

Modèle de Markov Caché (HMM hidden Markov Model)



En biologie on recherche souvent à mettre le bon label sur chaque résidu d'une séquence.

Par exemple :

- définir si un résidu appartient à un exon, un intron ou à une région intergénique.
- déterminer si une nouvelle séquence protéique appartient à une famille de protéines donnée
- etc...

Les modèles de Markov cachés (HMM) permettent de réaliser des modèles probabilistes d'une suite de problèmes linéaires labellisés.

Ils sont utilisés pour :

- déterminer la structure en gènes d'un fragment génomique
- réaliser des alignements multiples
- déterminer des profils
- identifier des sites de régulations
- etc...

Modèle de Markov Caché (HMM hidden Markov Model)

Première étape : modéliser le problème en terme d'états



Exemple simple : dans un casino, ils utilisent la plupart du temps un dé normal, mais occasionnellement aussi un dé pipé. Le dé pipé a une probabilité de 0.5 pour le 6 et de 0.1 pour les autres chiffres.

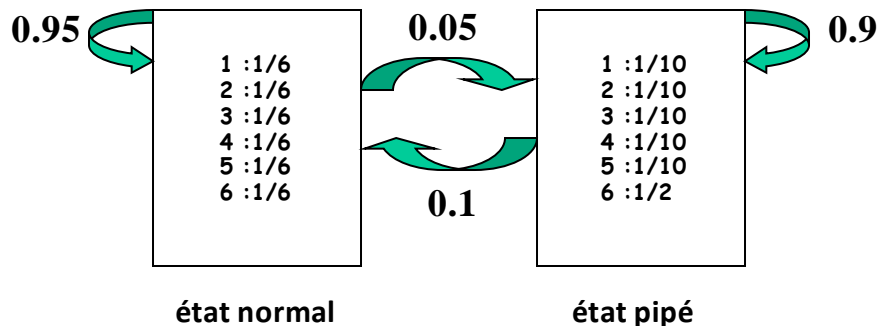
La probabilité de changer du dé normal au dé pipé avant chaque jet est de 0.05, et celle de passer du dé pipé au dé normal est de 0.1.

Le changement de dé suit donc un processus de Markov.

Dans chaque état du processus, le résultat d'un jet de dé est associé à des probabilités différentes.

L'ensemble du processus décrit peut être modélisé par un HMM :

On a deux états : dé normal, dé pipé



Qu'est ce qui est caché :

Observation : résultat du jet de dé
On ne sait pas quel dé est utilisé

L'état de la séquence d'observation
est caché

Modèle de Markov Caché (HMM hidden Markov Model)

Le modèle est décrit par deux ensembles de probabilités :

- probabilités de passer d'un état à l'autre : probabilités de transition
- probabilités d'observer un symbole pour un état donné : probabilités d'émission

A ceci s'ajoute le choix de l'état initial.

Un HMM est donc défini par :

- Un vecteur de probabilités initiales $\Pi = (\pi_i)$
- un vecteur de probabilités de transition
(probabilité de passer de l'état i à l'état j) $A = (a_{ij})$
- une matrice de probabilités d'émission
(probabilité que le symbole b soit observé dans l'état i) $E = (e_i(b))$

La probabilité d'une séquence d'observation x et d'une séquence d'état (chemin) π est donnée par :

$$P(x, \pi) = a_{0\pi_1} \prod_{i=1}^L e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}$$

Problème : en général on ne connaît pas π . On cherche à l'estimer.

Modèle de Markov Caché (HMM hidden Markov Model)

On parle de probabilités d'émission car souvent on pense aux HMMs comme étant des modèles qui génèrent ou émettent des séquences.

Par exemple, on peut générer à partir du HMM précédent du casino une séquence aléatoire de résultats de jets de dés en simulant les choix successifs des dés (normal ou pipé) puis la face du dé choisi.

De façon général, on peut générer une séquence x à partir d'un HMM comme suit :

- Premièrement, un état π_1 est choisi à partir de la probabilité a_{0i}
- Dans cet état π_1 , une observation est émise en fonction de la distribution e_{π_1} (distribution des probabilités d'émission associée à π_1)
- Un nouvel état π_2 est choisi en fonction des probabilités de transition $a_{\pi_1 i}$ (choix d'une transition sortante)
- etc

$P(x)$ est la probabilité que x ait été générée par le modèle

Les HMMs définissent donc un processus stochastique :

- non déterministe (une même séquence peut être générées de plusieurs manières différentes)
- markovien : le chemin (la séquence des états) constitue une chaîne de Markov simple puisque la probabilité de transition vers un état ne dépend que de l'état actuel et non des états rencontrés précédemment
- caché : on observe la séquence générée par le modèle mais pas la séquence des états qui génèrent ces observations

Modèle de Markov Caché (HMM hidden Markov Model)



En général on a une séquence d'observation (résultats d'une suite de jets de dé, une séquence d'ADN) et on est intéressé par la suite d'états sous-jacents (dé normal ou pipé, exon, intron ou séquence intergénique).

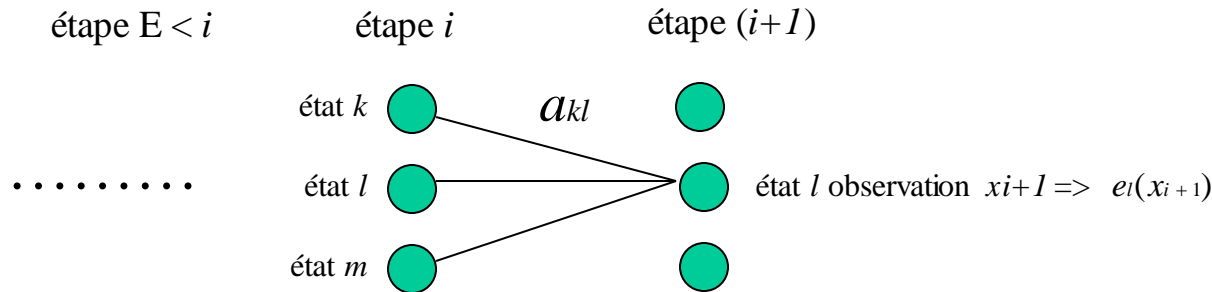
Trouver ce que la séquence d'observation signifie en considérant les états sous-jacents est appelé décodage (decoding).

Plusieurs méthodes existent pour trouver le chemin le plus probables des états cachés, mais la plus commune est l'algorithme de Viterbi.

L'algorithme de Viterbi

Algorithme de programmation dynamique permettant de trouver le chemin le plus probable d'états cachés qui correspond à la séquence d'observation (Viterbi path).

Le chemin le plus probable π^* va être déterminé de façon récursive.



Calcul du chemin le plus probable pour atteindre l'état l à l'étape $i+1$ sachant l'observation x_{i+1} . Ce chemin doit passer par un des trois états à l'étape i .

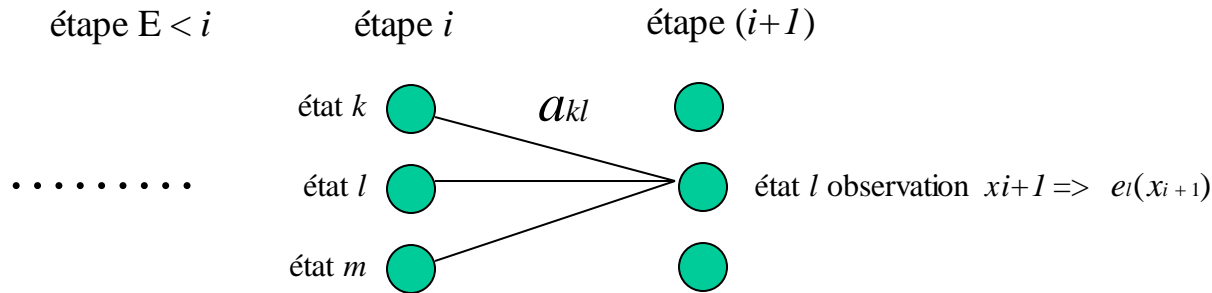
Par exemple, le chemin le plus probable finissant par kl (état k à l'étape i et l à l'étape $i+1$) va correspondre au chemin le plus probable pour atteindre l'état k à l'étape i sachant l'observation x_i suivi par l'état l . La probabilité de ce chemin sera donnée par:

$\Pr(\text{chemin le plus probable pour atteindre } k) \cdot \Pr(l|k) \cdot \Pr(\text{observer } x_{i+1} \text{ à l'état } l)$

Donc la probabilité du chemin pour atteindre l'état l est donnée par:

$\Pr(l \text{ à l'étape } i+1) = \max_{c=k,l,m} \Pr(c \text{ à l'étape } i) \cdot \Pr(l|c) \cdot \Pr(\text{observer } x_{i+1} \text{ à l'état } l)$

L'algorithme de Viterbi



$\Pr(l \text{ à l'étape } i+1) = \max_{c=k,l,m} \Pr(c \text{ à étape } i) \cdot \Pr(l|c) \cdot \Pr(\text{observer } x_{i+1} \text{ à l'état } l)$

$\Pr(l|c)$ correspond à la probabilité de transition a_{cl}

$\Pr(\text{observer } x_{i+1} \text{ à l'état } l)$ correspond à la probabilité d'émission $e_l(x_{i+1})$

Soit $v_k(i)$ la probabilité du chemin le plus probable se terminant à l'état k pour une observation x_i .

La probabilité du chemin le plus probable pour l'observation x_{i+1} est donnée par:

$$v_l(i+1) = e_l(x_{i+1}) \max(v_k(i) a_{kl})$$

Pour $i = 1$, c'est à dire la première étape, le chemin le plus probable pour atteindre cet état n'existe pas vraiment. On va utiliser la probabilité initiale d'être dans un état l et la probabilité d'émission de x_1 dans cet état.

$$v_l(1) = a_{0l} e_l(x_1)$$

L'algorithme de Viterbi

Comme on calcule à chaque étape, le chemin le plus probable en fonction seulement de l'étape précédente et que l'on veut connaître, étant donné une séquence d'observation, le chemin le plus probable dans l'ensemble du HMM, il va falloir trouver un moyen de garder en mémoire les chemins partiels calculés.

Ceci va être réalisé en gardant pour chaque état un back pointer qui va pointer sur le prédécesseur qui provoque l'état courant, soit :

$$ptr_{i+1}(l) = \arg \max_k (v_k(i) a_{kl})$$

La fonction argmax (argument of the maximum) est la valeur de l'argument pour laquelle une expression donnée atteint sa valeur maximum. Par exemple la valeur maximum de la fonction $x(10-x)$ est 25, ce qui arrive quand x vaut 5. On aura donc :

$$x \in \mathbf{R} \quad \arg \max x(10-x) = 5$$

Dans notre cas, l'opérateur argmax va sélectionner l'index k qui maximise l'expression entre parenthèse.

L'algorithme va donc dans un premier temps examiner toute la séquence avant de déterminer l'état final le plus probable, puis par backtracking au travers des pointers ptr , indiquer comment on est arrivé à cet état final le plus probable.

