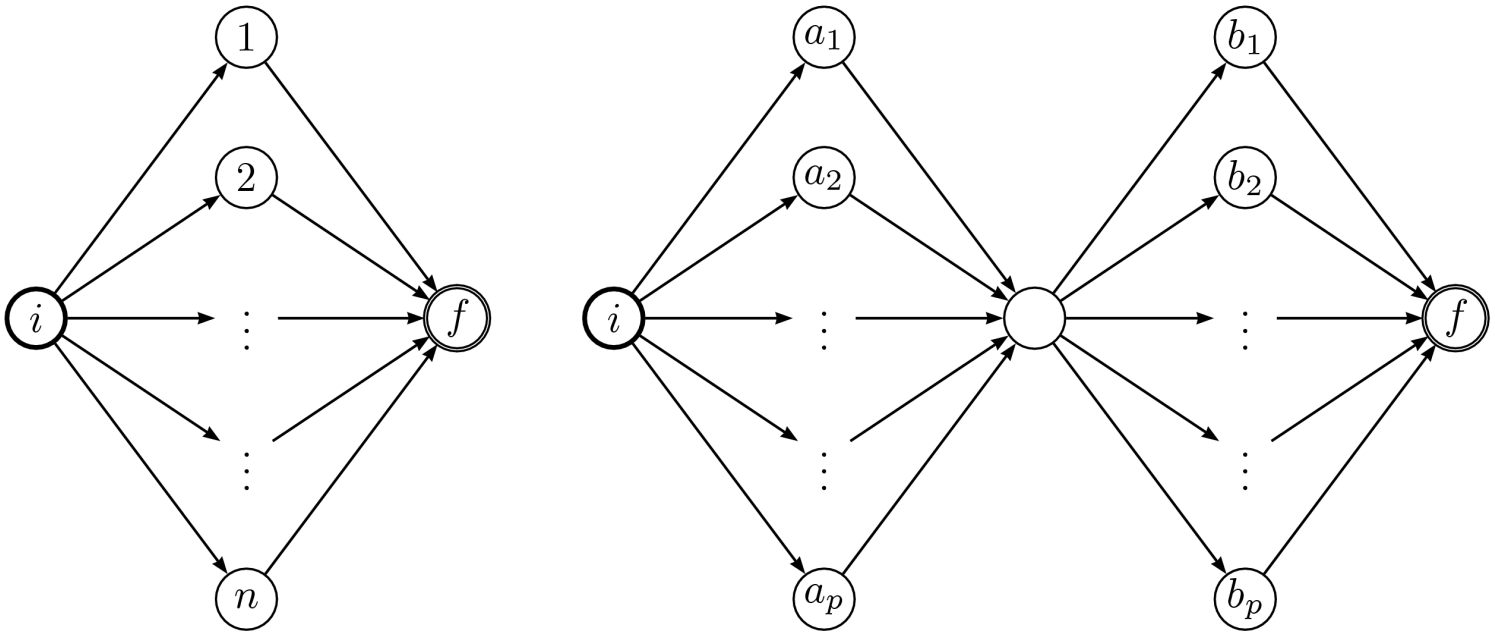




$$\delta'(\alpha) = \max\left(\delta_1(\alpha) + c_\varepsilon,\ c_\varepsilon + \delta_2(\alpha)\right)$$



Zoologie

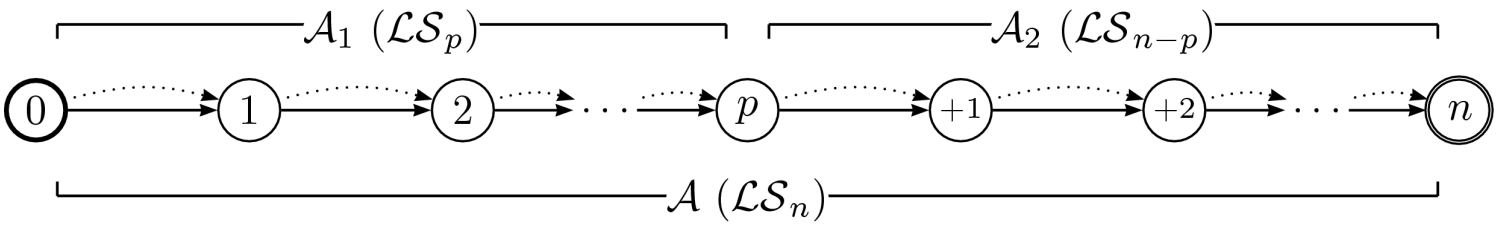


Selon les cas, $\mathcal{O}(1)$ à $\Omega(t^2)$ nouvelles transitions sont nécessaires ☞ il faut se restreindre à des formes d'automates plus simples.

Suppression récursive des ε-transitions

Propriété

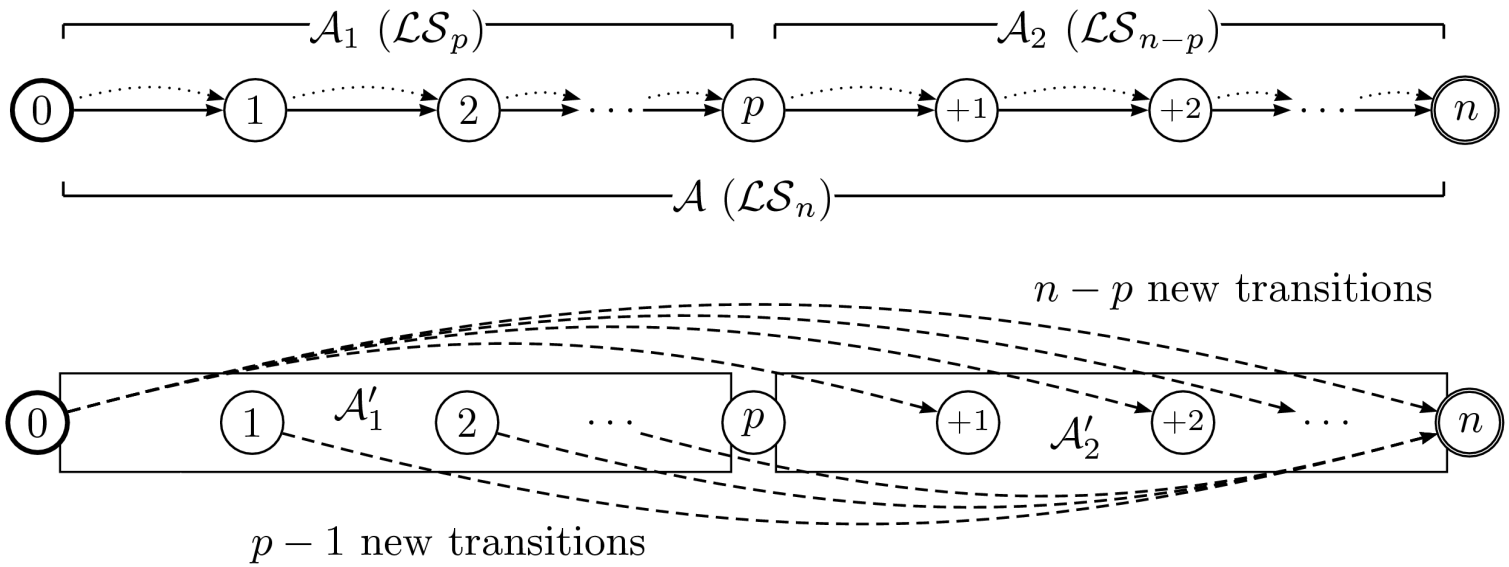
Pour des automates rectilignes avec t transitions, on peut enlever les ε-transitions au moyen de $\mathcal{O}(t \log t)$ nouvelles transitions.



Suppression récursive des ε-transitions

Propriété

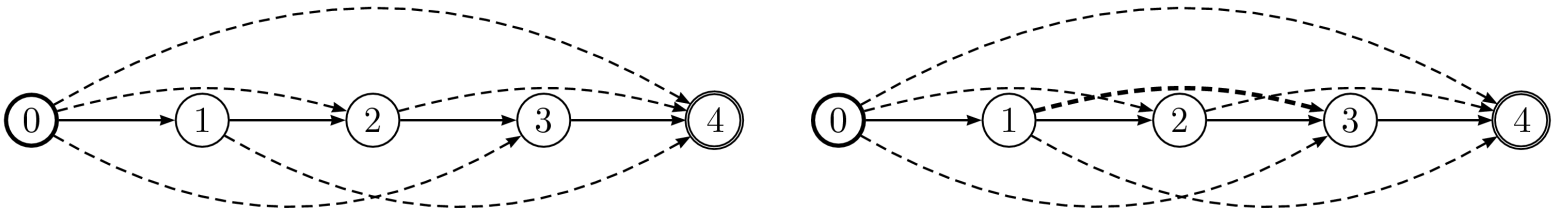
Pour des automates rectilignes avec t transitions, on peut enlever les ε-transitions au moyen de $\mathcal{O}(t \log t)$ nouvelles transitions.



Suppression récursive des ε-transitions

Propriété

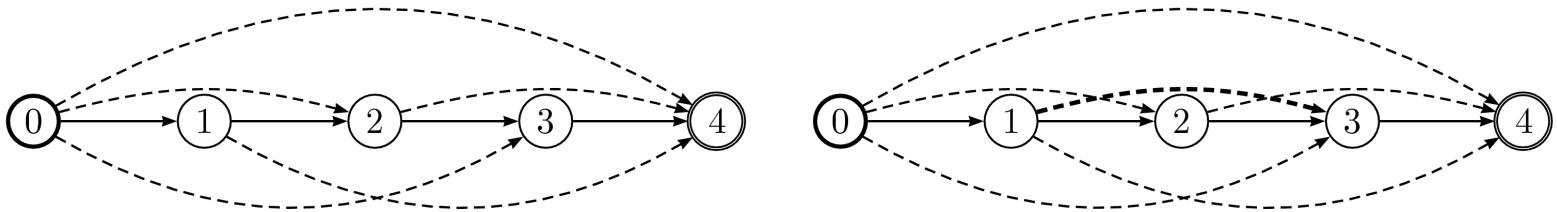
Pour des automates rectilignes avec t transitions, on peut enlever les ε-transitions au moyen de $\mathcal{O}(t \log t)$ nouvelles transitions.



Suppression récursive des ε-transitions

Propriété

Pour des automates rectilignes avec t transitions, on peut enlever les ε-transitions au moyen de $\mathcal{O}(t \log t)$ nouvelles transitions.



☞ Cette méthode permet de mettre sur le FPGA des WFA correspondant à des motifs plus grands.

Équivalence par chemins

- Un *chemin* est une suite de transitions étiquetées par des éléments de $\Sigma \cup \{\varepsilon\}$.
- Un chemin est *supérieur* à un autre si il a le même état initial, le même état final, le même label et s'il est de poids supérieur.
- Un chemin est *fermé* à gauche s'il commence par un état initial ou par un caractère différent de ε.
- Deux automates $\mathcal{A}$ et $\mathcal{A}'$ sont *équivalents par chemins* si chaque chemin fermé non étiqueté par ε de $\mathcal{A}$ a un chemin supérieur dans $\mathcal{A}'$ et réciproquement.

Deux automates équivalents par chemins calculent les mêmes poids pour les mots non vides.

Borne minimale

Théorème

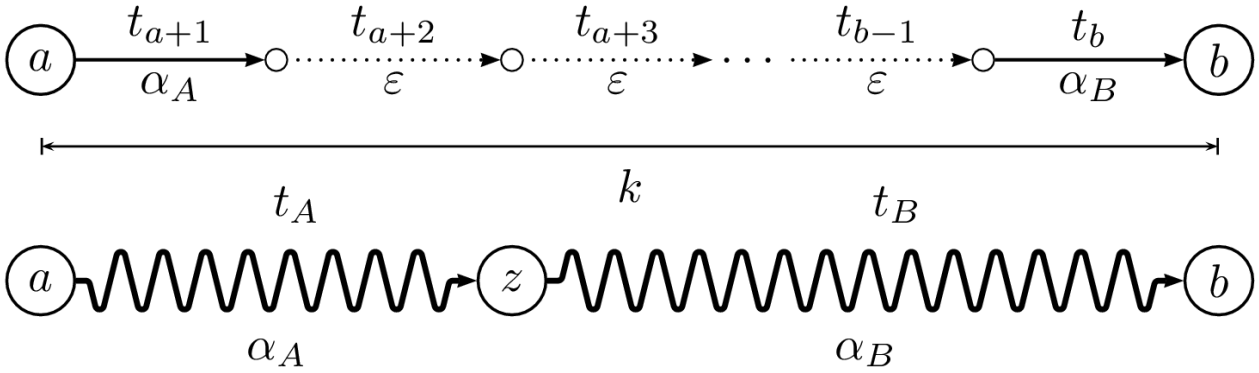
La borne $\Theta(t \log t)$ est optimale si on impose que les automates soient équivalents par chemins.

Borne minimale

Théorème

La borne $\Theta(t \log t)$ est optimale si on impose que les automates soient équivalents par chemins.

- Il n'y a pas de transitions arrière.



- t_A ou t_B doit avoir un span inclus dans $\{\lceil k/2 \rceil, \ldots, k-1\}$.