L'algorithme de Viterbi

Dans un HMM, on peut introduire deux états Begin et End qui modélisent le début et la fin du HMM. Ces sites n'émettent pas de symbole. Dans ce cas, l'algorithme de Viterbi est:

Initialisation ($i = 0$) $v_0(0) = 1, v_k(0) = 0$ pour $k > 0$

Récurtivité ($i = 1$ à L) $v_l(i) = e_l(x_i) \max_k (v_k(i-1)a_{kl})$
 $ptr_i(l) = \arg \max_k (v_k(i-1)a_{kl})$

Terminaison $P(x, \pi^*) = \max_k (v_k(L)a_{k0})$
 $\pi^*_L = \arg \max_k (v_k(L)a_{k0})$

Traceback ($i = L$ à 1) $\pi^*_{i-1} = ptr_i(\pi^*_i)$

Si l'état End n'est pas modélisé,
 a_{k0} disparaît

extrait de Durbin et al. (1998), *Biological sequence analysis*, Cambridge University Press)

Le problème pratique le plus ennuyeux avec cet algorithme est que la multiplication de plusieurs probabilités conduit toujours à de très petites valeurs qui conduisent à des effets de bord sur les ordinateurs. C'est pourquoi l'algorithme doit être appliqué dans l'espace des logs, *i.e.*, doit calculer le $\log(v_l(i))$, ce qui conduit les produits à devenir des sommes et les nombres gardent alors des valeurs raisonnables.

Le Forward algorithme

L'algorithme de Viterbi permet de trouver, étant donné une séquence d'observation, le chemin le plus probable dans un HMM (la succession la plus probable d'états) permettant d'expliquer cette séquence d'observation.

Une autre question peut être : Quelle est la probabilité de cette séquence d'observation?

Comme plusieurs chemins différents peuvent conduire à une même séquence d'observation x , nous devons additionner les probabilités de tous les chemins possibles pour obtenir la probabilité de x .

$$P(x) = \sum_{\pi} P(x, \pi)$$

Or, le nombre de chemin possible augmente de façon exponentielle avec la longueur de la séquence, donc calculer cette probabilité en énumérant tous les chemins n'est pas praticable. Cette probabilité peut être aussi calculée par un algorithme de programmation dynamique comme dans le cas de l'algorithme de Viterbi. On remplacera simplement l'étape de maximisation par la somme des probabilités de tous les chemins possibles. Cet algorithme s'appelle le forward algorithme.

La quantité correspondant à la variable $v_k(i)$ dans l'algorithme de Viterbi sera remplacée par :

$$f_k(i) = P(x_1 \dots x_i, \pi_i = k)$$

Elle correspond à la probabilité de la séquence d'observation jusqu'à x_i inclus, avec comme contrainte que le chemin π à l'étape i corresponde à l'état k .

D'où l'équation de récursivité :

$$f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) a_{kl}$$

Le Forward algorithme

Algorithme

Initialisation ($i = 0$) $f_0(0) = 1, f_k(0) = 0$ pour $k > 0$

Récurtivité ($i = 1$ à L) $f_i(i) = e_i(x_i) \sum_k f_k(i-1) a_{ki}$

Terminaison $P(x) = \sum_k f_k(L) a_{k0}$

De même que pour l'algorithme de Viterbi, il est conseillé de travailler dans l'espace des logs.

extrait de Durbin et al. (1998), Biological sequence analysis, Cambridge University Press)

Le Backward algorithm

(algorithme progressif/rétrograde ou rétrogressif)

On peut être intéressé par connaître la probabilité que l'observation x_i provienne de l'état k étant donné la séquence d'observation x , *i.e.*,

$$P(\pi_i = k | x)$$

Ceci est la probabilité à posteriori de l'état k au temps i (à l'étape i) quand la séquence émise est connue.

On calcule d'abord la probabilité de la séquence d'observation entière avec le $i^{\text{ème}}$ symbole produit par l'état k :

$$\begin{aligned} P(x, \pi_i = k) &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_L | x_1 \dots x_i, \pi_i = k) \\ &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_L | \pi_i = k) \end{aligned}$$

Le premier terme correspond à $fk(i)$ calculé par le forward algorithm (probabilité de la séquence d'observation se terminant à l'état k avec comme symbole x_i , calcul progressif des probabilités).

Le second terme exprime le fait que ce qui suit k dépend uniquement de l'état k . Ce terme est appelé $bk(i)$ (calcul rétrogressif des probabilités qui représente la probabilité d'obtenir les autres observations ultérieures à un état initial ici k)

Il est calculé de façon analogue à ce qui a été décrit pour le forward algorithm mais avec une itération récursive commençant par la fin de la séquence.

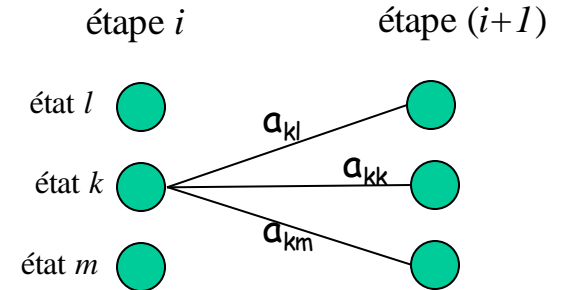
Le Backward algorithm

Algorithme

Initialisation ($i = L$) $b_k(L) = a_{k0}$ pour tout k

Récurtivité ($i = L-1$ à 1) $b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$

Terminaison $P(x) = \sum_l a_{0l} e_l(x_1) b_l(1)$



The Backward Algorithm

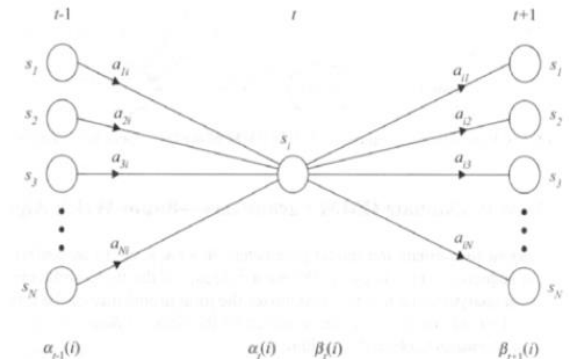
La terminaison est rarement nécessaire car $P(x)$ est généralement calculée par le forward algorithm.

On peut donc écrire l'équation

$$P(x, \pi_i = k) = P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_L | x_1 \dots x_i, \pi_i = k)$$

On obtient la probabilité *a posteriori* (probabilité d'être dans l'état à l'étape i):

$$P(\pi_i = k | x) = \frac{f_k(i) b_k(i)}{P(x)}$$



Package R : « HMM »

```
>hmm = initHMM(c("s","p"), c("ma","co","me"), startProbs=c(.4,0.6),  
transProbs = matrix(c(0.6,0.3,0.4,0.7),2), emissionProbs  
=matrix(c(0.6,0.1,0.3,0.4,0.1,0.5),3))
```

```
> print(hmm)
```

```
$States
```

```
[1] "s" "p"
```

```
$Symbols
```

```
[1] "ma" "co" "me"
```

```
$startProbs
```

s	p
0.4	0.6

```
$transProbs
```

```
to
```

	s	p
s	0.6	0.4
p	0.3	0.7

```
From
```

```
$emissionProbs
```

```
symbols
```

```
states
```

	ma	co	me
s	0.6	0.3	0.1
p	0.1	0.4	0.5

Package R : « HMM »

```
> observations = c("ma", "co", "me")  
> print(observations)  
[1] "ma" "co" "me"  
> viterbi = viterbi(hmm, observations)  
> print(viterbi) [1] "s" "p" "p"  
> forward = forward(hmm, observations)
```

symbols
states

	1	2	3
s	0.24	0.0486	0.004572
p	0.06	0.0552	0.029040

```
> backward = backward(hmm, observations)  
> print (exp(backward))
```

index
states

	1	2	3
s	0.1076	0.26	1
p	0.1298	0.38	1

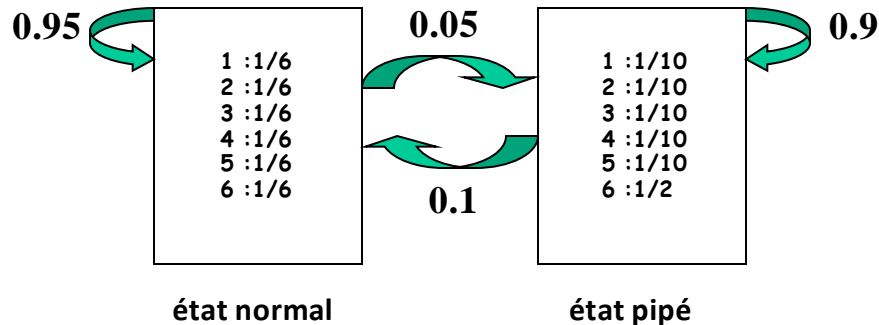
```
> posterior = posterior(hmm, observations)  
> print (posterior)
```

index
states

	1	2	3
s	0.768297	0.359372	0.1360228
p	0.231703	0.6240628	0.8639772

Estimation des paramètres d'un HMM

Jusqu'à présent nous avons travaillé avec des HMM complètement décrits (on connaissait toutes les probabilités (comme exemple du casino)).



Cependant quand on établit un HMM, une fois la topologie du modèle construite, il va falloir estimer les paramètres du modèle, à savoir, les probabilités initiales, les probabilités de transition et les probabilités d'émission.

Le plus souvent nous disposons d'un ensemble d'exemples pour lesquels nous connaissons le chemin dans le HMM. Des algorithmes existent pour estimer ces paramètres si les chemins sont inconnus, notamment l'algorithme de Baum-Welch.

Cas où les chemins des exemples sont connus :

On peut calculer le nombre de fois où chaque transition particulière ou chaque émission particulière de symbole est rencontrée dans l'ensemble d'apprentissage.

Soit :

A_{kl} le nombre de fois où nous observons une transition de l'état k vers l'état l dans l'ensemble d'apprentissage

$E_k(b)$ le nombre de fois où le symbole b est émis par l'état k

a_{kl} et $e_k(b)$, les probabilités de transition et d'émission sont données par estimation des maximums de vraisemblance :

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \quad \text{et} \quad e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

$\sum_{l'} A_{kl'} \rightarrow$ Nombre total de transitions de k
 $\sum_{b'} E_k(b') \rightarrow$ Nombre total d'observations à l'état k

Problème quand un état k n'est jamais observé dans l'ensemble d'apprentissage car les estimations des probabilités sont alors indéfinies (numérateur et dénominateur égaux à 0). Pour éviter ce problème, on ajoute à A_{kl} et $E_k(b)$ des pseudocounts prédéterminés. On a alors :

$$a_{kl} = \frac{A_{kl} + r_{kl}}{\sum_{l'} A_{kl'} + r_{kl'}} \quad \text{et} \quad e_k(b) = \frac{E_k(b) + r_k(b)}{\sum_{b'} E_k(b') + r_k(b')}$$

Les pseudocounts doivent refléter les connaissances que l'on a a priori sur les biais des valeurs des probabilités

Cas où les chemins sont inconnus : L'apprentissage par l'algorithme de Baum-Welch.

Trouver le modèle μ qui maximise la probabilité d'une séquence d'observation $x = (x_1, x_2, \dots, x_n)$.

Impossible de trouver un modèle de façon analytique. Utilisation d'un algorithme itératif qui permet d'estimer les paramètres du modèle qui maximisent la probabilité d'une séquence d'observation.

Utilisation d'une procédure par itération : méthode couramment utilisé l'algorithme de Baum-Welch (utilisation de l'algorithme forward-backward) :

Cet algorithme calcule les A_{kl} et les $E_k(b)$ comme le nombre attendu de fois où chaque transition et chaque émission sont utilisées étant donné les séquences d'apprentissage. Il utilise les mêmes valeurs forward and backward que celles calculées lors de la méthode de calcul des probabilités à *posteriori*.

Soit une séquence d'observation $x = x_1 \dots x_T$, notre but est de trouver les paramètres $\theta = \langle A, E \rangle$ qui maximise la probabilité $P(x | \theta)$ de générer x avec le modèle.

Soit une séquence d'observation $x = x_1 \dots x_T$, notre but est de trouver les paramètres $\theta = \langle A, E \rangle$ qui maximise la probabilité $P(x | \theta)$ de générer x avec le modèle.