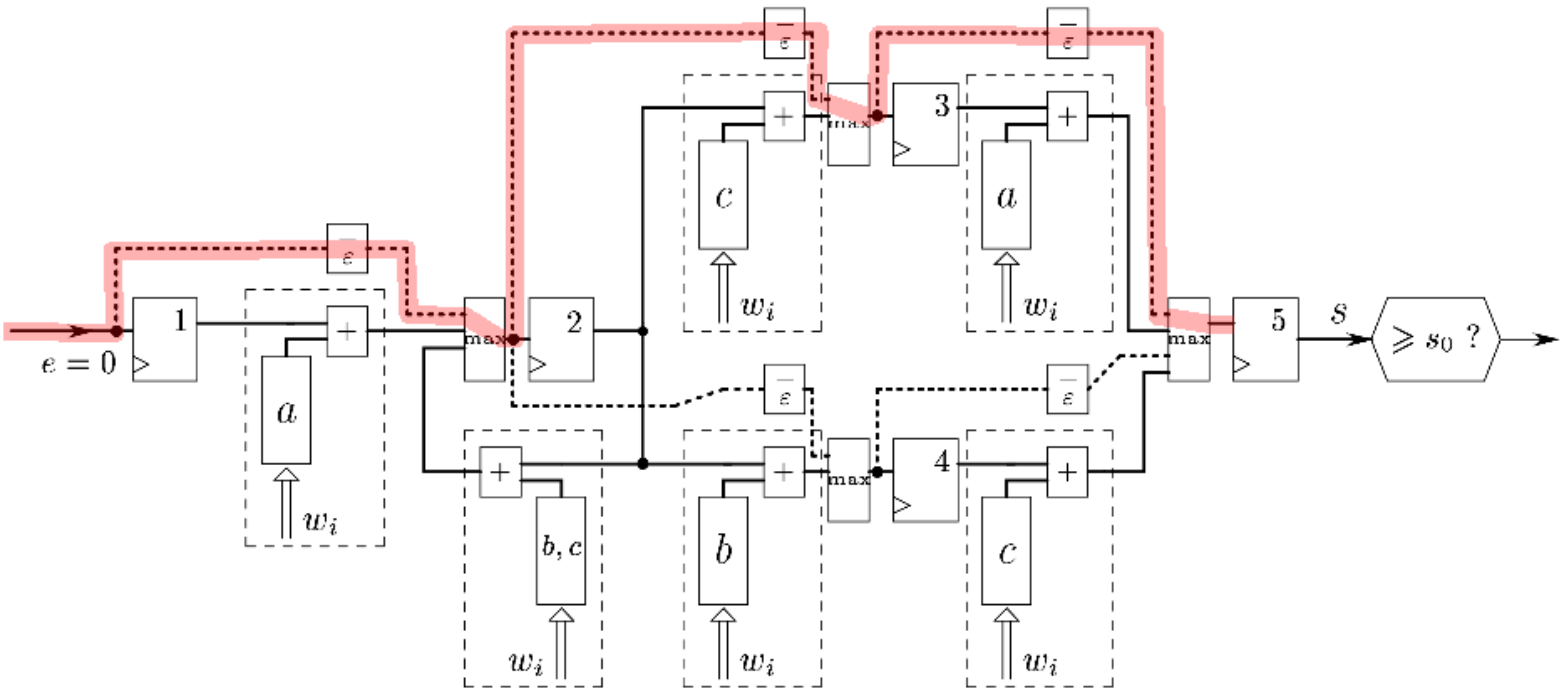
Travaux sur le même thème

Hromkovic a montré qu'il existe des expressions rationnelles de taille $\mathcal{O}(n)$ telles que tout NFA (sans ε-transitions) les reconnaissant soit de taille $\Omega(n \log n)$:

$$(a + \varepsilon)(b + \varepsilon)(c + \varepsilon)...$$

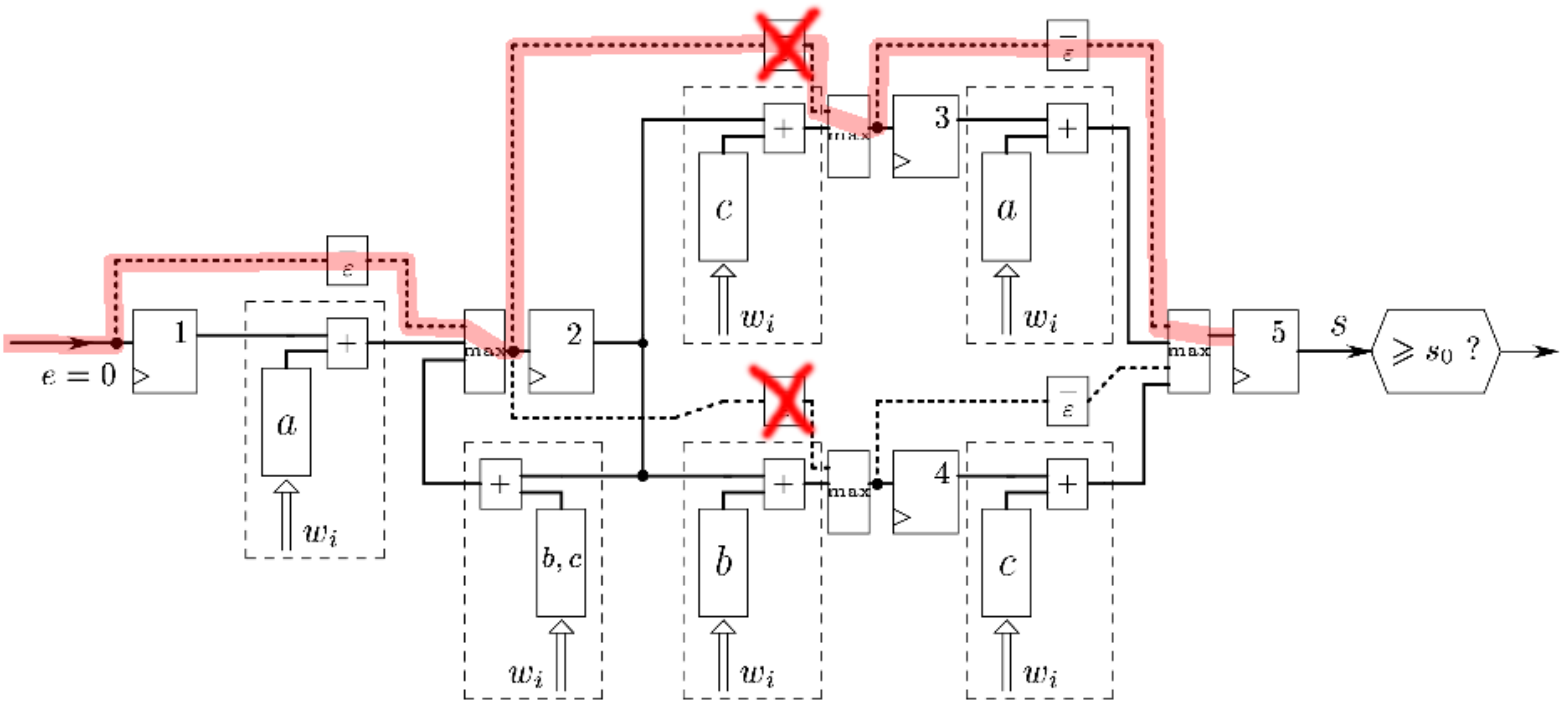
D'autres travaux ont proposé des meilleures bornes, mais toujours en partant d'expressions rationnelles.

Suppressions “à quelques ε-transitions près”



Le chemin critique permet tout de même de réaliser quelques ε-transitions successives (1 à 4).

Suppressions “à quelques ε-transitions près”



Le chemin critique permet tout de même de réaliser quelques ε-transitions successives (1 à 4).

Suppressions “à quelques ε-transitions près”

Le chemin critique permet tout de même de réaliser quelques ε-transitions successives (1 à 4). On peut espérer économiser quelques transitions lors de la suppression.

Suppressions "à quelques ε-transitions près"

Le chemin critique permet tout de même de réaliser quelques ε-transitions successives (1 à 4). On peut espérer économiser quelques transitions lors de la suppression.

Propriété

Supprimer les ε-transitions d'un automate rectiligne

- sous contrainte d'équivalence par chemins
- et en laissant au plus n_ε ε-transitions successives

demande au moins $\Omega(t \log(t/n_\varepsilon))$ nouvelles transitions.

Suppression dans des automates quelconques

Question ouverte

Sur un automate quelconque, combien faut-il de nouvelles transitions pour supprimer les ε-transitions ?

Intuition pour les automates acycliques : $\mathcal{O}(\mathcal{W} n \log n)$, où $\mathcal{W}$ est la tranche maximale de l'automate.

Limites des approches logicielles

Goulots d'étranglement :

- puissance de calcul
  ☞ La taille des bases de données augmente plus vite que la puissance de calcul des processeurs
- accès aux données : entrées/sorties

Architectures spécialisées avec FPGAs

- puissance de calcul
- accès aux données : entrées/sorties

Architectures spécialisées avec FPGAs

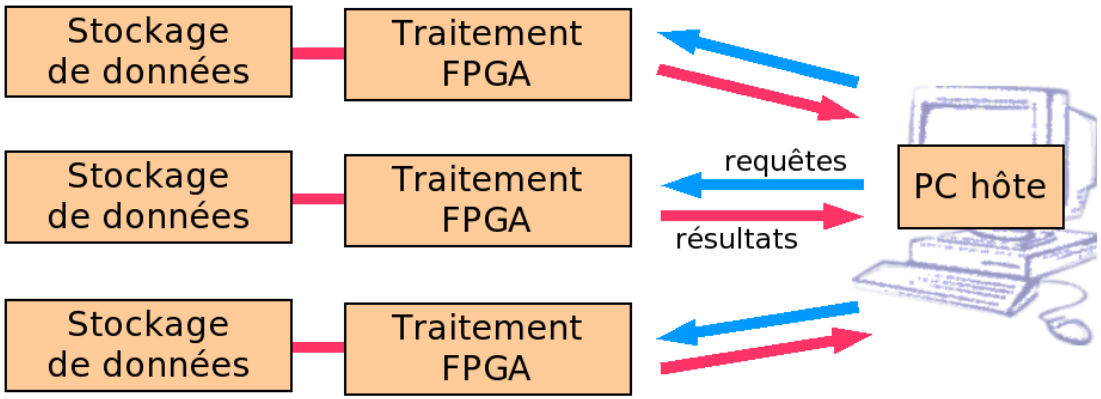
- puissance de calcul
  $\longrightarrow$ paralléliser les calculs répétitifs sur FPGA
- accès aux données : entrées/sorties
  $\longrightarrow$ faciliter les accès aux données

☞ placer des FPGAs à proximité de dispositifs de stockage

Architectures spécialisées avec FPGAs

- puissance de calcul
  $\longrightarrow$ paralléliser les calculs répétitifs sur FPGA
- accès aux données : entrées/sorties
  $\longrightarrow$ faciliter les accès aux données

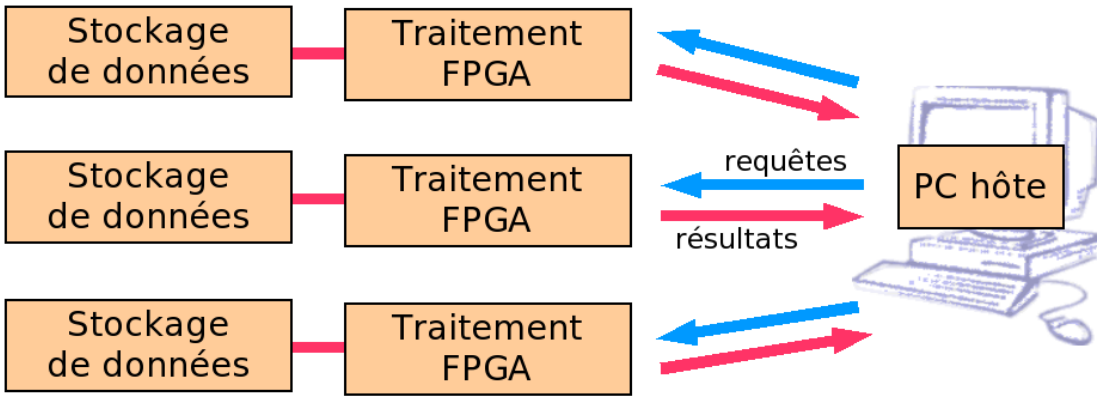
☞ placer des FPGAs à proximité de dispositifs de stockage



Architectures spécialisées avec FPGAs

- puissance de calcul
  $\longrightarrow$ paralléliser les calculs répétitifs sur FPGA
- accès aux données : entrées/sorties
  $\longrightarrow$ faciliter les accès aux données

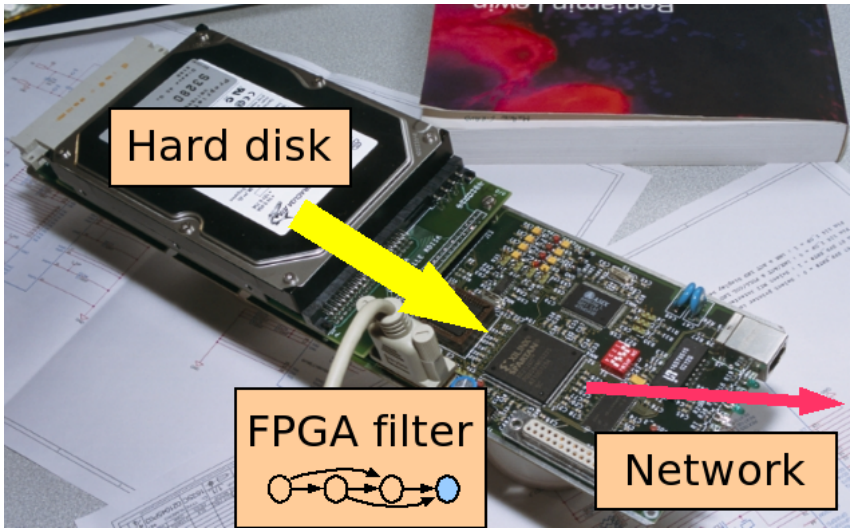
☞ placer des FPGAs à proximité de dispositifs de stockage