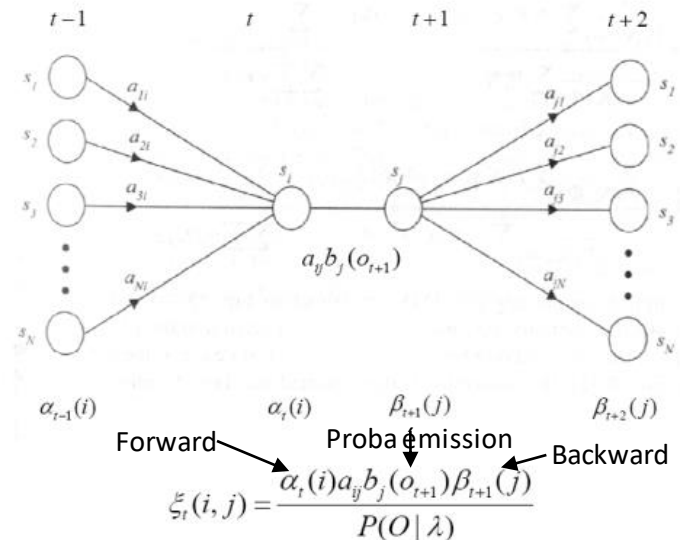
La probabilité que a_{kl} soit utilisée à la position i de la séquence d'observation x est donnée par :

- la probabilité que le modèle a émis les symboles (x_1, x_2, \dots, x_i) et se trouve dans l'état k à la position i . Peut être obtenu en utilisant l'algorithme forward.
- la probabilité d'émettre la séquence restante si on est à l'état l à la position $i+1$. Peut être obtenu en utilisant l'algorithme backward.
- la probabilité de passer de l'état k (position i) à l'état l (position $i+1$) (probabilité de transition a_{kl}) en émettant le caractère x_{i+1} .

$$P(\pi_i = k, \pi_{i+1} = l | x, \theta) = \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x)}$$

θ : paramètres du modèle, c'est-à-dire l'ensemble des probabilités de transition et d'observation.

Forward-Backward Algorithm Example



Probabilité a_{kl} que l'on passe de l'état k à l'état l à la position i de la séquence est donc :

$$P(\pi_i = k, \pi_{i+1} = l | x, \theta) = \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x)}$$

On peut donc en déduire le nombre attendu de fois que a_{kl} est utilisé en sommant sur toutes les positions et toutes les séquences d'apprentissage :

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1)$$

(Eq. 1)

avec $f_k^j(i)$
 $b_l^j(i)$

La variable forward $f_k(i)$ calculée
pour la séquence j
La variable backward correspondante

De même, on peut calculer la probabilité que la symbole x_i soit émis par l'état k . Cela correspond à la probabilité d'être à l'état k à la position i , c'est-à-dire à la probabilité *à posteriori* :

$$E_k(x_i) = P(\pi_i = k | x) = \frac{f_k(i)b_k(i)}{P(x)}$$

On peut donc en déduire le nombre attendu de fois où le symbole a apparaît à l'état k en sommant sur toutes les positions et toutes les séquences d'apprentissage :

$$E_k(a) = \sum_j \frac{1}{P(x^j)} \sum_{\{i|x_i^j=a\}} f_k^j(i)b_k^j(i) \quad (\text{Eq. 2})$$

La somme interne est réalisée sur les positions i pour lesquelles le symbole émis est a .

Une fois ces estimations calculées, les nouveaux paramètres sont calculés comme précédemment.

$$a_{kl} = \frac{A_{kl} + r_{kl}}{\sum_{l'} A_{kl'} + r_{kl'}} \quad \text{et} \quad e_k(b) = \frac{E_k(b) + r_k(b)}{\sum_{b'} E_k(b') + r_k(b')} \quad (\text{Eq. 3})$$

On ré-estime les probabilités en utilisant les nouvelles valeurs calculées des a_{kl} et $e_k(b)$ comme paramètres θ du modèle

Critère de convergence : arrêt quand le changement de la valeur de la nouvelle log-vraisemblance (log likelihood) est inférieure à un seuil donné prédéfini ou si le nombre d'itérations est dépassé.

La log-vraisemblance du modèle est donnée pour n séquences d'observation dans l'apprentissage par :

$$\sum_{j=1}^n \log P(x^j | \theta)$$

Initialisation : Choix arbitraire des paramètres du modèle

Récurrence

Initialisation de toutes les variables A et E par les valeurs r de pseudocounts ou à 0

Pour chaque séquence $j = 1 \dots n$:

Calculer $f_k(i)$ en utilisant l'algorithme forward

Calculer $b_k(i)$ en utilisant l'algorithme backward

Ajouter la contribution de la séquence j à A (eq. 1) et E (eq. 2)

Calcul des nouveaux paramètres du modèle (eq. 3)

Calcul du nouveau log likelihood du modèle

Terminaison :

Stop si le changement de la valeur du log likelihood est inférieure à un seuil prédéfini

Ou si le nombre maximum d'itération est atteint.

L'algorithme de Baum-Welch est un cas spécifique d'une approche générale d'estimation probabiliste des paramètres appelée EM algorithme (expectation-maximisation).

On peut aussi utiliser l'algorithme de Viterbi pour estimer les paramètres mais moins bonnes performances.

Identification des gènes codant pour des
protéines dans les génomes ou grands
fragments génomiques

Un modèle simple de gènes procaryote

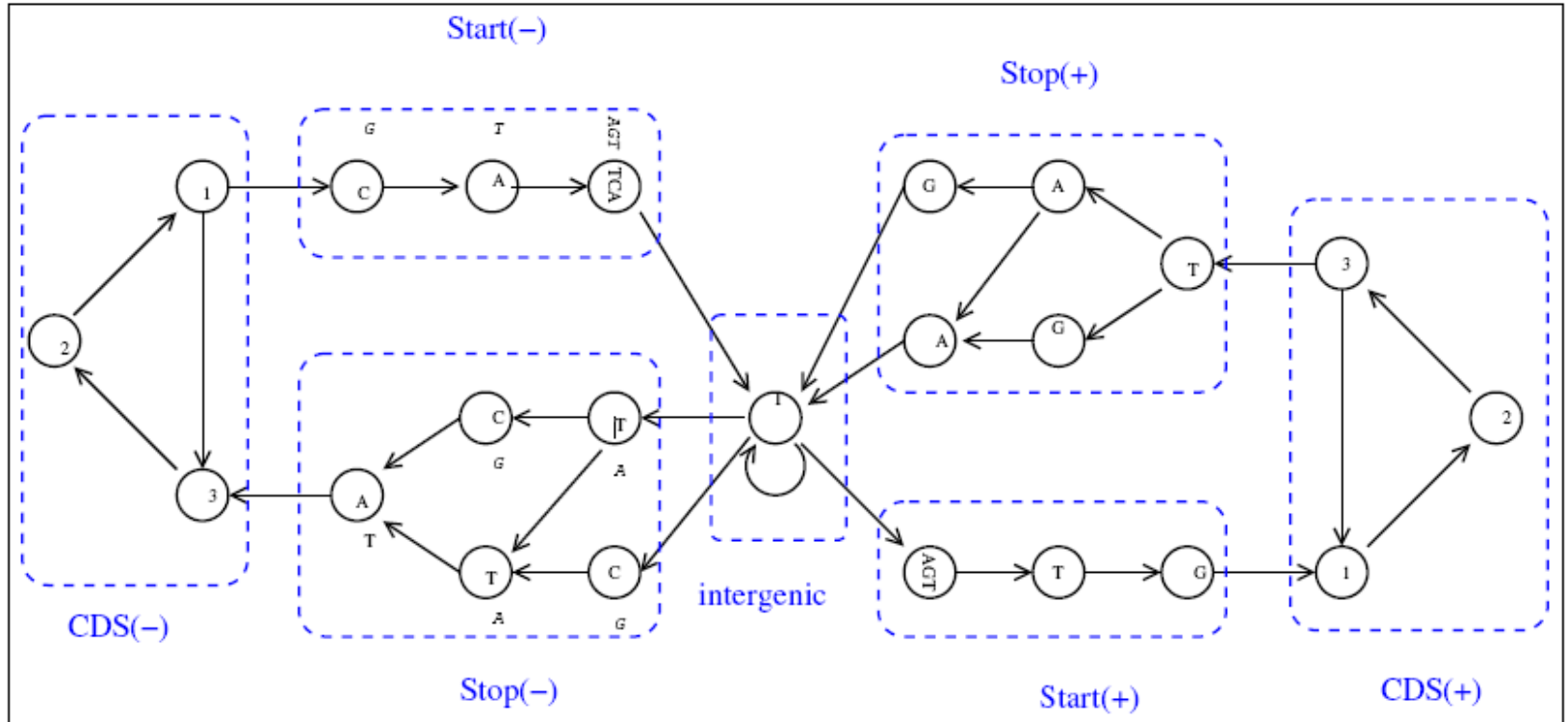


FIG. 1 – Example of a simple HMM dedicated to bacterial coding sequences detection

Les GHMM : Generalized Hidden Markov Model

Les HMM vu précédemment ne permettent pas de prendre en compte la longueur des régions à identifier (CDS, exons, introns etc.).

L'idée est que la probabilité d'observation d'une suite de nucléotides d'un même état n'est pas uniquement le produit des probabilités associées à chaque nucléotides. Elle dépend aussi de la longueur de cette suite, c'est-à-dire que la probabilité qu'un segment de séquence soit dans un état donné est aussi fonction de sa longueur.

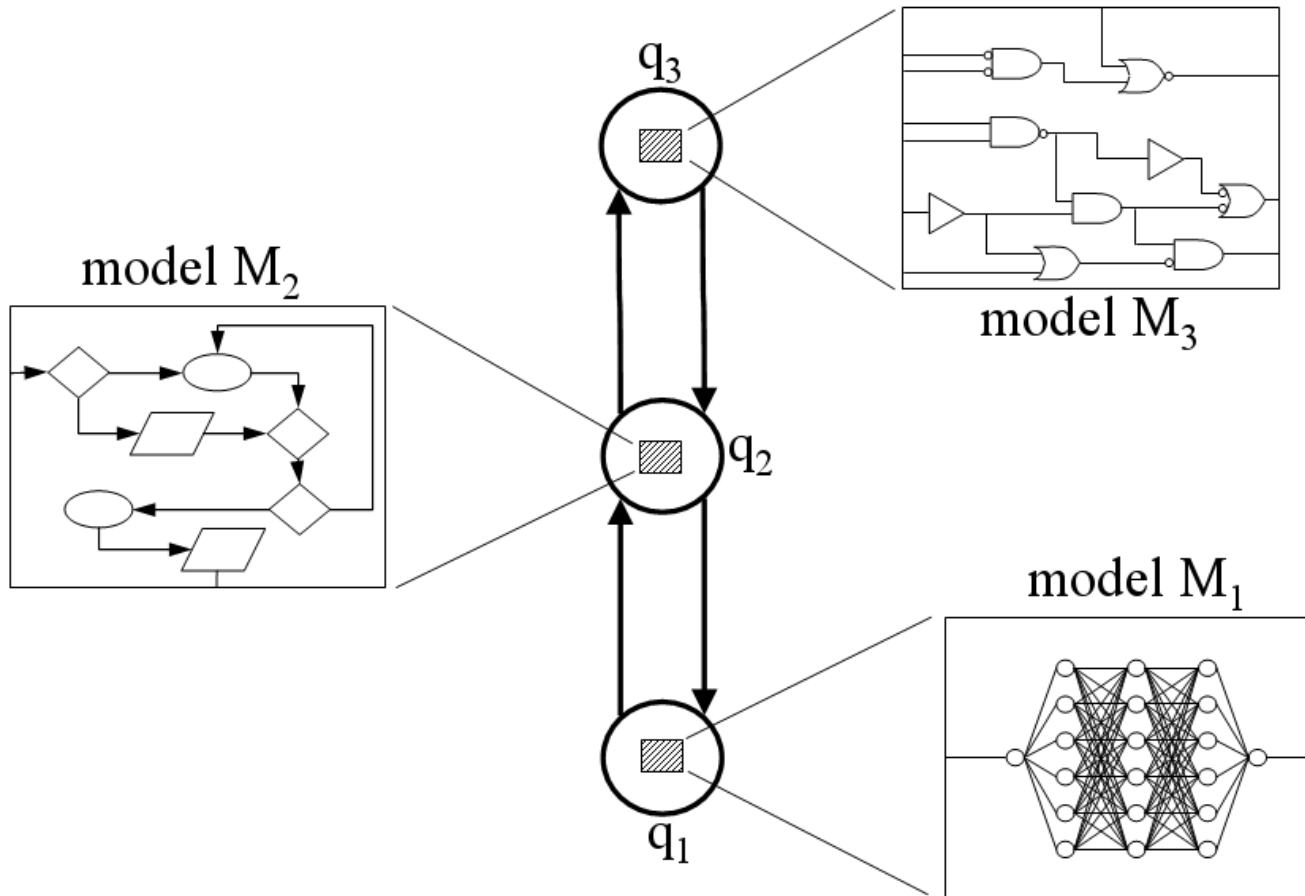
Pour pouvoir prendre cela en compte, il faut recourir à des GHMM.