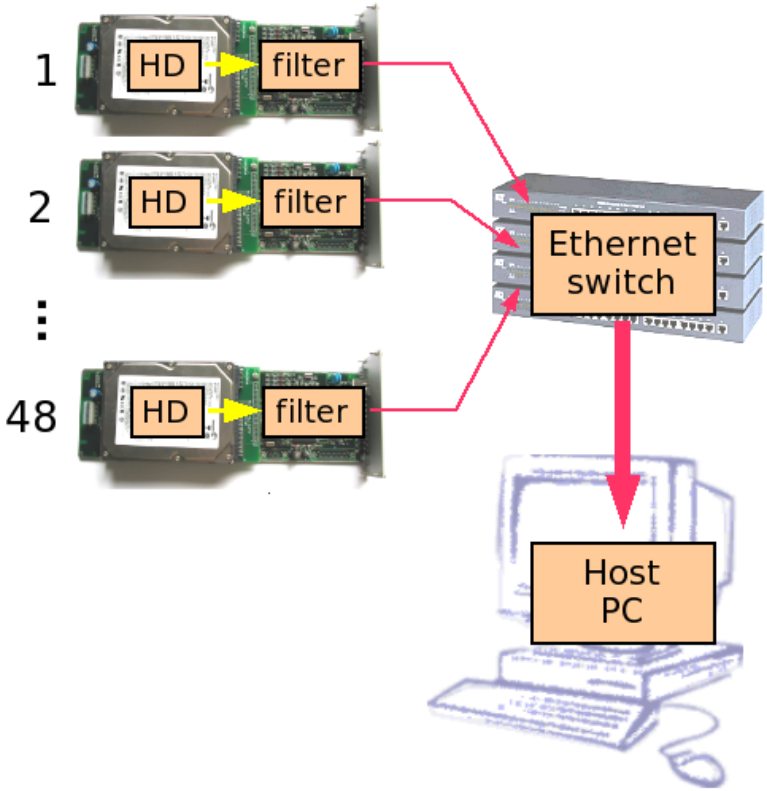


- avec un disque dur
  R-disk
- avec de la mémoire
  ReMiX

Accélération de la recherche sur FPGA

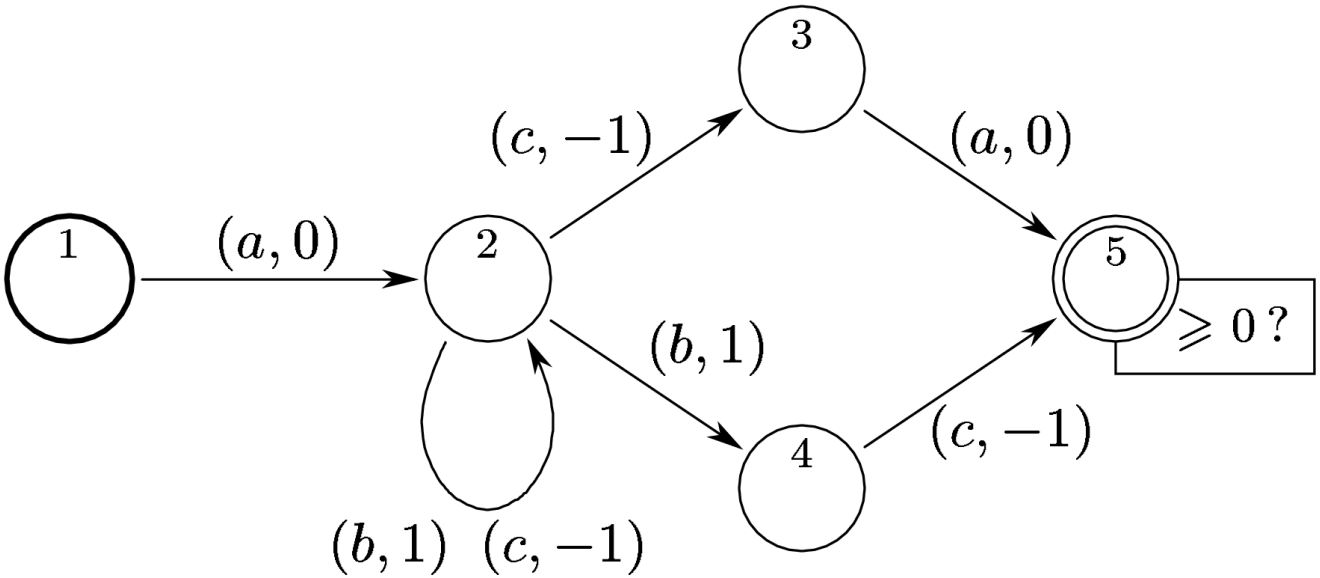
Filtrage de données directement à la sortie des disques durs





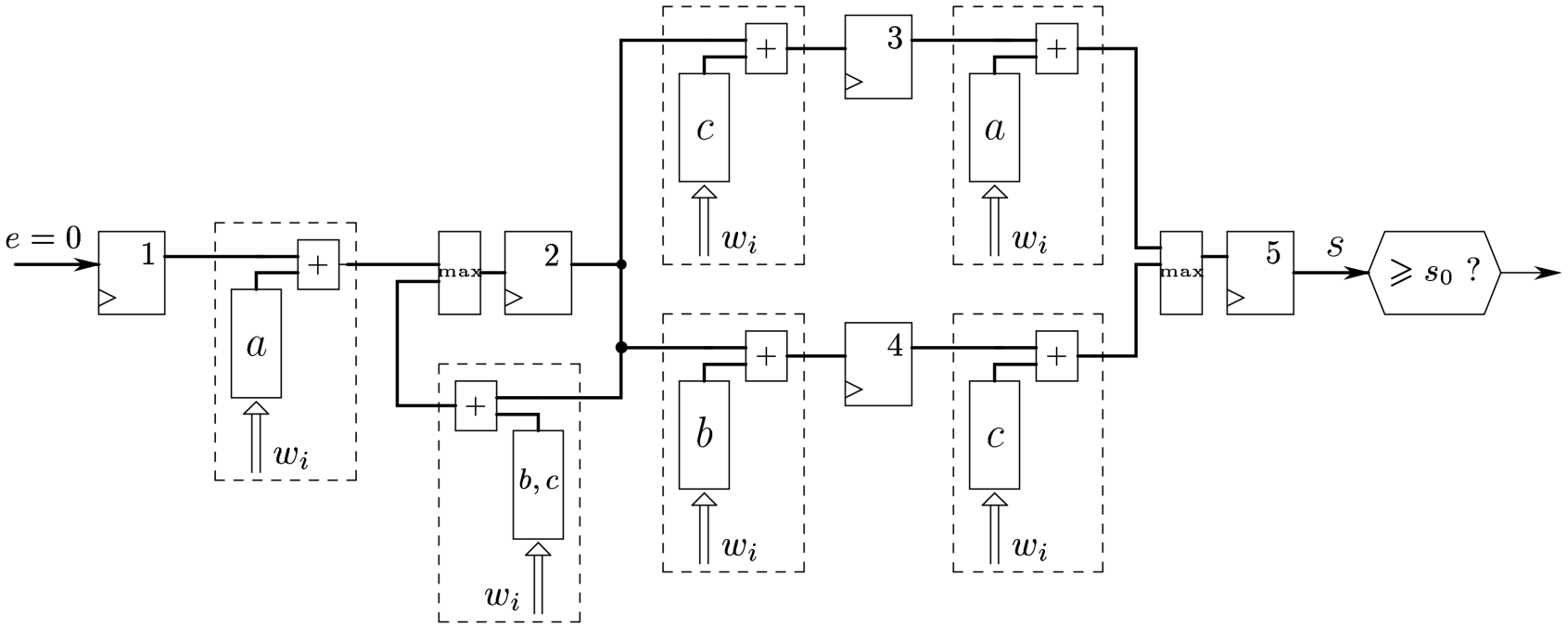
Système “économique” : $<$ 200 euros de composants

Accélération de la recherche sur FPGA



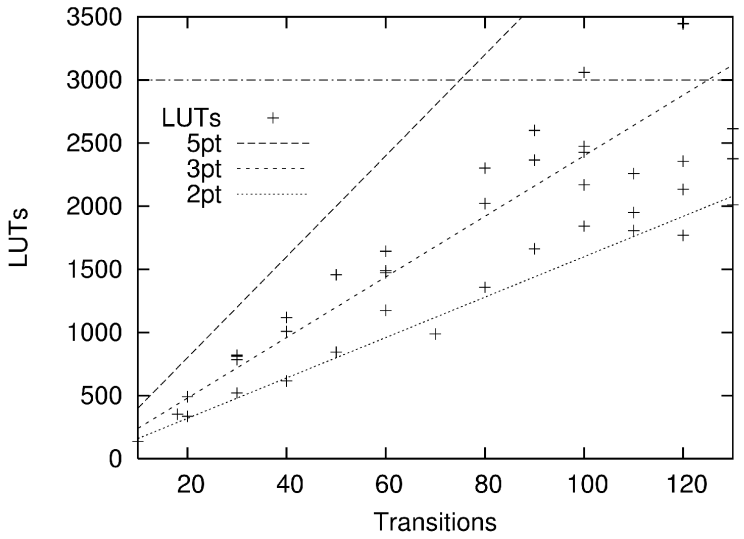
$$\mathcal{L}_2 = \{w \in \mathcal{L}_1 \mid |w|_b \geq |w|_c\}$$

Accélération de la recherche sur FPGA



- Poids à p bits, additions, maximums ▸ généralisation
- 1 caractère par cycle à 40 MHz

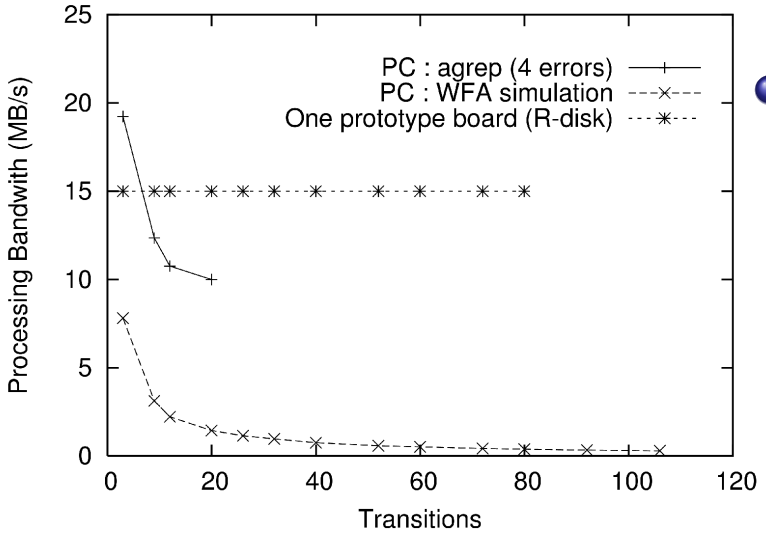
Limites de taille dans le FPGA



- $\leq 5p$ LUTs par transition : au maximum $600/p$ transitions
- Poids de $p = 8$ bits
  - maximum : 75 transitions (Spartan II)
  - en réalité : 80-120 transitions

- Répartition des transitions
  - 1 motif avec 75 transitions
  - Motifs protéiques sur des banques nucléiques
    3 motifs de 25 transitions sur 3 ORFs
  - Spartan 3 : la limite devient 600 transitions

Vitesse



- Parallélisme
  - grain fin : 6 Gop/s (Spartan II, 40 MHz)
  - grain fort : R-disk 48 replication ou distribution
- Vitesse d'entrée : 16 Mo/s sur une carte
  - 4 à 10 fois plus rapide qu'un PC (2 GHz, 728 Mo RAM)
  - Temps de calcul pour EMBL (70 Gbp / 18 Go)
    PC : $>$ 1 heure, 1 carte : 35 min, 48 cartes : 40 s
- Temps de compilation : 3-5 minutes

Partie I
Programmation dynamique

1. Alignements
2. Relations de programmation dynamique
3. Architectures systoliques
4. Automates pondérés avec ε-transitions
5. R-disk

Bibliographie

- D. Lavenier et M. Giraud, chapter “Bioinformatics applications”, *Reconfigurable Computing*, M. Gokhale, P. Graham, 2005
- M. Giraud, L. Noé, G. Kucherov et D. Lavenier, *Recherches de motifs et de similarités en bioinformatique : modélisations, solutions logicielles et matérielles*, MajecSTIC, 2005
- Thèses de L. Noé et de M. Giraud

Suppression d'ε-transitions

Philippe Veber

R-disk

Steven Derrien (R2D2), Gilles Georges, Stéphane Guyetant (CEA), Ludovic L'Hours (R2D2), Mickaël Paty, Frédéric Raimbault (Vannes), Stéphane Rubini (Brest)

Indexation et graines

Laurent Noé, Gregory Kucherov (Loria / LIFL)

Récepteurs olfactifs

Francis Galibert*, Emmanuelle Morin, Jacques Nicolas, Pascale Quignon*, Élodie Retout, Maud Rimbault*,

WAPAM, automates pondérés

Grégory Ranchy, Anne-Sophie Valin

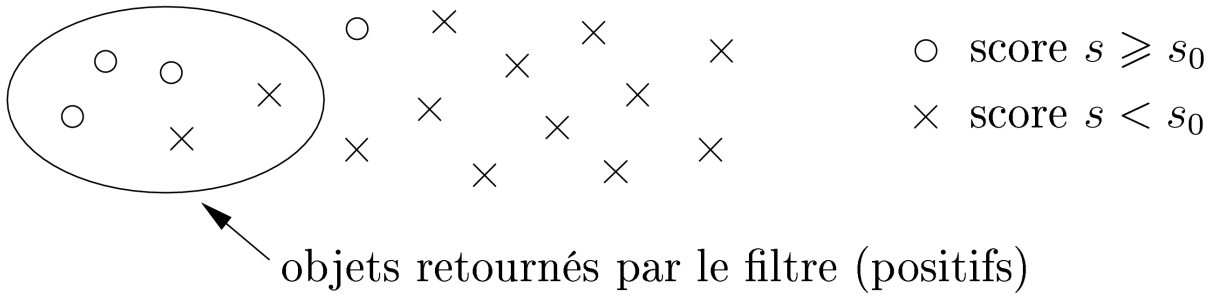
Mathieu Giraud, Dominique Lavenier

* : UMR 6061 (Rennes), Génétique et développement

Partie II
Heuristiques à base de graines

Pourquoi utiliser une heuristique ?

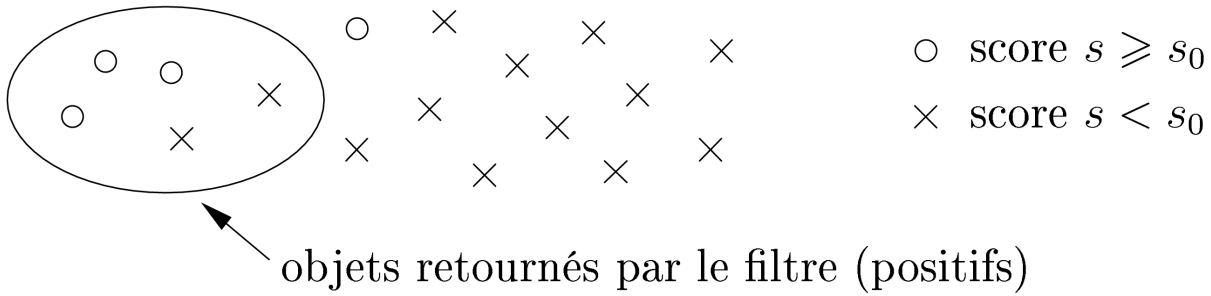
Un calcul d'alignement local est un **filtrage** retournant certaines positions dans la banque.



Vrais positifs $T^{\oplus}$ Faux positifs $F^{\oplus}$
Faux négatifs $F^{\ominus}$ Vrais négatifs $T^{\ominus}$

Pourquoi utiliser une heuristique ?

Un calcul d'alignement local est un **filtrage** retournant certaines positions dans la banque.



Vrais positifs $T^{\oplus}$ Faux positifs $F^{\oplus}$
Faux négatifs $F^{\ominus}$ Vrais négatifs $T^{\ominus}$

$$\text{Selectivité } S_\ell = \frac{(T^{\oplus}+F^{\oplus})}{(T^{\oplus}+F^{\oplus}+T^{\ominus}+F^{\ominus})}$$

$$\text{Sensibilité } S_n = \frac{T^{\oplus}}{(T^{\oplus}+F^{\ominus})} \qquad \text{Spécificté } S_p = \frac{T^{\oplus}}{(T^{\oplus}+F^{\oplus})}$$

BLAST, l'article le plus cité de tous les temps

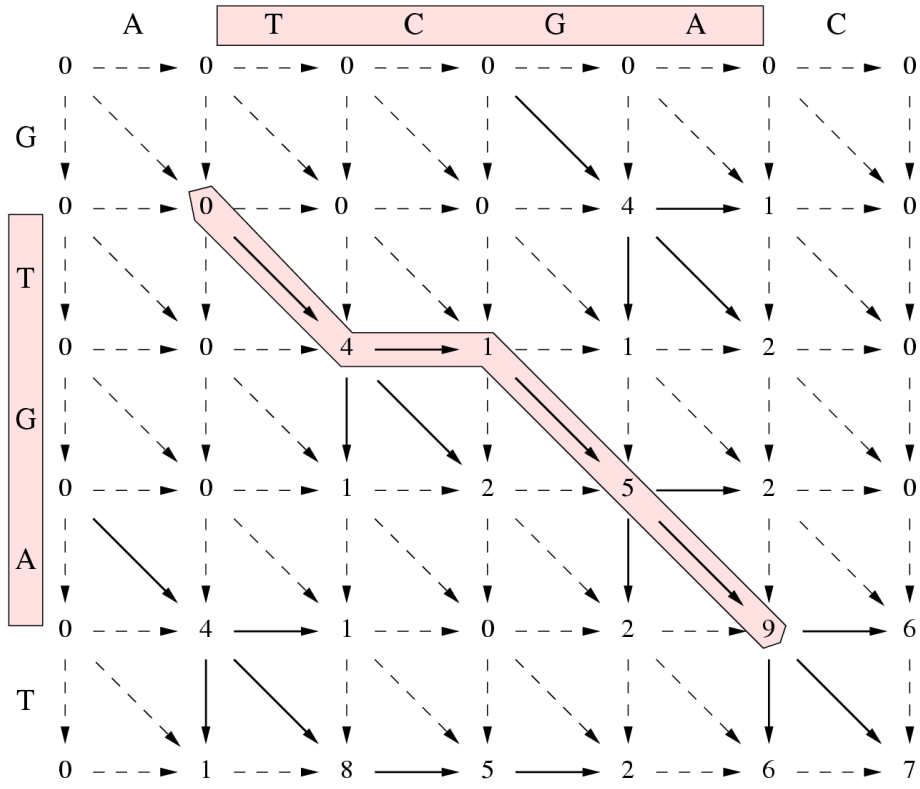
Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool (1990)
Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST : a new generation of protein database search programs (1997)

BLAST, l'article le plus cité de tous les temps

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool (1990)
Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST : a new generation of protein database search programs (1997)

Requête ...	nucléique (Σ_4)	protéique (Σ_{20})
Banque nucléique (Σ_4)	*blastn* / *tblastx*	*blastn*
Banque protéique (Σ_{20})	*blastx*	*blastp*

- *blastn* compare directement deux chaînes nucléiques,
- *tblastx* compare les traductions protéiques sur les 6 phases

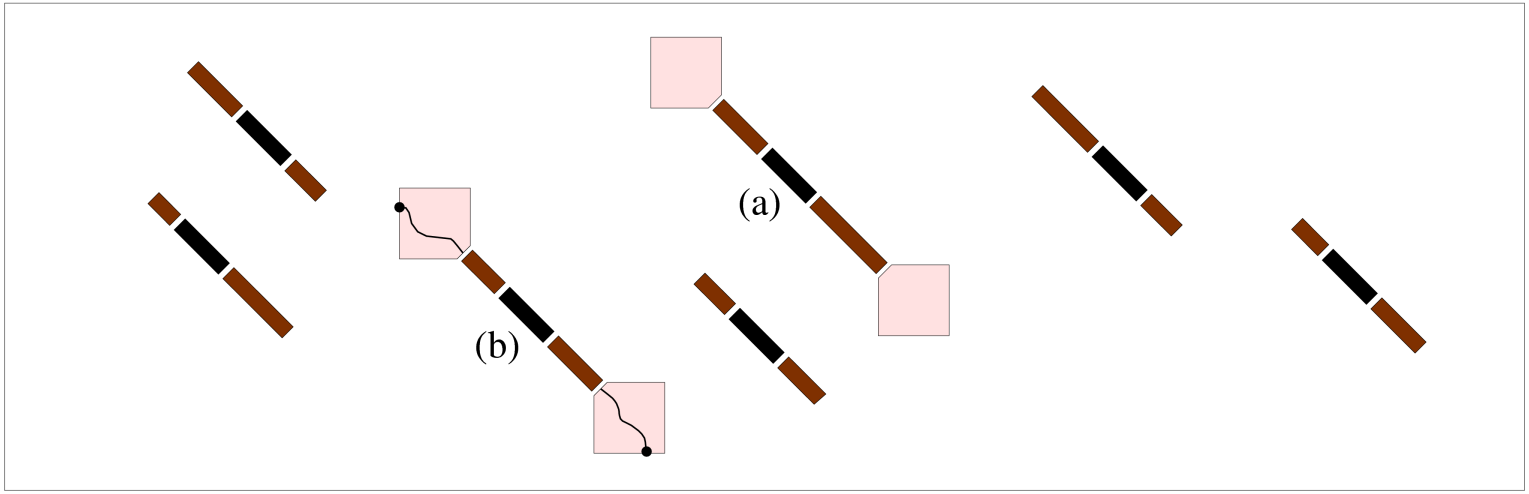

A T C G A C
G T G A T
0 0 0 0 0 0 0
0 0 0 0 4 1 0
0 0 4 1 1 2 0
0 0 1 2 5 2 0
0 4 1 0 2 9 6
0 1 8 5 2 6 7

T C G A

| | |

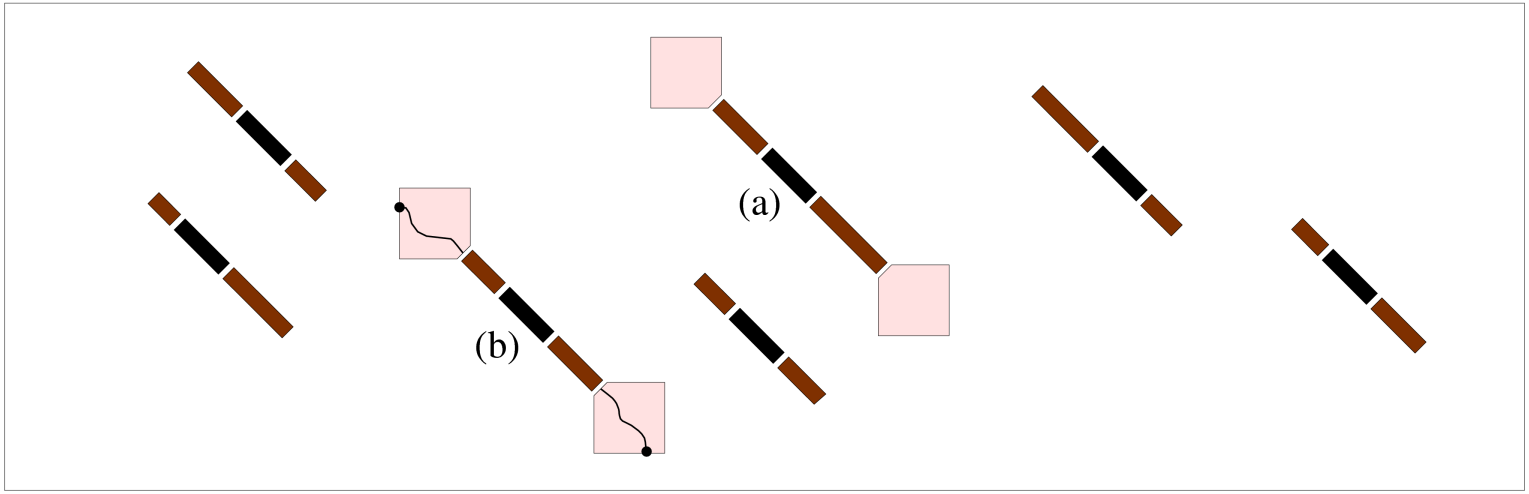
T – G A

Idée de Blast



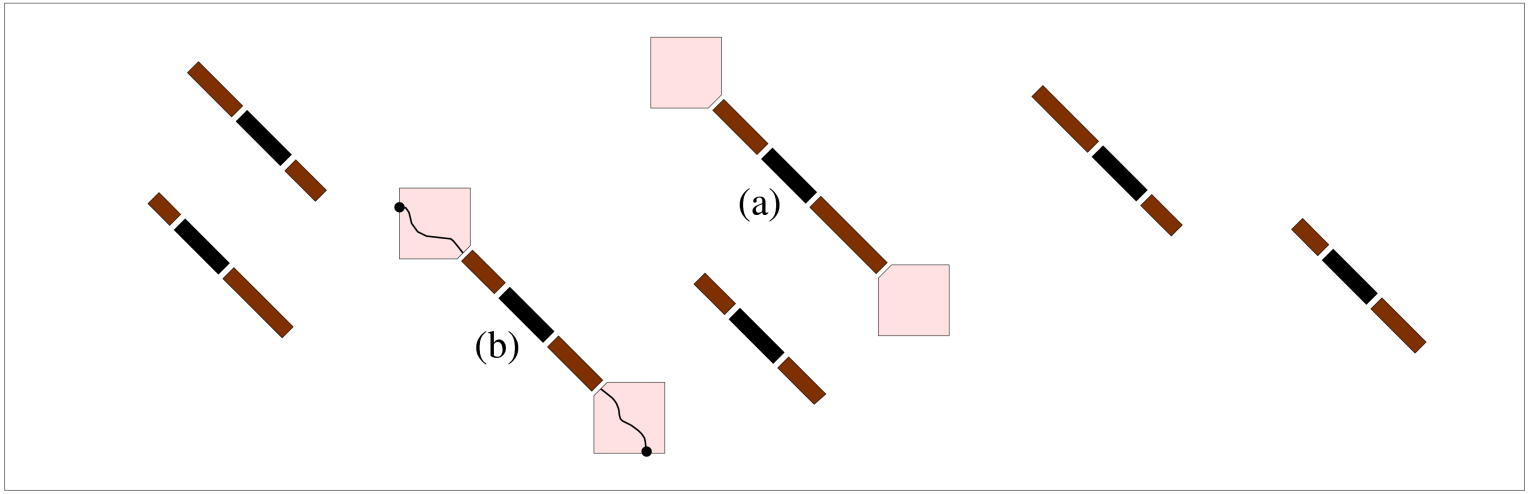
- 1. Localiser des *graines* de taille $w = 11$ (80%)

Idée de Blast



- 1. Localiser des *graines* de taille $w = 11$ (80%)
- 2. Étendre les graines avec une petite tolérance

Idée de Blast



- 1. Localiser des *graines* de taille $w = 11$ (80%)
- 2. Étendre les graines avec une petite tolérance
- 3. Réaliser les calculs de programmation dynamique sur un nombre réduit de positions

Amélioration des graines

Plus le w est petit, plus on détecte d'alignements, mais plus l'étape 2 est surchargée.