Un GHMM est défini par :

- un ensemble fini d'états $Q = \{q_0, q_1, \dots, q_m\}$
- Un vecteur de probabilités initiales $\Pi = (\pi_i)$
- un vecteur de probabilités de transition $A = (a_{ij})$
- une matrice de probabilités d'émission $E = (e_i(b))$
- une distribution des longueurs (duration)

GHMM et HMM : principales différences

- Chaque état émet maintenant une sous séquence et non plus juste un symbole
- Les longueurs sont maintenant explicitement modélisées
- Les probabilités d'émission peuvent maintenant être obtenues par n'importe quel modèle probabiliste (des modèles différents peuvent être utilisés suivant l'état à identifier)
- Cela tend à réduire le nombre d'état, donc plus simple et plus facile à modifier



Avantages :

- * abstraction des sous-modèles
- * architecture plus simple
- * modélisation de la durée des états

Désavantage :

- * complexité du décodage

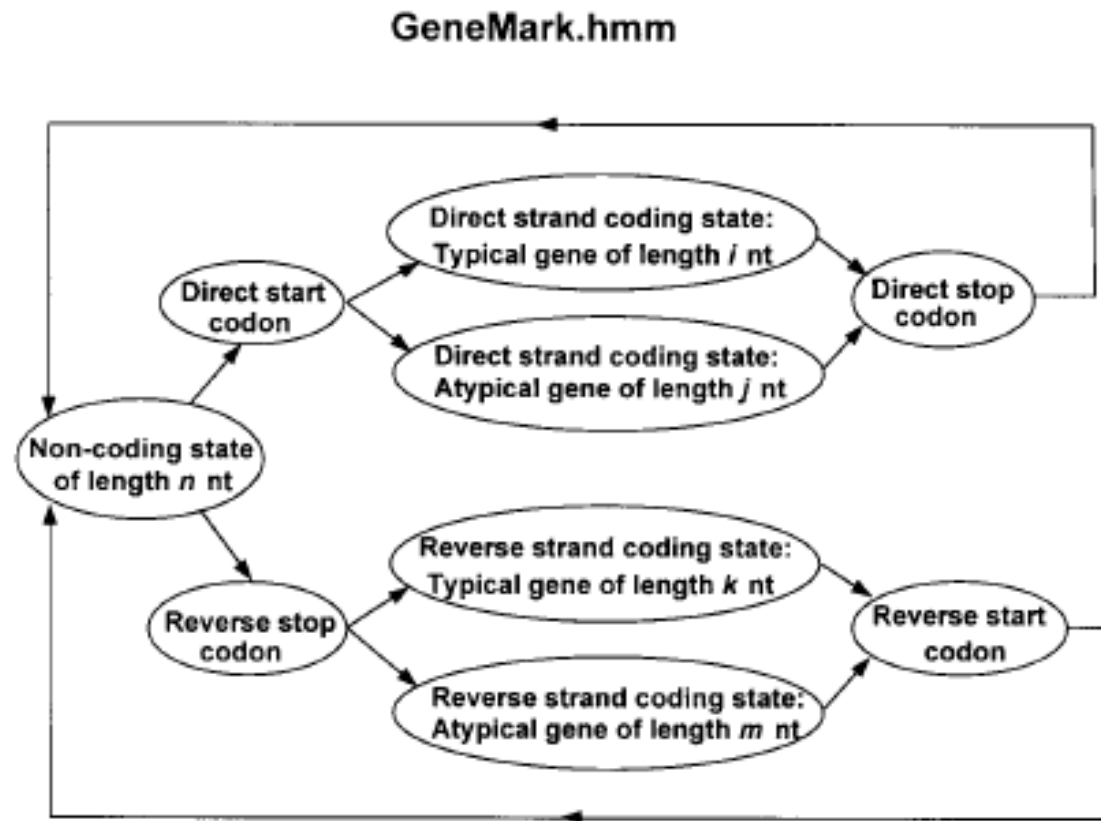


Figure 1. Hidden Markov model of a prokaryotic nucleotide sequence used in the GeneMark.hmm algorithm. The hidden states of the model are represented as ovals in the figure, and arrows correspond to allowed transitions between the states.

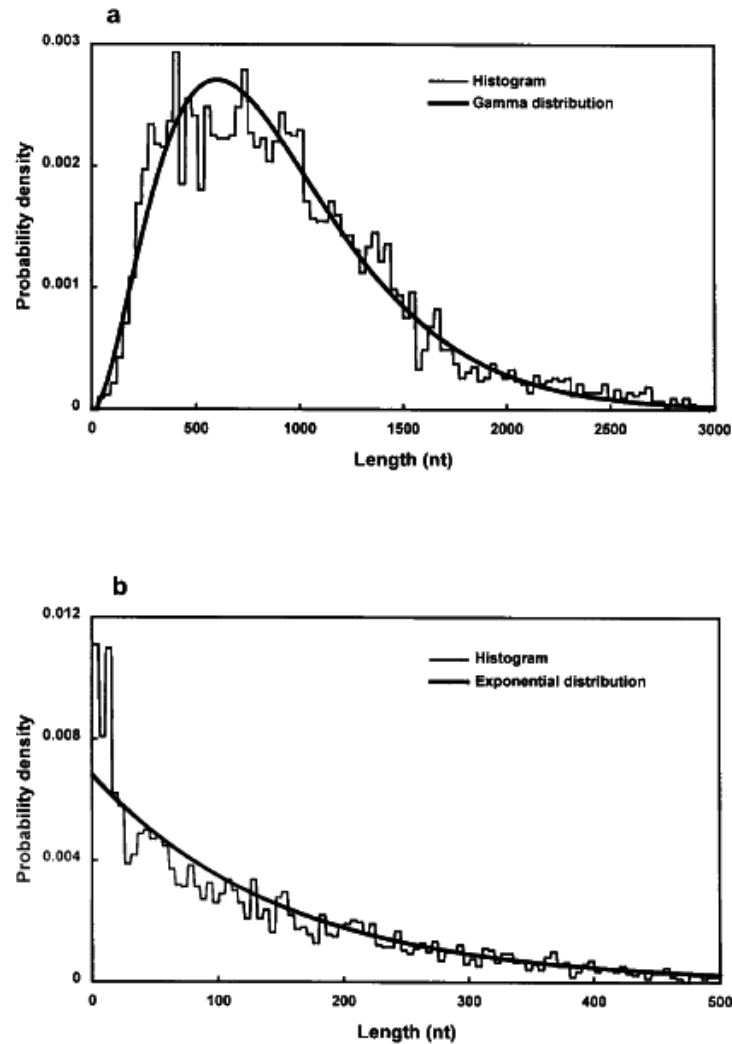
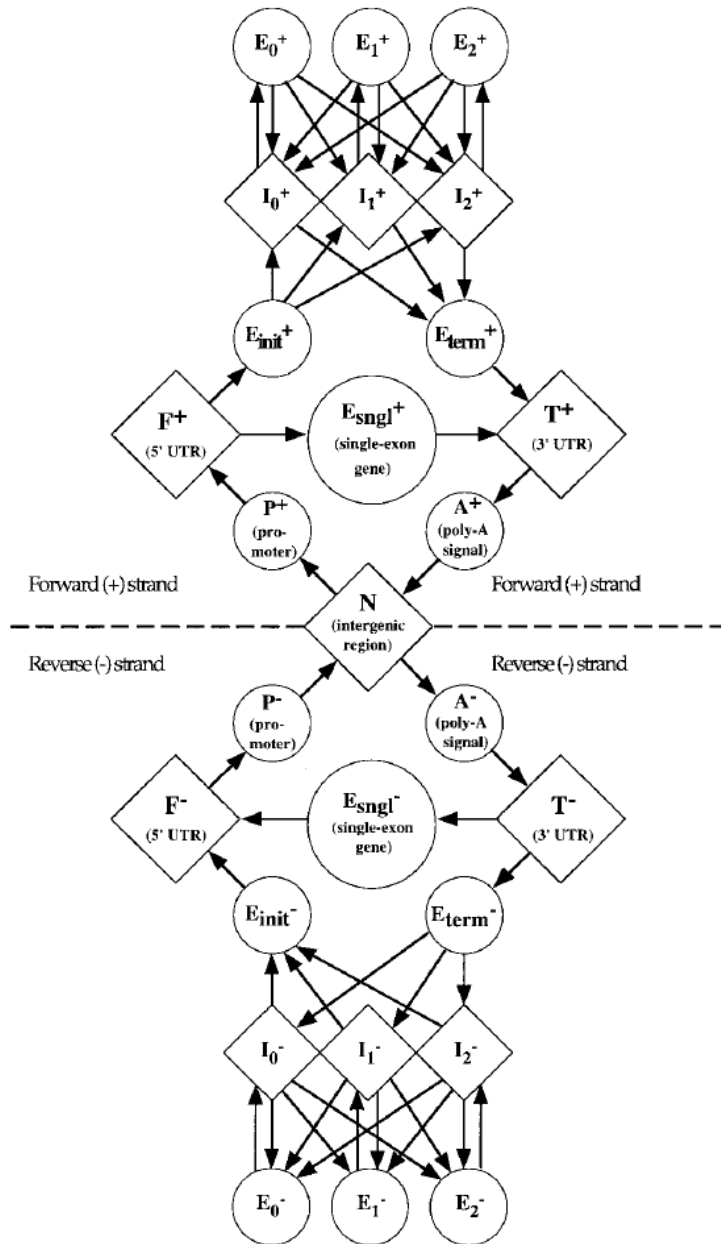


Figure 2. Length distribution probability densities of protein-coding and non-coding regions derived from the annotated *E.coli* genomic DNA (histograms). (a) Coding regions; the solid curve is the approximation by γ distribution $g(d) = N_c(d/D_c)^2 \exp(-d/D_c)$, where d is the length in nt, $D_c = 300$ nt, N_c is the coefficient chosen to normalize the distribution function on the interval from 30 nt (the minimal length of coding region) to 7155 nt (the maximal length). (b) Non-coding regions; the solid curve is the approximation by exponential distribution $f(d) = N_n \exp(-d/D_n)$, where $D_n = 150$ nt. The coefficient N_n normalizes the distribution function on the interval from 1 to 1000 nt.