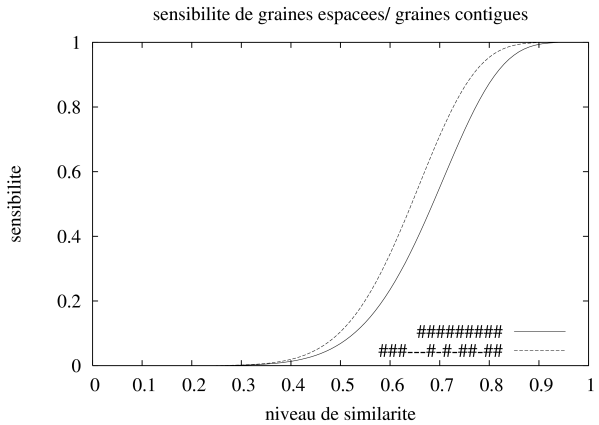
- Graines contigües : ATATTTACGTG ($w = 11$)

Amélioration des graines

Plus le w est petit, plus on détecte d'alignements, mais plus l'étape 2 est surchargée.

- Graines contigües : ATATTTACGTG ($w = 11$)
- Graines espacées : A–TTGC–TT–ATTG
  (#–####–##–####, poids 11, étendue 14)

- Graines vecteurs
- Graines sous-ensemble
- Graines multiples



Conception de graines espacées

Conception de graines sur l'alphabet { −, # }

- aléatoire (Flash, Buhler)

Conception de graines espacées

Conception de graines sur l'alphabet { -, # }

- aléatoire (Flash, Buhler)
- détectant le plus d'alignements (PatternHunter)

###---#-#-##-##

Conception de graines espacées

Conception de graines sur l'alphabet { -, # }

- aléatoire (Flash, Buhler)
- détectant le plus d'alignements (PatternHunter)

  ###---#-#-##-##
- détectant tous les alignements
  avec un nombre maximum de substitutions (Burkhardt et Kärkkäinen)

  ###-#--###-#--###-#

  (poids 12, tous les alignements de taille ≥ 25 avec ≤ 2 substitutions)

Paralléliser Blast...

- recherches balayant des banques entières
  (EMBL : 80 milliards de bases $\longrightarrow$ 20 Go)
- calculs parallélisables
  - grain fin : recherche de graines, extension(s)
  - grain fort : la banque peut être répartie

Paralléliser Blast...

- recherches balayant des banques entières
  (EMBL : 80 milliards de bases $\longrightarrow$ 20 Go)
- calculs parallélisables
  - grain fin : recherche de graines, extension(s)
  - grain fort : la banque peut être répartie

En pratique, la recherche de graine est limitée par la quantité de mémoire disponible et surtout par son accès.

$\longrightarrow$ utilisation de filtres Bloom

$\longrightarrow$ un accès par cycle sur FPGA

Machines proposées

- BLAST, mpiBLAST...
- BioSCAN (1993) (étape 1 ?)

Machines proposées

- BLAST, mpiBLAST...
- BioSCAN (1993) (étape 1 ?)
- FPGA : BEE2 (2000), Mercury (2002), RC-BLAST (2005)
- autres heuristiques : IRISA / Rdisk (2003), DASH (2004)

$\longrightarrow$ Il y a très peu d'architectures accélérant d'autres calculs bioinformatique que la comparaison de séquences.

Indexation des graines

Étape 1 de Blast $\longrightarrow$ il faut déterminer si une graine était présente ou non dans la requête

Comment stocker tous les mots de taille $w = 11$ présents dans une requête (de taille 100 ... 10^9) ?

Représentation d'ensembles

Est-ce qu'un élément appartient à un ensemble donné ?

- dictionnaire des mots correctement orthographiés

Représentation d'ensembles

Est-ce qu'un élément appartient à un ensemble donné ?

- dictionnaire des mots correctement orthographiés
- liste de morceaux disponibles d'un fichier échangé en P2P
- liste des URLs des pages conservées dans un cache web

Représentation d'ensembles

Est-ce qu'un élément appartient à un ensemble donné ?

- dictionnaire des mots correctement orthographiés
- liste de morceaux disponibles d'un fichier échangé en P2P
- liste des URLs des pages conservées dans un cache web
- ensemble des graines dans une requête (w-mots)

Représentation d'ensembles

Dans un univers U, soit un ensemble $S = \{x_1, \ldots, x_n\} \subset U$.
Soit un élément $y \in U$. Appartient-il a S ?

- mémoire de $|U|$ bits (1 bit par élément)

Représentation d'ensembles

Dans un univers U, soit un ensemble $S = \{x_1, \ldots, x_n\} \subset U$.
Soit un élément $y \in U$. Appartient-il a S ?

- mémoire de $|U|$ bits (1 bit par élément)
- mémoire d'environ $n \cdot \log_2 |U|$ bits (liste des éléments)

On n'a qu'une mémoire trop petite, de taille M.
Comment se souvenir de S ?

Représentation d'ensembles avec faux positifs

Peut-on tolérer un faible taux de faux positifs ?

- dictionnaire des mots correctement orthographiés
- liste de morceaux disponibles d'un fichier échangé en P2P
- liste des URLs des pages conservées dans un cache web
- ensemble des graines dans une requête (w-mots)

Représentation d'ensembles avec faux positifs

Peut-on tolérer un faible taux de faux positifs ?

- dictionnaire des mots correctement orthographiés
  $\longrightarrow$ on laisse passer quelques mots incorrects !
- liste de morceaux disponibles d'un fichier échangé en P2P
- liste des URLs des pages conservées dans un cache web
- ensemble des graines dans une requête (w-mots)

Représentation d'ensembles avec faux positifs

Peut-on tolérer un faible taux de faux positifs ?

- dictionnaire des mots correctement orthographiés
  $\longrightarrow$ on laisse passer quelques mots incorrects !
- liste de morceaux disponibles d'un fichier échangé en P2P
  $\longrightarrow$ on demande à tort un morceau de fichier
- liste des URLs des pages conservées dans un cache web
  $\longrightarrow$ on demande à tort une adresse que le cache n'a pas
- ensemble des graines dans une requête (w-mots)

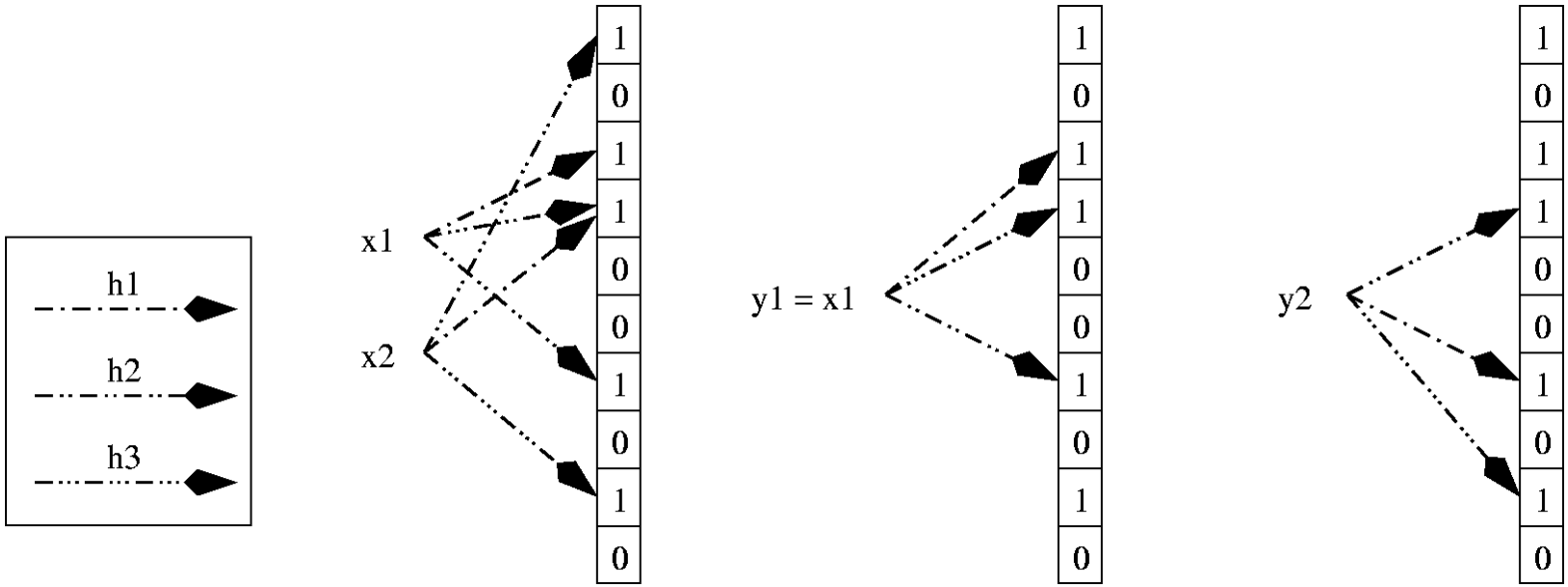
Représentation d'ensembles avec faux positifs

Peut-on tolérer un faible taux de faux positifs ?

- dictionnaire des mots correctement orthographiés
  $\longrightarrow$ on laisse passer quelques mots incorrects !
- liste de morceaux disponibles d'un fichier échangé en P2P
  $\longrightarrow$ on demande à tort un morceau de fichier
- liste des URLs des pages conservées dans un cache web
  $\longrightarrow$ on demande à tort une adresse que le cache n'a pas
- ensemble des graines dans une requête (w-mots)
  $\longrightarrow$ on envoie quelques faux positifs à l'extension

Fonctions de hachage

Fonction $h : U \longrightarrow [1 \ldots M]$ avec $M < |U|$.
La fonction ventile "aléatoirement" les éléments de U.



$S = \{x_1, x_2\}$, $M = 10$, $h(x_1) = 3, h(x_2) = 9$.
$\longrightarrow y_1 = x_1$ vrai positif, $y_2 (\neq x_1)$ faux positif

Fonctions de hachage

Fonction $h : U \longrightarrow [1 \ldots M]$ avec $M < |U|$.
La fonction ventile "aléatoirement" les éléments de U.

Remplissage

On représente $S = \{x_1, \ldots, x_n\}$ par $S' = \{h(x_j) \,|\, x_j \in S\}$
$\longrightarrow$ mise à 1 des positions $h(x_j)$ dans la mémoire

Fonctions de hachage

Fonction $h : U \longrightarrow [1 \ldots M]$ avec $M < |U|$.
La fonction ventile "aléatoirement" les éléments de U.

Remplissage

On représente $S = \{x_1, \ldots, x_n\}$ par $S' = \{h(x_j) \,|\, x_j \in S\}$
$\longrightarrow$ mise à 1 des positions $h(x_j)$ dans la mémoire

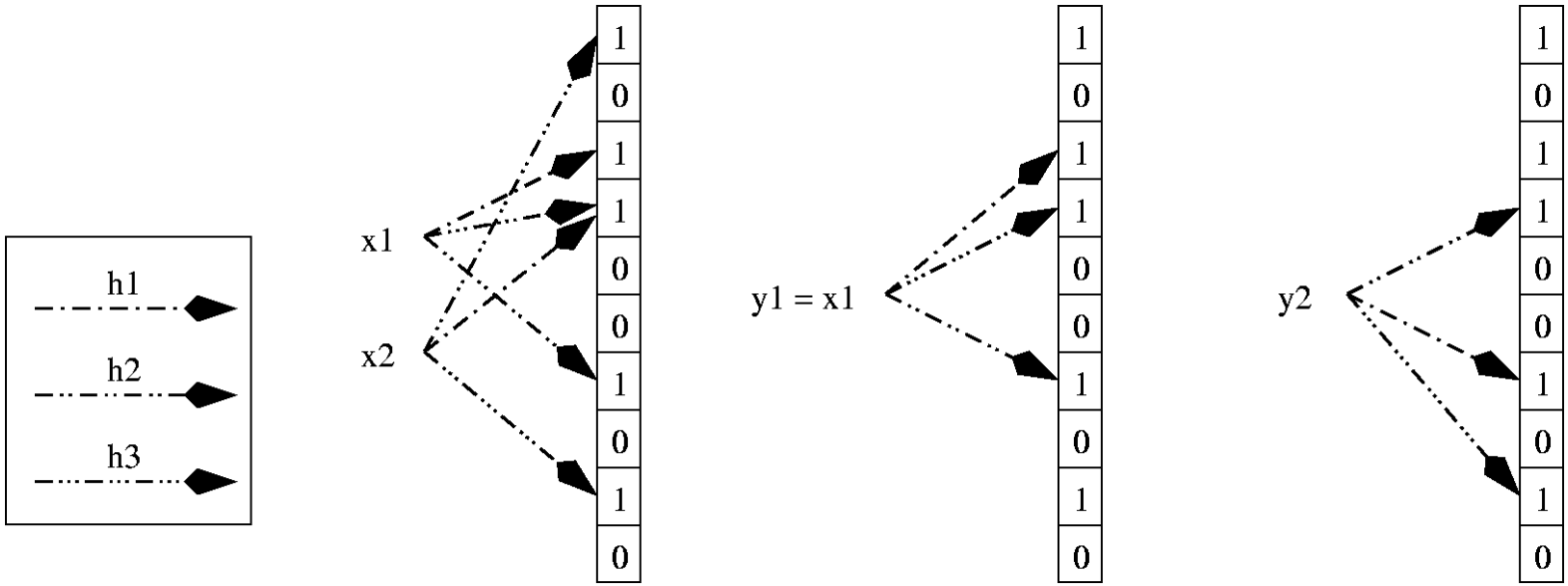
Utilisation

On prédit que $y \in S$ si $h(y) \in S'$
$\longrightarrow$ interrogation de la mémoire à la position $h(y)$

Taux de faux positifs : $(1 - (1 - 1/M)^n) \sim 1 - e^{-n/M}$

Filtres Bloom

Bloom, puis Carter et al. (1970's)



3 fonctions de hachage, $S = \{x_1, x_2\}$, $M = 10$.

$\longrightarrow$ $y_1 = x_1$ vrai positif, $y_2 (\neq x_1)$ faux positif

Filtres Bloom

d fonctions de hachage différentes $h_1, h_2, \ldots h_d$.

Remplissage

On représente $S = \{x_1, \ldots, x_n\}$ par $S' = \{h_i(x_j) \mid x_j \in S\}$
$\longrightarrow$ mise à 1 des positions $h_i(x_j)$ dans la mémoire

Filtres Bloom

d fonctions de hachage différentes $h_1, h_2, \ldots h_d$.

Remplissage

On représente $S = \{x_1, \ldots, x_n\}$ par $S' = \{h_i(x_j) \mid x_j \in S\}$
$\longrightarrow$ mise à 1 des positions $h_i(x_j)$ dans la mémoire

Utilisation

On prédit que $y \in S$ si $\forall i, h_i(y) \in S'$
$\longrightarrow$ interrogation de la mémoire aux positions $h_i(y)$

Taux de faux positifs : $(1 - (1 - 1/M)^{dn})^d \sim (1 - e^{-dn/M})^d$

Avantages des filtres Bloom

- Le taux de faux positifs est inférieur à celui d'une simple fonction de hachage...
- ... il est donc possible de stocker plus de données avec la même mémoire et le même taux d'erreur.
- La mémoire nécessaire est de $n \log_2(1/\varepsilon) \log_2 e$, soit seulement un facteur de $\log_2 e$ au-dessus de l'optimum théorique.
- Meilleur choix du nombre de fonctions de hachage : $d = \ln 2 \cdot (m/n)$.

FPGAs et filtres Bloom

Mémoire

12 BRAM de 512 octets ($= 2^{12} = 4096$ bits) chacun
8 BRAM disponible pour le filtre.

- duplication de mémoire
- accès dual-port [Dharma, 2004], accès partagé ▸ BRAM

FPGAs et filtres Bloom

Mémoire

12 BRAM de 512 octets ($= 2^{12} = 4096$ bits) chacun
8 BRAM disponible pour le filtre.

- duplication de mémoire
- accès dual-port [Dharma, 2004], accès partagé ▸ BRAM

Fonctions de hachage

À base de XOR [Rama 94].

$$h_q(x) = x_1 \cdot q(1) \oplus x_2 \cdot q(2) \oplus x_3 \cdot q(3) \oplus \ldots \oplus x_n \cdot q(n)$$

$\longrightarrow$ une position par cycle (@ 40 MHz) $\longrightarrow$ 40 Mbp/s

Utilisation des filtres Bloom

- Graines de poids w ($2w$ bits) : $|U| = 2^{2w}$ bits
- Mémoire totale sur FPGA
  - $\longrightarrow$ 2^{15} bits accessible à chaque cycle
  - $\longrightarrow$ insuffisant pour des graines de taille ≥ 8

Utilisation des filtres Bloom

- Graines de poids w ($2w$ bits) : $|U| = 2^{2w}$ bits
- Mémoire totale sur FPGA
  - $\longrightarrow 2^{15}$ bits accessible à chaque cycle
  - $\longrightarrow$ insuffisant pour des graines de taille ≥ 8
- hachage simple $\longrightarrow 1/50$ faux positifs

Utilisation des filtres Bloom

- Graines de poids w ($2w$ bits) : $|U| = 2^{2w}$ bits
- Mémoire totale sur FPGA
  $\longrightarrow 2^{15}$ bits accessible à chaque cycle
  $\longrightarrow$ insuffisant pour des graines de taille ≥ 8
- hachage simple $\longrightarrow 1/50$ faux positifs
- 4 fonctions de hachage, mémoire repliquée de 2^{14} bits avec 2 ou 4 accès multiples
  $\longrightarrow$ moins de $1/400$ faux positifs
- La taille des graines n'est impliquée que dans les fonctions de hachage !
- graines de taille 8 – 16
  $\longrightarrow$ le hachage occupe $< 10\%$ du FPGA
  $\longrightarrow$ place pour un étage d'extension

Partie II
Heuristiques à base de graines

Bonus
Les gènes olfactifs du chien

10. Découverte de récepteurs olfactifs (OR)

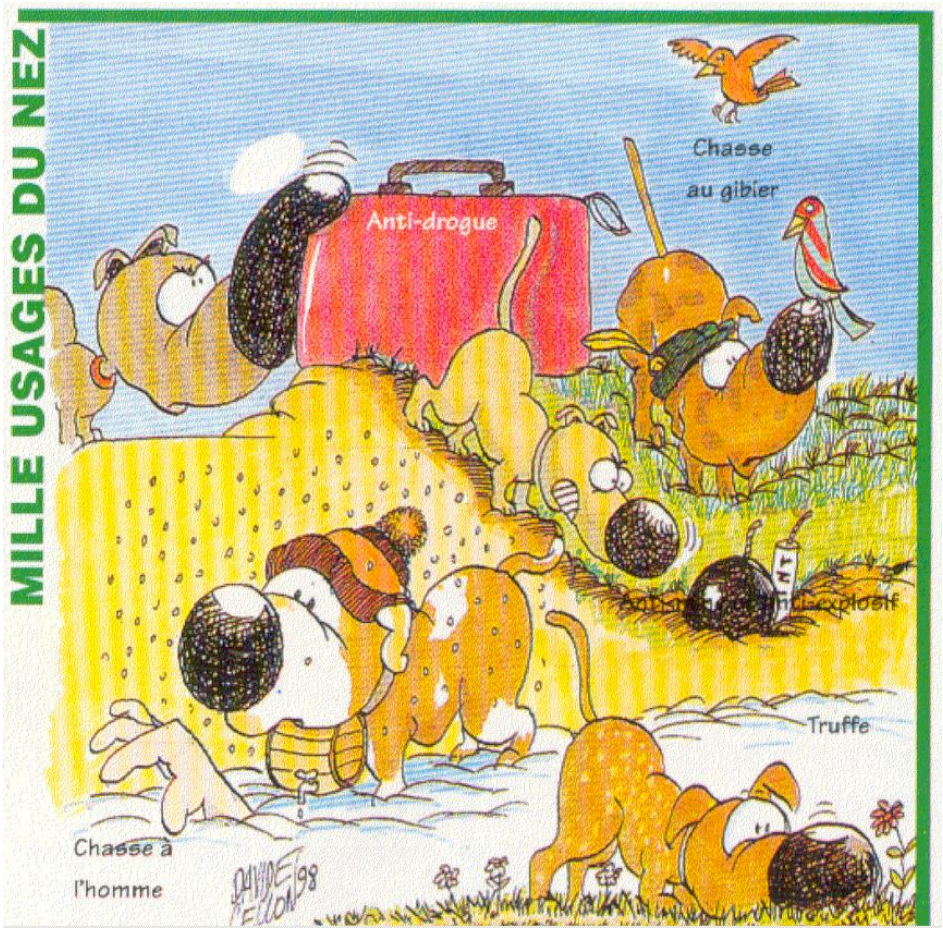
11. Assemblage ciblé

Intérêt du chien pour l'olfaction

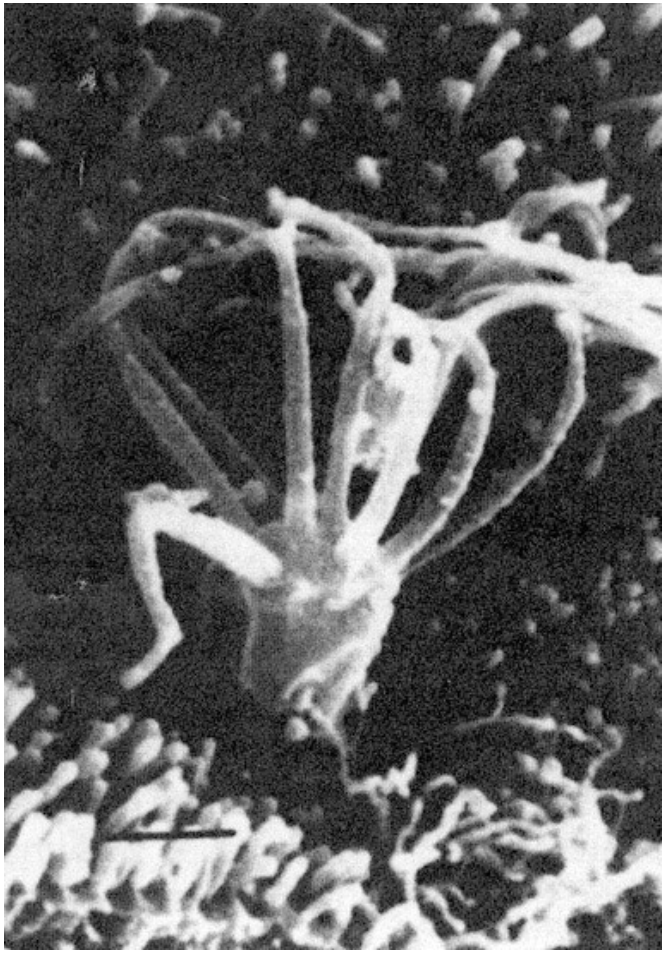
Espèce macrosmate

☞ odorat très développé

- chasse
- police
- truffes

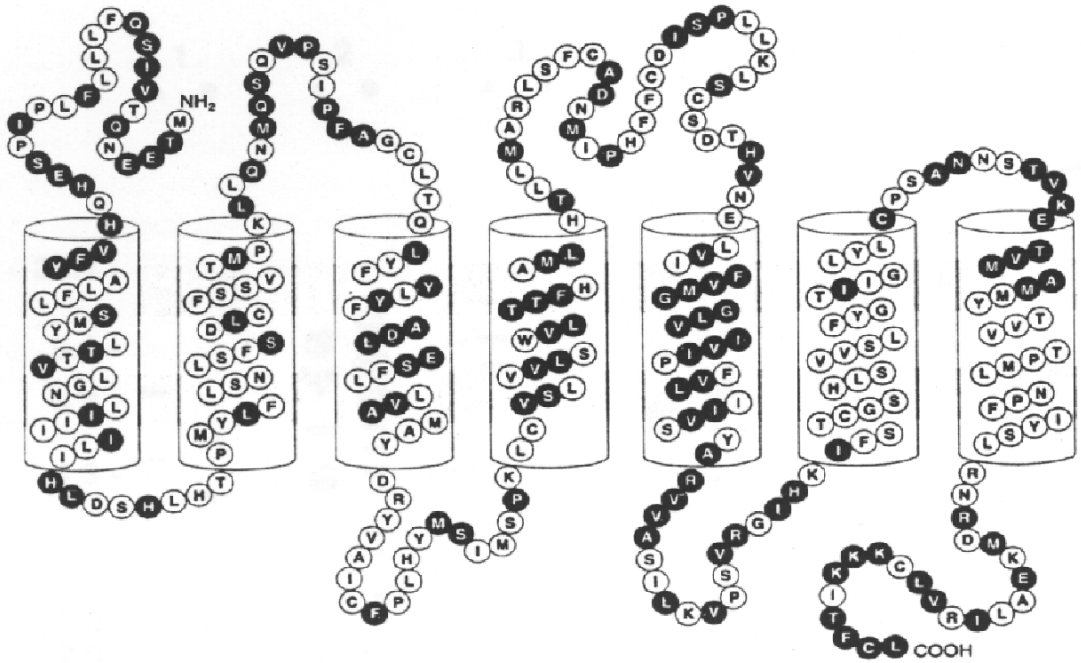


Les récepteurs olfactifs (OR)



- Premières molécules impliquées dans l'olfaction
- Expression dans les spermatozoides
- Expression embryonnaire
  ☞ rôle dans le développement
- Guidance axonale

Les récepteurs olfactifs (OR)



(Buck, Axel, 1991)

Récepteurs à sept domaines transmembranaires
Super-famille des récepteurs couplés aux protéines G

Identification du répertoire complet

- 2003/2004 : nouveau séquençage 7.6× (Broad Institute)
- Analyse de 633 ORs connus avec PRATT
  ☞ découverte de motifs

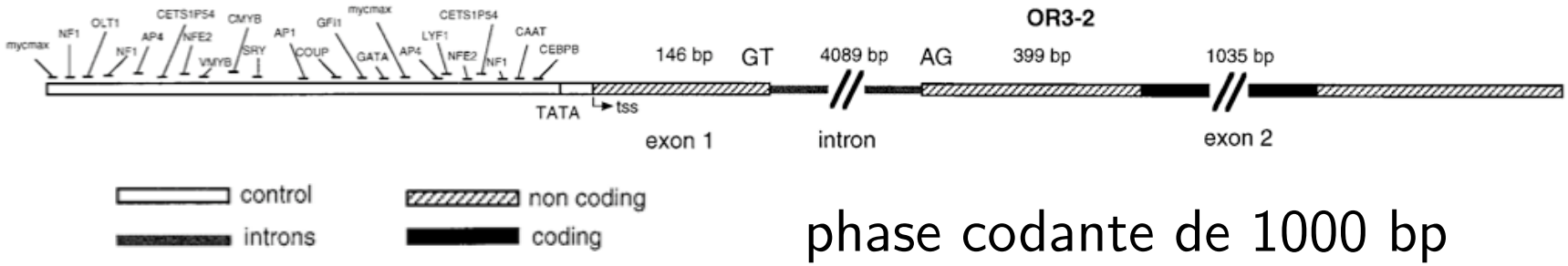
	**II**	P-M-Y-x-[FL]-L-x(2)-[FL]-[AMS]-x(2)-[DE]
	**III**	L-x(1,3)-M-x-[FILY]-D-R-x(2)-A-[IV]-[CS]-x-P-L-x-[HY]-x(3)-[ILM]
	**III**	L-x(3)-M-x(0,1)-Y-x-[FLR]-[LY]-x(2)-[FILV]-[ACS]
	**VI**	K-x-[FL]-[AGHNST]-T-C-x-[AS]-H-x(3)-[AIV]
	**VII**	N-P-[FILMV]-[IV]-Y-[AGST]-[AILMV]-[KR]-x(2)-[DEKQ]

☞ Recherche de *propriétés locales* : 5 motifs différents.