Modèles de gènes : génomes eucaryotes

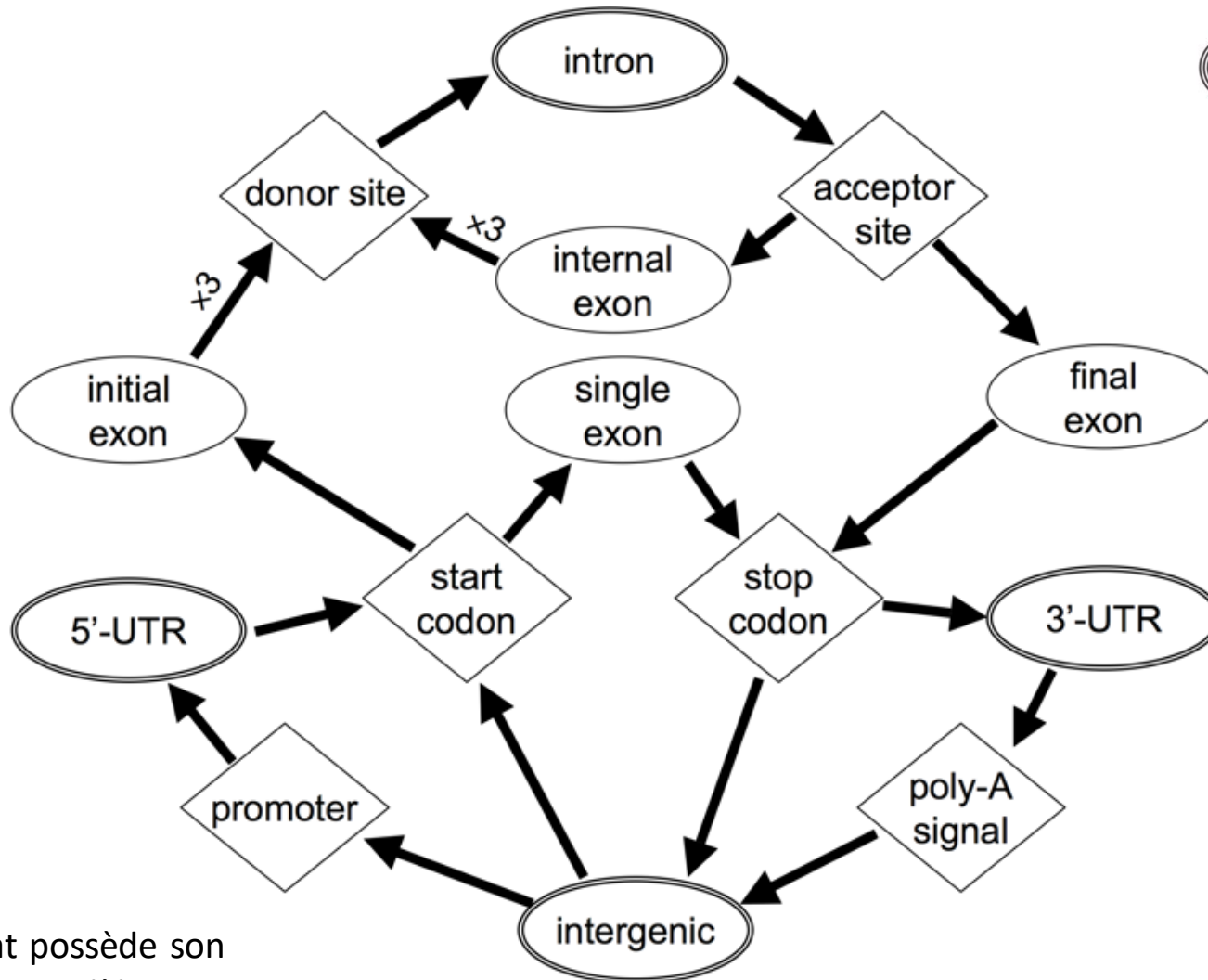


GHMM : modèle de GENSCAN

(extrait de J. Mol. Biol. (1997) 268, 78-94)

Losanges : états de longueur fixe
(états représentant l'information
de type signal)

Ovales : états de longueur variable
(états représentant l'information de
type contenu)



Chaque état possède son
propre sous-modèle

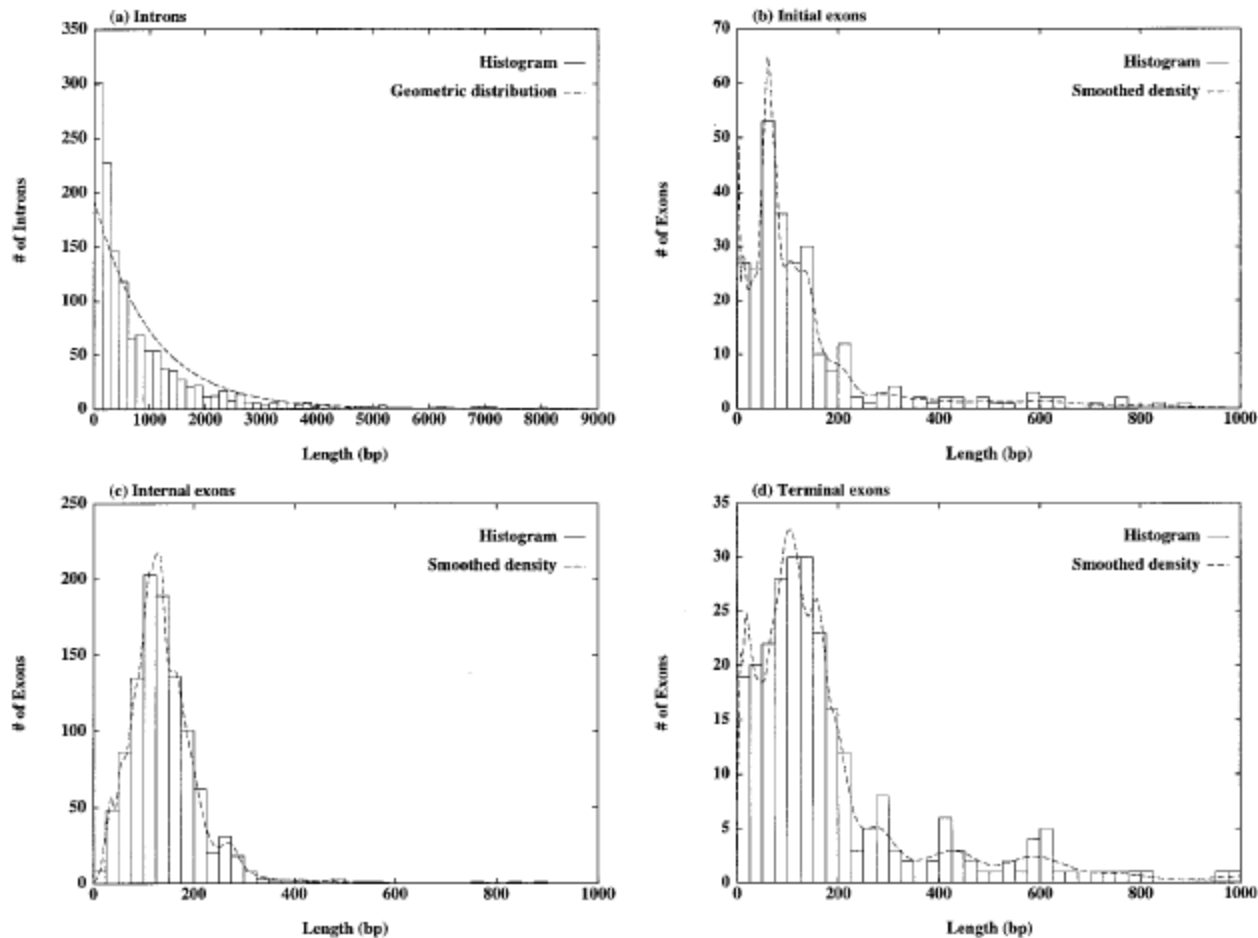


Figure 4. Length distributions are shown for (a) 1254 introns; (b) 238 initial exons; (c) 1151 internal exons; and (d) 238 terminal exons from the 238 multi-exon genes of the learning set \mathcal{L} . Histograms (continuous lines) were derived with a bin size of 300 bp in (a), and 25 bp in (b), (c), (d). The broken line in (a) shows a geometric (exponential) distribution with parameters derived from the mean of the intron lengths; broken lines in (b), (c) and (d) are the smoothed empirical distributions of exon lengths used by GENSCAN (details given by Burge, 1997). Note different horizontal and vertical scales are used in (a), (b), (c), (d) and that multimodality in (b) and (d) may, in part, reflect relatively small sample sizes.

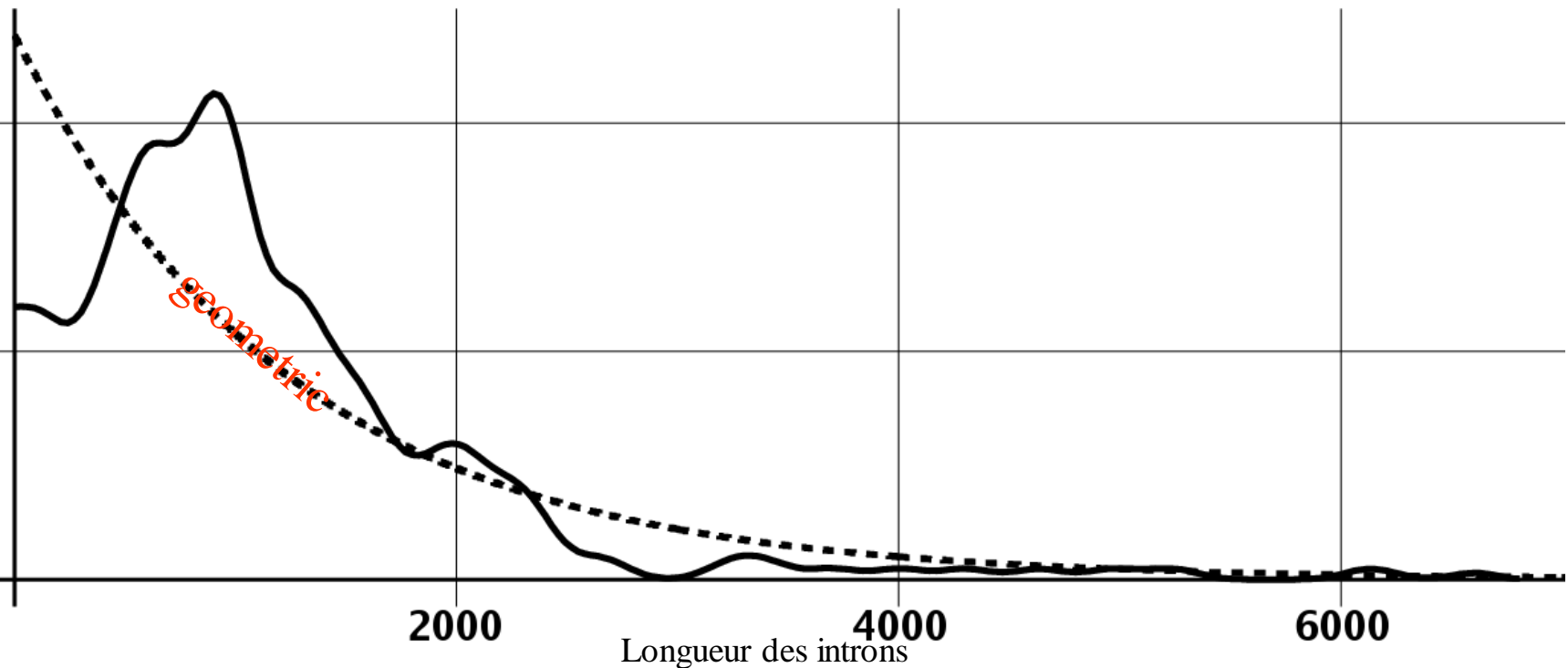
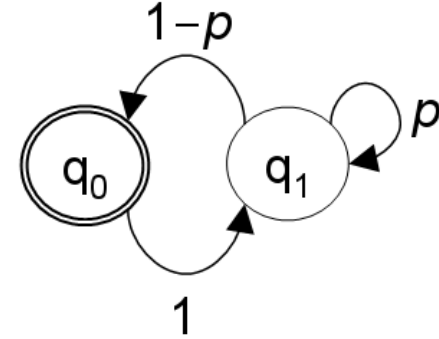
Prise en compte de la longueur dans le calcul de la probabilité:

p : probabilité que le processus reste dans l'état intron

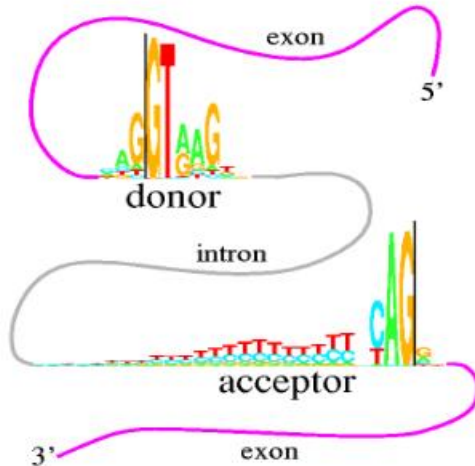
$1-p$: probabilité de passer à l'état exon