Les gènes OR

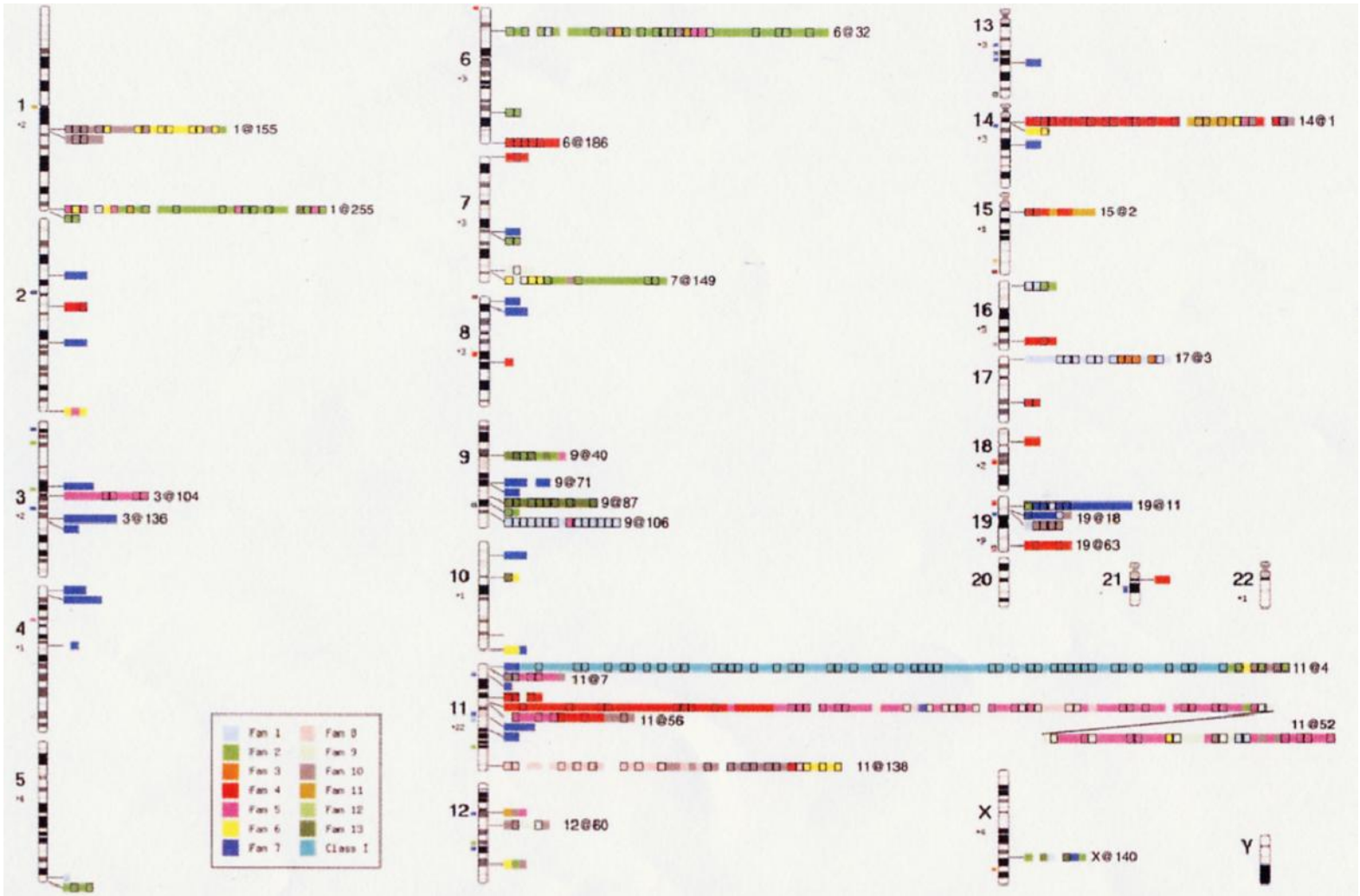
- Découverts en 1991 par Linda Buck et Richard Axel chez le rat (Prix Nobel de Médecine 2004)
- Chez les mammifères : environ 1000 gènes

Homme	900 gènes	plus de 50% de pseudogènes
Souris	1300 – 1500 gènes	20% de pseudogènes
Rat	2000 gènes	31% de pseudogènes
Chien	?	?

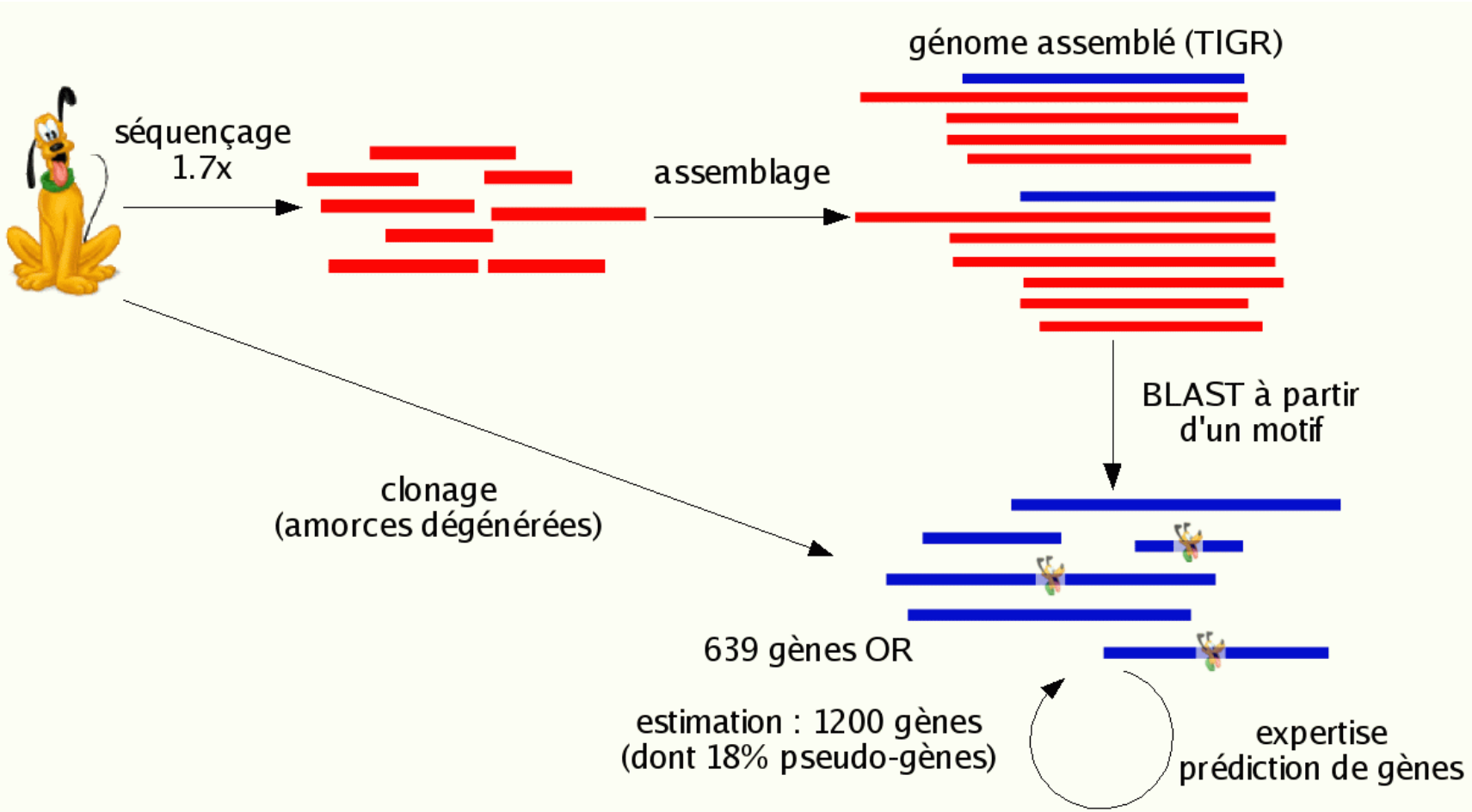


phase codante de 1000 bp

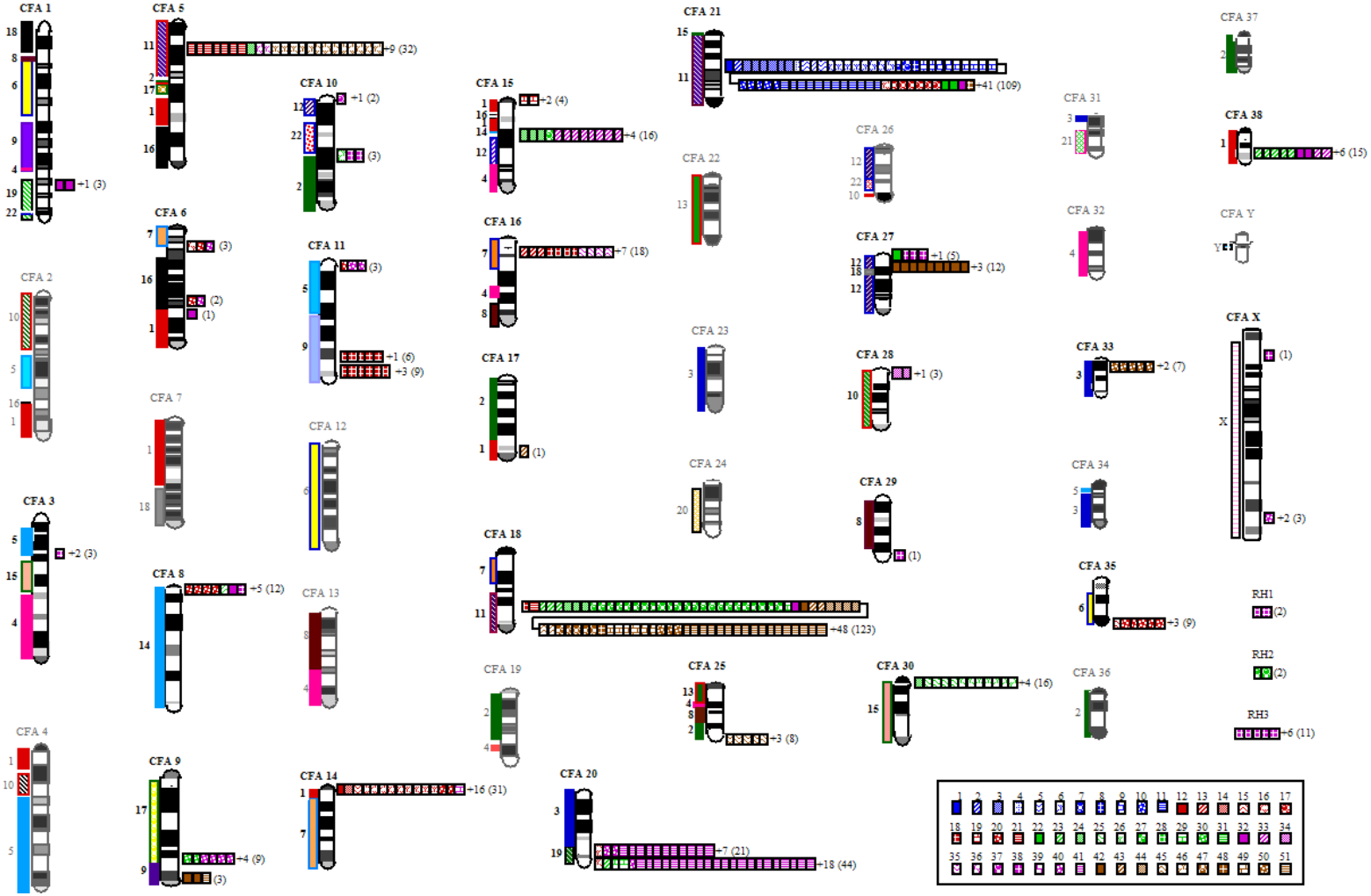
Organisation génomique humaine



Caractérisation de 639 OR canins

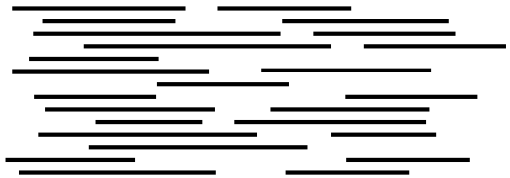


Organisation génomique canine



Attendre l'assemblage ...