$p^{d-1}(1-p)$ = probabilité que l'intron est une longueur de $d-1$ dans le cas d'une loi géométrique

$$P(x_0 \dots x_{d-1} \mid \theta) = \left(\prod_{i=0}^{d-1} P_e(x_i \mid \theta) \right) \underbrace{p^{d-1}(1-p)}_{\text{distribution géométrique}}$$



Détection des jonctions d'épissage :



Choix pour établir une matrice de poids à partir d'un ensemble de séquences d'intérêt alignées :

- Positions considérées comme indépendantes (Weight Matrix Method, WMM)
- Positions adjacentes considérées comme dépendantes (Weight Array Method, WAM)

- ✓ Pour jonctions 3' d'épissage (acceptor) utilisation d'un modèle WAM
- ✓ Pour les jonctions 5' d'épissage (donor), il y a bien dépendance entre positions dans le motif mais ces positions ne sont pas adjacentes. Introduction d'un nouveau modèle Maximal Dependence Decomposition

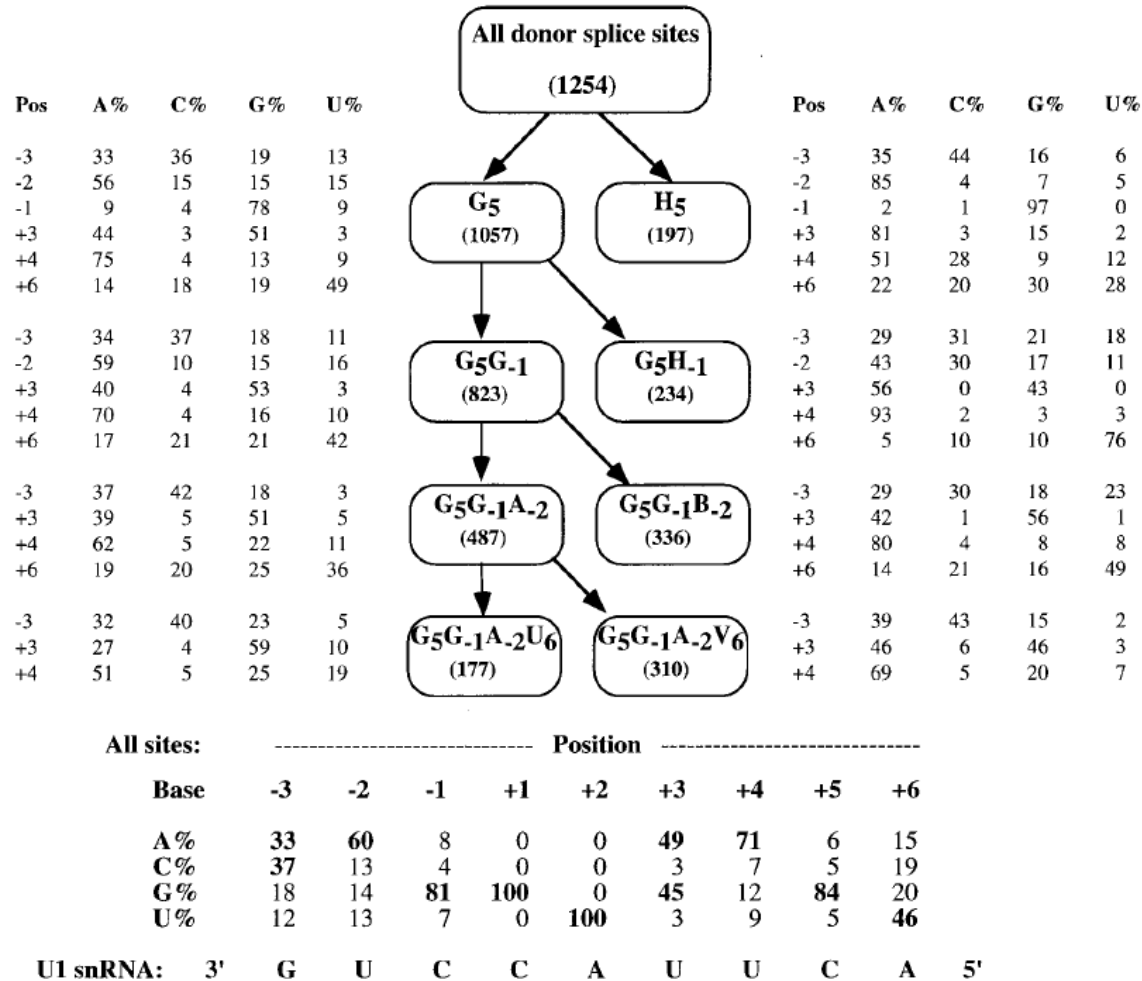


Figure 2. The subclassification of donor splice sites according to the maximal dependence method is illustrated. Each box represents a subclass of donor splice sites corresponding to a particular pattern of matches and mismatches to the consensus nucleotide(s) at a set of positions in the donor site, e.g. G_5 is the set of donor sites with G at position +5 and G_5G_{-1} is the set of donors with G at both positions +5 and -1. Here, H indicates A, C or U; B indicates C, G or U; and V indicates A, C or G. The number of sites in each subset is given in parentheses. The data set and donor site position conventions are as described in the legend to Table 4. The frequencies (percentages) of each of the four nucleotides at each variable position are indicated for each subclass immediately adjacent to the corresponding box. Data for the entire set of 1254 donor sites are given at the bottom of the Figure: the frequencies of consensus nucleotides are shown in boldface. The sequence near the 5' end of U1 snRNA which base-pairs with the donor site is shown at the bottom in 3' to 5' orientation.

Comparaison des performances de plusieurs prédicteurs de gènes

Table 1. Performance comparison for Buset/Guigó set of 570 vertebrate genes
A Comparison of GENSCAN with other gene prediction programs

Program	Sequences	Accuracy per nucleotide				Accuracy per exon				
		Sn	Sp	AC	CC	Sn	Sp	Avg.	ME	WE
GENSCAN	570 (8)	0.93	0.93	0.91	0.92	0.78	0.81	0.80	0.09	0.05
FGENEH	569 (22)	0.77	0.88	0.78	0.80	0.61	0.64	0.64	0.15	0.12
GeneID	570 (2)	0.63	0.81	0.67	0.65	0.44	0.46	0.45	0.28	0.24
Genie	570 (0)	0.76	0.77	0.72	n/a	0.55	0.48	0.51	0.17	0.33
GenLang	570 (30)	0.72	0.79	0.69	0.71	0.51	0.52	0.52	0.21	0.22
GeneParser2	562 (0)	0.66	0.79	0.67	0.65	0.35	0.40	0.37	0.34	0.17
GRAIL2	570 (23)	0.72	0.87	0.75	0.76	0.36	0.43	0.40	0.25	0.11
SORFIND	561 (0)	0.71	0.85	0.73	0.72	0.42	0.47	0.45	0.24	0.14
Xpound	570 (28)	0.61	0.87	0.68	0.69	0.15	0.18	0.17	0.33	0.13
GeneID+	478 (1)	0.91	0.91	0.88	0.88	0.73	0.70	0.71	0.07	0.13
GeneParser3	478 (1)	0.86	0.91	0.86	0.85	0.56	0.58	0.57	0.14	0.09

B GENSCAN accuracy for sequences grouped by C + G content and by organism

Subset	Sequences	Accuracy per nucleotide				Accuracy per exon				
		Sn	Sp	AC	CC	Sn	Sp	Avg.	ME	WE
C + G <40	86 (3)	0.90	0.95	0.90	0.93	0.78	0.87	0.84	0.14	0.05
C + G 40-50	220 (1)	0.94	0.92	0.91	0.91	0.80	0.82	0.82	0.08	0.05
C + G 50-60	208 (4)	0.93	0.93	0.90	0.92	0.75	0.77	0.77	0.08	0.05
C + G >60	56 (0)	0.97	0.89	0.90	0.90	0.76	0.77	0.76	0.07	0.08
Primates	237 (1)	0.96	0.94	0.93	0.94	0.81	0.82	0.82	0.07	0.05
Rodents	191 (4)	0.90	0.93	0.89	0.91	0.75	0.80	0.78	0.11	0.05
Non-mam. Vert.	72 (2)	0.93	0.93	0.90	0.93	0.81	0.85	0.84	0.11	0.06

ME = Missed Exon

WE = Wrong Exon

(extrait de J. Mol. Biol. (1997) 268, 78-94)

Avec des séquences protéiques : GENOMESCAN (intégration de cette information dans le modèle GENSCAN.