36 M traces, 36 Gpb
shotgun 7.6x (Broad Institute, 2003)



assemblage complet

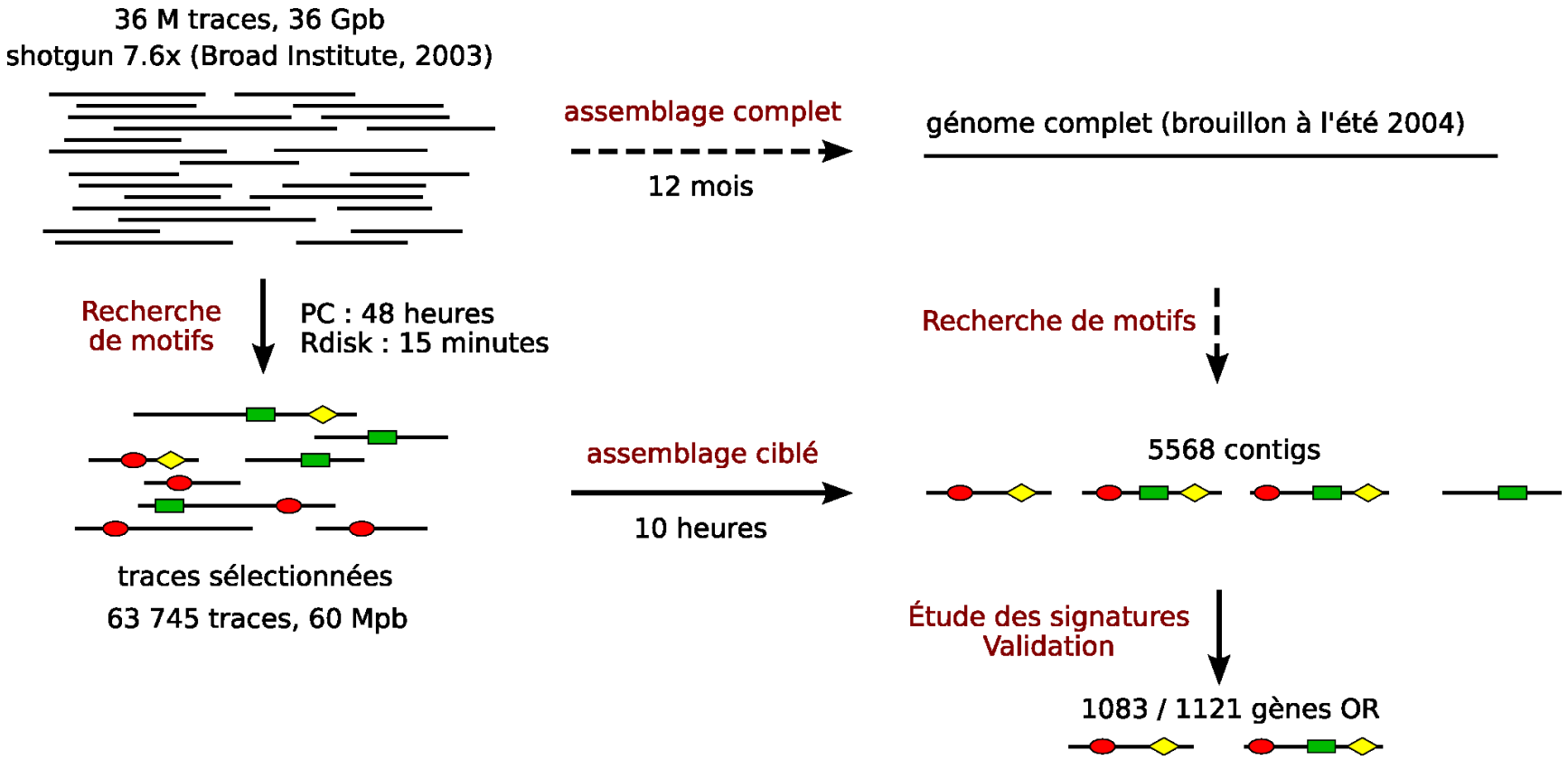
12 mois

génome complet (brouillon à l'été 2004)

Recherche de motifs

Étude des signatures
Validation

... ou travailler directement sur les traces !



Résultats : le répertoire des gènes OR

1121 OR (dont plus de 400 nouveaux)

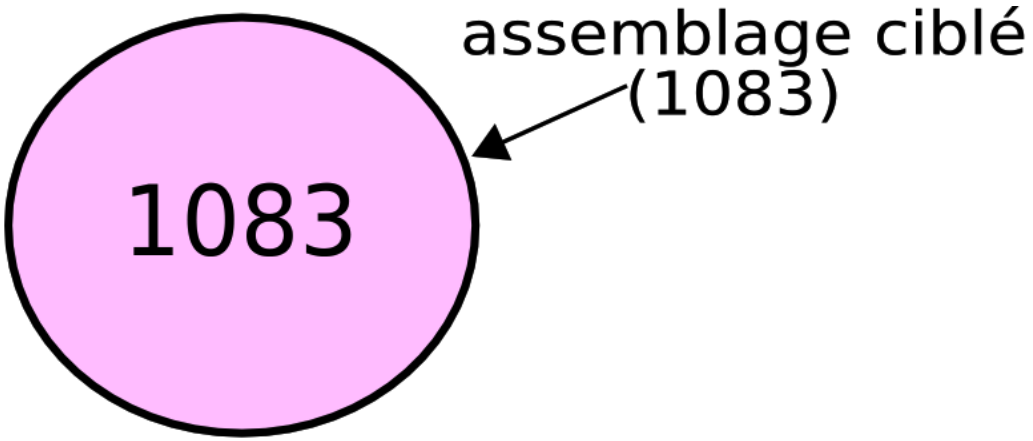
- dont 1036 avec des protéines entières
- 213 pseudogènes (20%)

Juillet 2004 ☞ premier brouillon de l'assemblage 7.6×
(N_{50} = 128 Kb, souris : 22,3 Mb)

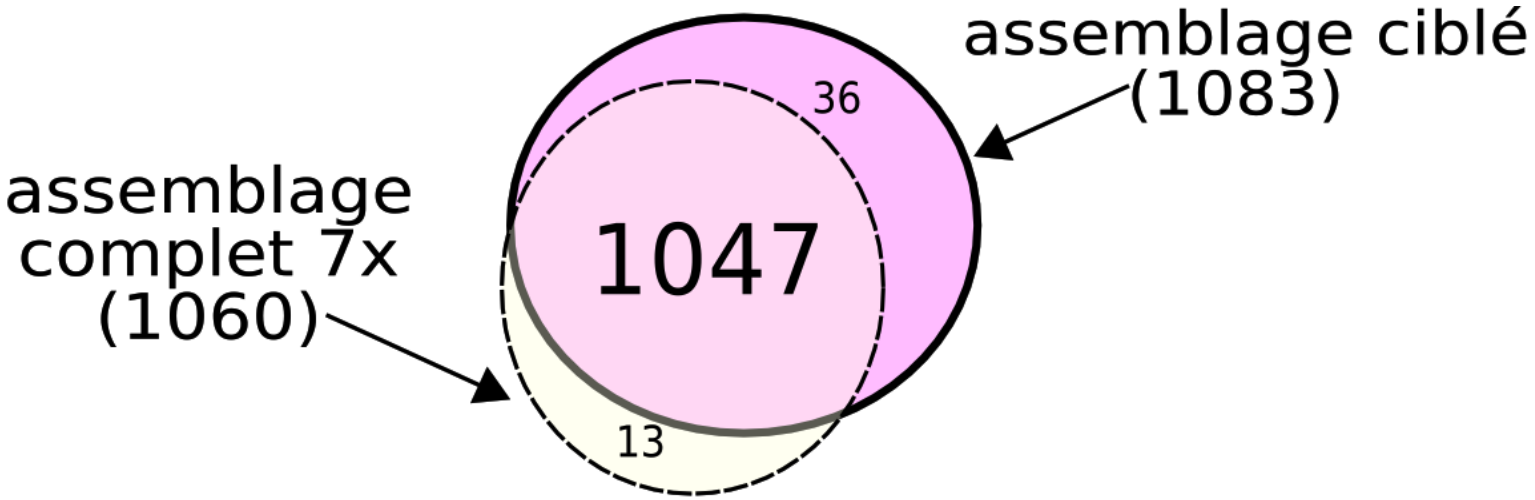
- ☞ comparaison des résultats
- ☞ localisation sur les chromosomes canins

CFA18	215 gènes
CFA20	79 gènes
CFA21	194 gènes
CFAUn	*125 gènes*

Validation des résultats

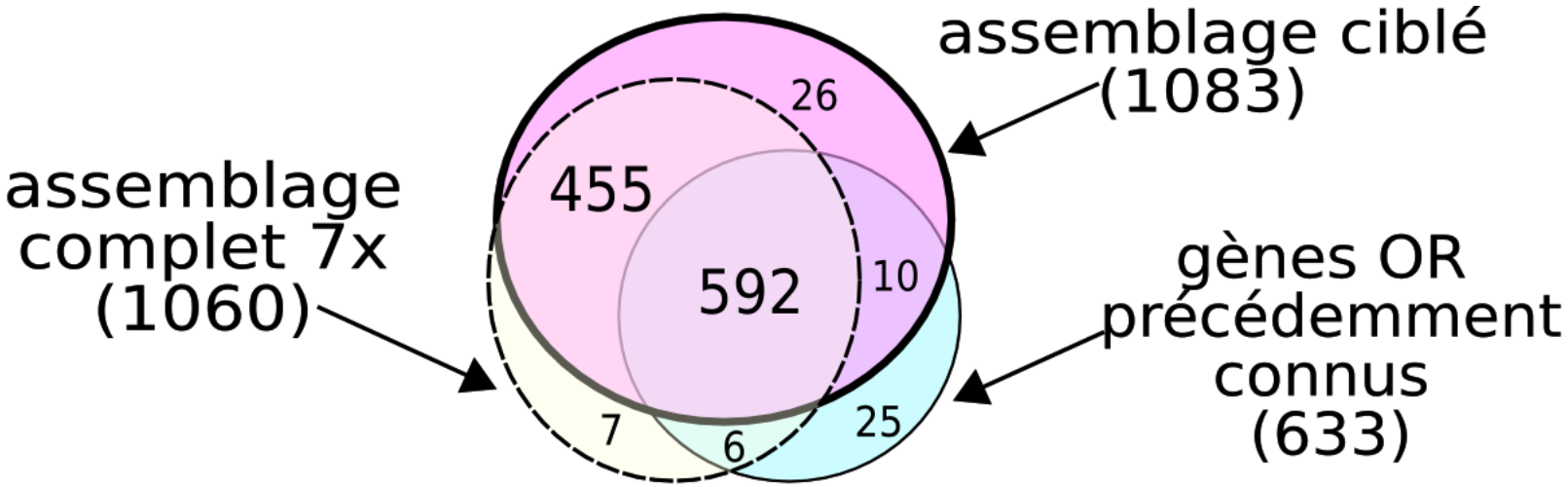


Validation des résultats



☞ L'assemblage ciblé permet efficacement de trouver les gènes sans l'assemblage complet.

Validation des résultats



☞ L'assemblage ciblé permet efficacement de trouver les gènes sans l'assemblage complet.

Collaborer avec des biologistes

- avril 2004 : exposé du problème
- avril – juillet 2004 : découverte des gènes
- et toujours du service : “lancer les BLASTs”
- article “bio” mi-2005

Qui est intéressé par quel résultat ?

- Le biologiste : les 400 nouveaux gènes OR

Qui est intéressé par quel résultat ?

- Le biologiste : les 400 nouveaux gènes OR
- L'informaticien :
  - accélération de la recherche de motifs
  - assemblage

Qui est intéressé par quel résultat ?

- Le biologiste : les 400 nouveaux gènes OR
- L'informaticien :
  - accélération de la recherche de motifs
  - assemblage
- Le bio-informaticien :
  - méthode, quelles applications ?
  - choix et pertinence des motifs

Conclusion

Architectures pour la génomique

- Programmation dynamique : architectures systoliques
  ASIC (1985), FPGA (1990, 2000 –)
  $\longrightarrow$ OK, encodage modulo : réducteur ?)
- Heuristiques à base de graines
  compromis sensibilité / vitesse de calcul
  $\longrightarrow$ souvent réduit à Blast
- Il n'y a pas que la comparaison de séquences !
  $\longrightarrow$ modèles, motifs, automates, grammaires...

Conclusion

Architectures pour la génomique

- Programmation dynamique : architectures systoliques
  ASIC (1985), FPGA (1990, 2000 –)
  $\longrightarrow$ OK, encodage modulo : réducteur ?)
- Heuristiques à base de graines
  compromis sensibilité / vitesse de calcul
  $\longrightarrow$ souvent réduit à Blast
- Il n'y a pas que la comparaison de séquences !
  $\longrightarrow$ modèles, motifs, automates, grammaires...

- Architecture : complexités en espace, chemin critique
- Besoin de solutions théoriques aux problèmes en architecture (ε-transitions, filtres Bloom...)
- Temps de conception et de déploiement ?

Publicité

- vendredi 2 décembre, 10h30 : soutenance de thèse
- puis 3-4 mois de visite
  (LBIT, Montréal, Nadia El-Mabrouk, ARN ?)
- après ?
  - Symbiose ?
  - Lille, équipe Bioinfo (H. Touzet / G. Kucherov) ?

Merci !

Résumé

La génomique génère une masse considérable de données. La banque de séquences nucléiques GenBank double de volume tous les 14 mois, et près d'un millier de génomes sont en cours de séquencage. Le traitement de cette masse d'information est un défi que les concepteurs d'architectures spécialisées ont relevé depuis plus d'une dizaine d'années. Il existe maintenant sur le marché des accélérateurs dédiés au calcul génomique.
L'application favorite de ces architectures est le calcul de similarités entre plusieurs séquences. Ce calcul est souvent accéléré par des architectures systoliques. D'autres approches utilisent les heuristiques à base de graines comme celles utilisées dans les programmes FASTA et BLAST.
L'exposé présentera le principe de ces architectures, leurs performances et les pistes de recherche actuelles dans ce domaine.