Combine le résultats d'une méthode *ab initio* de recherche de la structure en exons-introns des gènes (GenScan) avec une recherche par similarité avec les protéines connues (BlastX)

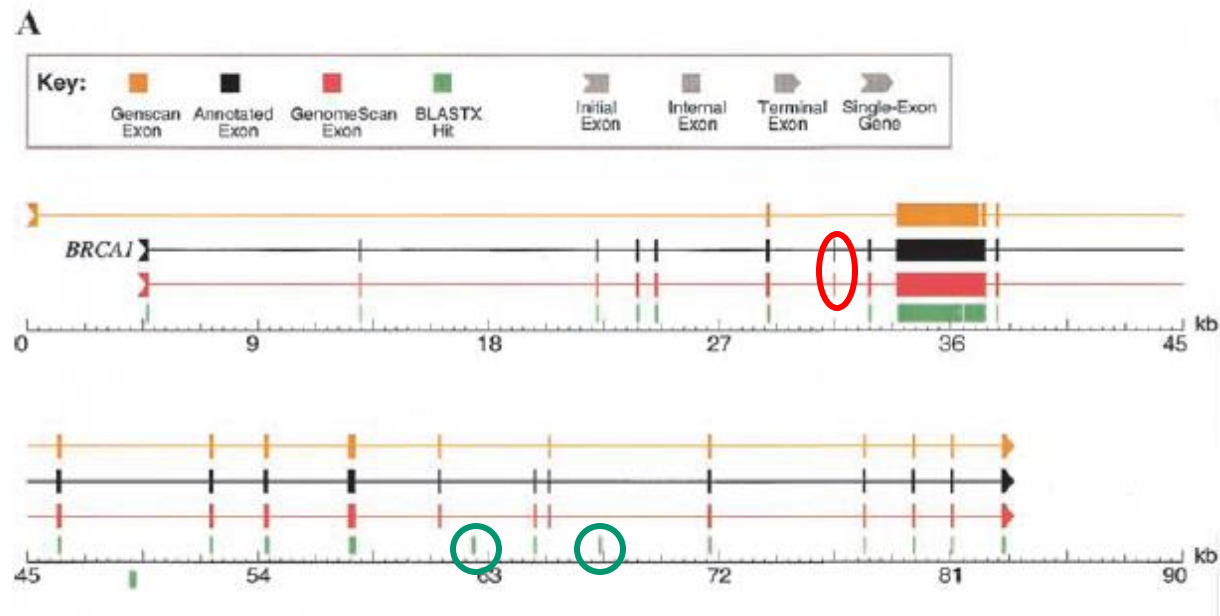
Différence par rapport à GenScan (méthode utilisant un HMM) : la probabilité du chemin est renforcée si la région prédite comme exon présente un hit avec BlastX et est diminuée dans le cas contraire

- Les hits de BlastX qui tombent en N-ter ou C-ter de la protéine sont utilisés pour mieux prédire les codons initiateur et terminateur
- Deux hits BlastX adjacents sur la protéine mais séparés par une distance ≥ 60 pb dans la séquence génomique sont utilisés pour prédire les régions introniques putatives
- si plusieurs hits BlastX chevauchants (information redondante), les résultats sont pré-traités pour ne garder que le hit le plus significatif.

Prise en compte de la similarité

Avec des séquences protéiques : GENOMESCAN (intégration de cette information dans le modèle GENSCAN.

Exemple d'un résultat de prédiction (extrait de Genome Research (2001), 11, 803-816)



Hit BlastX faux-positifs -> pas de prédiction d'exon

Exon 7 : prédit par GenomeScan alors que pas prédit par GenScan et pas de hit BlastX -> pas seulement addition des deux résultats mais fait également sa propre inférence. Utilise le fait qu'il y a une incompatibilité de phase de l'intron entre les exons 6 et 8 -> nécessite la présence d'un autre exon entre les deux.

Prise en compte de la similarité

Avec des séquences génomiques : TWINSCAN (Bioinformatics (2001), 17 suppl. 1, S140-S148, intégration de cette information dans le modèle GENSCAN.

Exploite la similarité entre deux génomes apparentés : ceci permet de détecter les régions codantes et les éléments de régulation.

target sequence : séquence à annoter

informant sequence : séquence génomique apparentée

Les répétitions sont masquées. Alignement des deux séquences avec le programme Wu-BlastN

Codage de la conservation

La conservation de la séquence est traduite en symboles sous la séquence ADN :

| = match

: = mismatch

. = non aligné

Les gaps dans la séquence « target » sont ignorés et ceux dans la séquence « informant » sont considérés comme des mismatches.

	10	20	30
position dans la séquence target	123456789	123456789	123456789 123456789
séquence target	ATTTAGCCTACTGAAATGGACCGCTTCAGCATGGTATCC		
séquence conservation	: 	: :	: :

Fig. 1. An example DNA sequence together with the corresponding conservation sequence.

Calcul de la probabilité

Quand on recherche le chemin le plus probable dans le HMM, calcul de la probabilité que la séquence soit dans l'état i (par exemple : exons, UTR etc.)

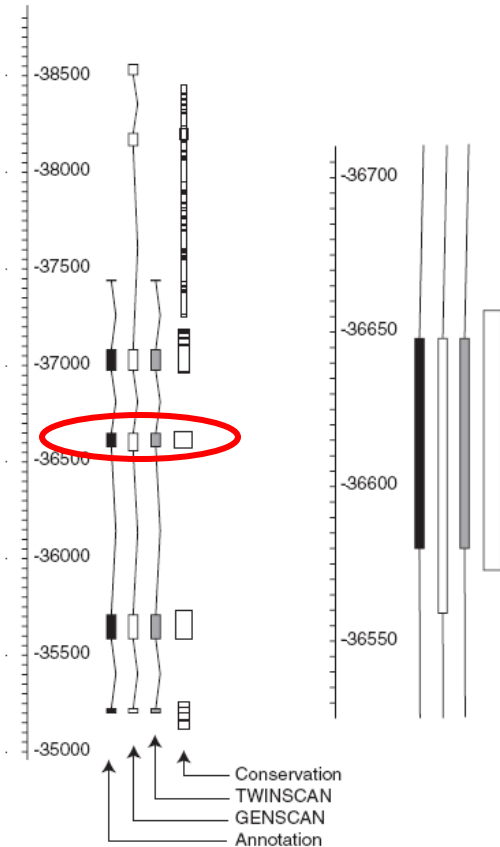
$$P_e(i) = \text{Proba_GeneScan}(i) \times \text{Proba_Conservation}(i)$$

Calcul de la probabilité de conservation

Modèle de Markov d'ordre 5

Estimée par état (codant, UTR, non codant...) à partir de séquences annotées formant l'ensemble d'apprentissage.

Avec des séquences génomiques : TWINSCAN (intégration de cette information dans le modèle GENSCAN.



Exemple d'un résultat de prédiction (extrait de Bioinformatics (2001), 17 suppl. 1, S140-S148)

GenScan ne prédit pas le premier exon et en prédit deux supplémentaires.

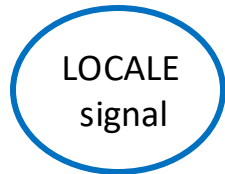
TwinScan prédit correctement le premier exon car le niveau de conservation de séquence à son niveau augmente la probabilité totale de l'exon.

Exon 3 : mauvais site d'épissage en 5' choisi par GenScan. Bonne prédiction de TwinScan car la conservation de séquence s'arrête avant la position de fin de l'exon prédit par GenScan

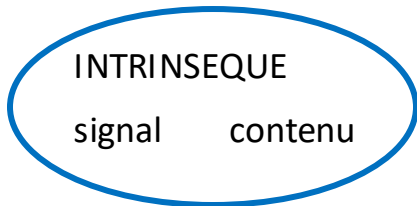
Fig. 5. Detailed view of the annotation, gene predictions and conservation at the L44L gene (AAB47245.1) from the *Mus musculus* Bruton's tyrosine kinase locus (U58105.1). The magnification at right shows the region around exon 3. The width of boxes representing BLAST alignments corresponds to the quality of the alignment. The image comes from ACEDB.

Evolution de l'intégration des sources d'informations

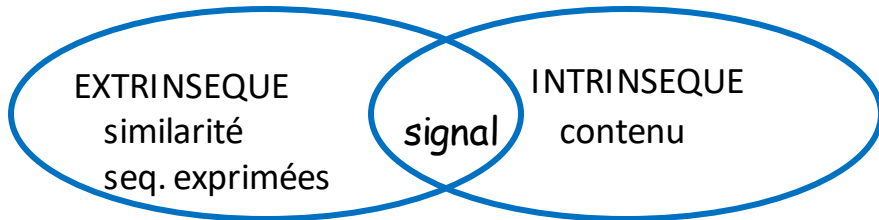
(extrait de la thèse de Sylvain Foissac, 2004)



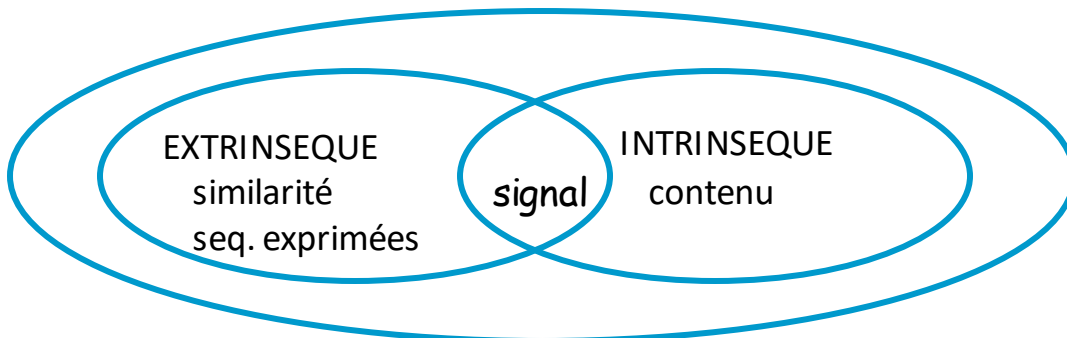
Deux sources traitées par des méthodes indépendantes (ex : Staden, 1984; Gelfand, 1990)



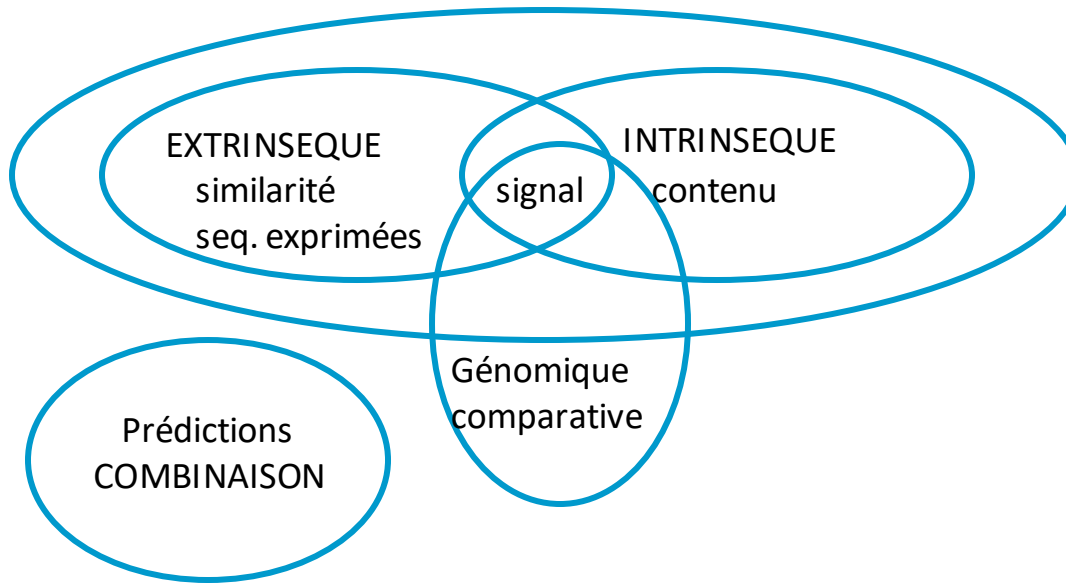
Intégration de ces deux sources dans un même logiciel (ex : Guigo et al., 1992, logiciel GENSCAN (Burge et Karlin, 1997))



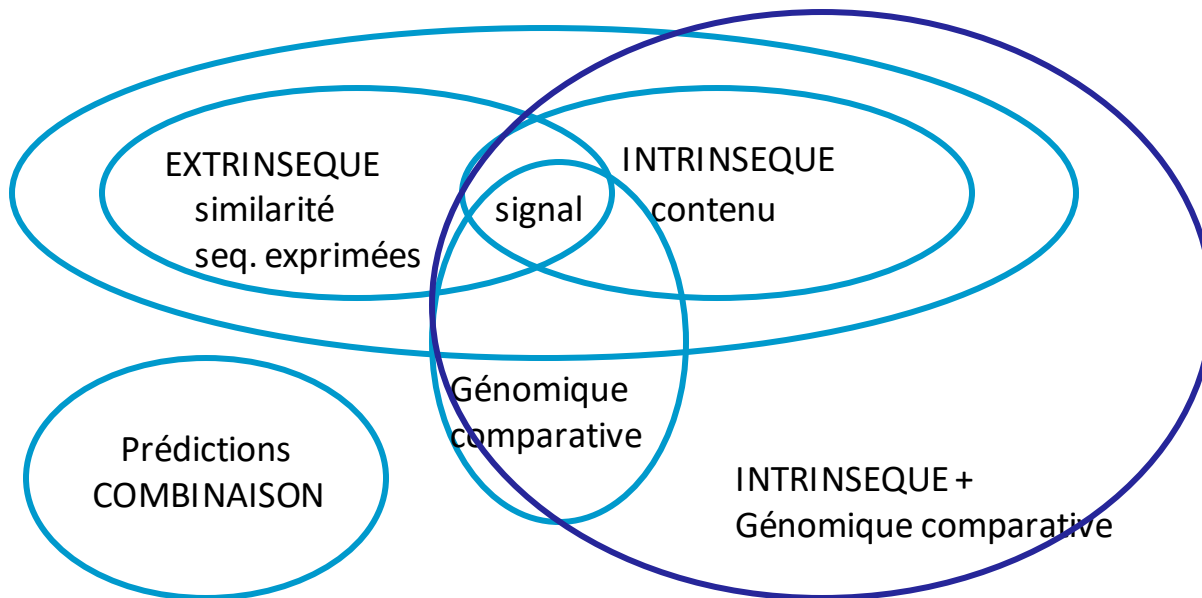
Augmentation des données d'expression, prise en compte de la similarité de séquences (Borodovsky et al., 1994, Fickett, 1995)



Intégration de ces deux types d'information dans de nombreux logiciels. GENOMESCAN (Yeh et al., 2001) résulte de l'intégration de similarité protéique dans GENSCAN

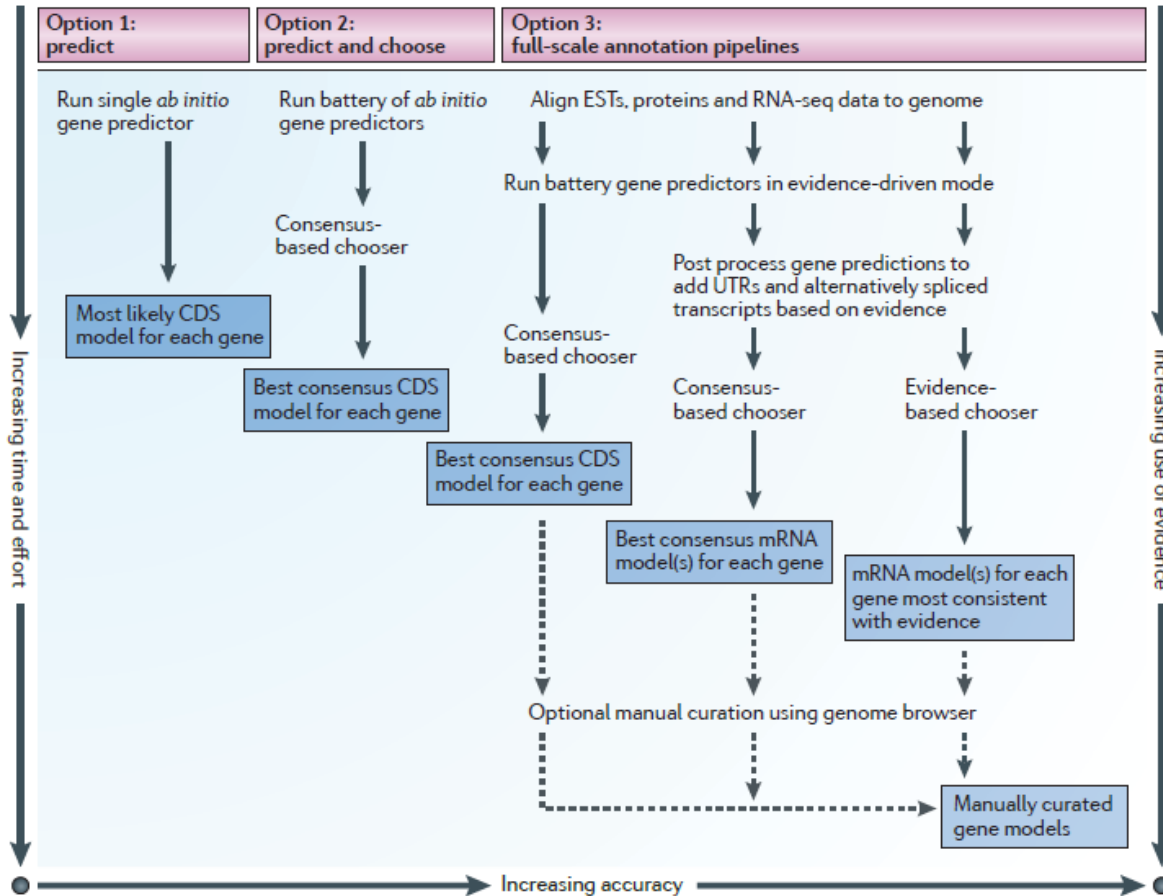


Conservation entre séquences génomiques exploitées en combinaison seulement avec des informations de type signal



Génomique comparative intégrée uniquement dans des méthodes intrinsèque. Par exemple, TWINSCAN (Korf et al., 2001) intègre la génomique comparative dans GENSCAN.

Synthèse sur les approches



(extrait de Nature reviews Genetics (2012) 13, 329-42. M. Yandell and D. Ence, A beginner's guide to eukaryotic genome annotation)

Figure 2 | **Three basic approaches to genome annotation and some common variations.** Approaches are compared on the basis of relative time, effort and the degree to which they rely on external evidence, as opposed to *ab initio* gene models. The y axis shows increasing time and effort; the x axis shows increasing use of external evidence and, consequently, increasing accuracy and completeness of the resulting gene models. The type of final product produced by each kind of pipeline is shown in the dark blue boxes. Relative positions in the figure are for summary purposes only and are not based on precisely computed values. See TABLE 1 for a list of commonly used software components. CDS, coding sequence; EST, expressed sequence tag; RNA-seq, RNA sequencing; UTR, untranslated region.

Une Revue récente :

Ejigu GF, Jung J. Review on the Computational Genome Annotation of Sequences Obtained by Next-Generation Sequencing. Biology (Basel). 2020 Sep 18;9(9):295. doi: 10.3390/biology9090295. PMID: 32962098; PMCID: PMC7565776.

Environnements intégrés pour l'annotation des génomes procaryotes



- ✓ **RAST server** (BMC Genomics. 2008 Feb 8;9:75. doi: 10.1186/1471-2164-9-75)

DESCRIPTION:

We describe a fully automated service for annotating bacterial and archaeal genomes. The service identifies protein-encoding, rRNA and tRNA genes, assigns functions to the genes, predicts which subsystems are represented in the genome, uses this information to reconstruct the metabolic network and makes the output easily downloadable for the user (<https://rast.nmpdr.org/>)

- ✓ **Prokka** (Bioinformatics. 2014, 30:2068-69)

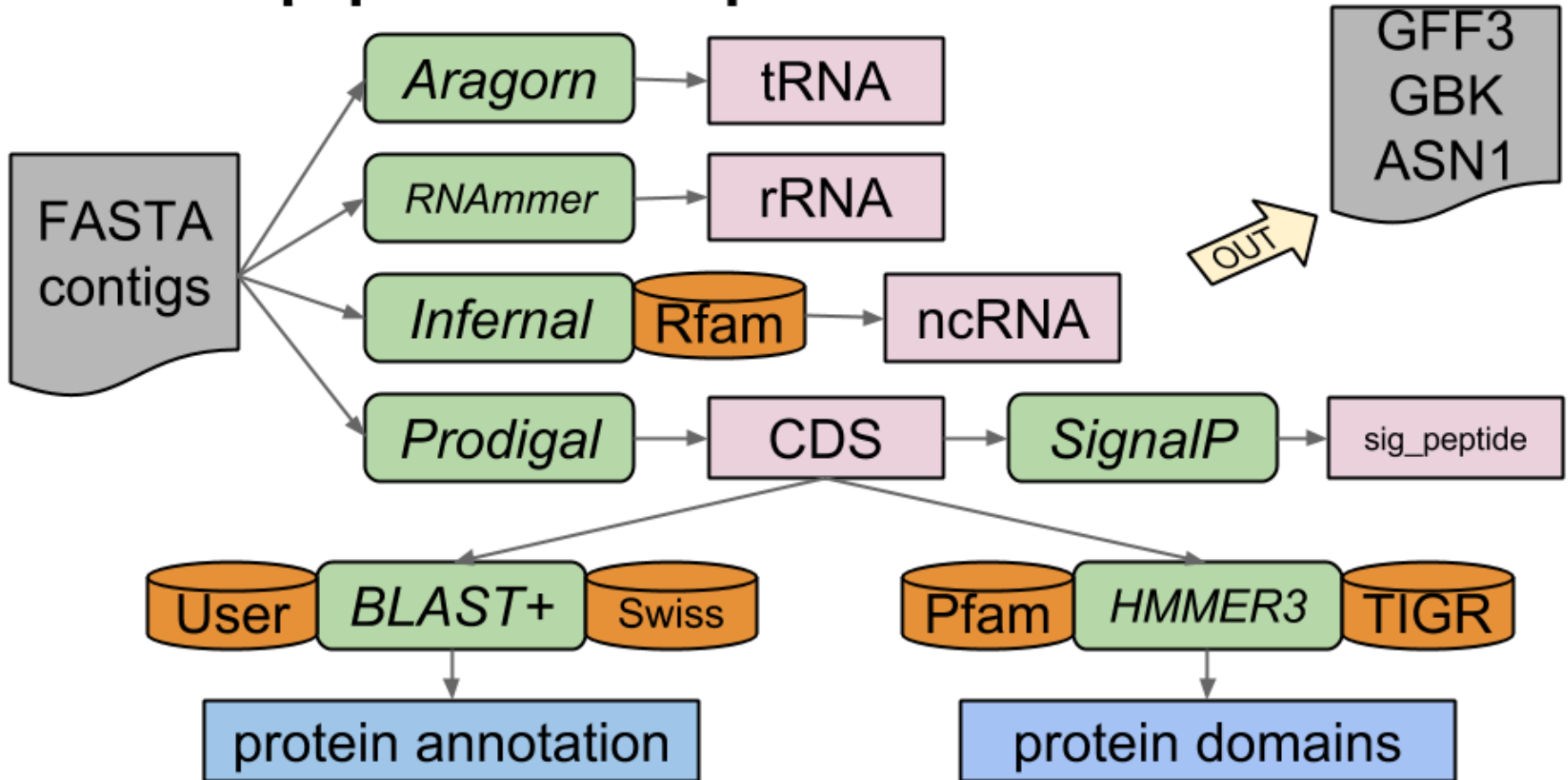
DESCRIPTION:

Prokka, a command line software tool to fully annotate a draft bacterial genome in about 10min on a typical desktop computer. It produces standards-compliant output files for further analysis or viewing in genome browsers.

Tool (reference)	Features predicted
Prodigal (Hyatt 2010)	Coding sequence (CDS)
RNAmmer (Lagesen <i>et al.</i> , 2007)	Ribosomal RNA genes (rRNA)
Aragorn (Laslett and Canback, 2004)	Transfer RNA genes
SignalP (Petersen <i>et al.</i> , 2011)	Signal leader peptides
Infernal (Kolbe and Eddy, 2011)	Non-coding RNA

Méthodes utilisées dans Prokka
(issu de Bioinformatics. 2014,
30:2068-69)

Prokka pipeline (simplified)



Environnements intégrés pour l'annotation des génomes procaryotes



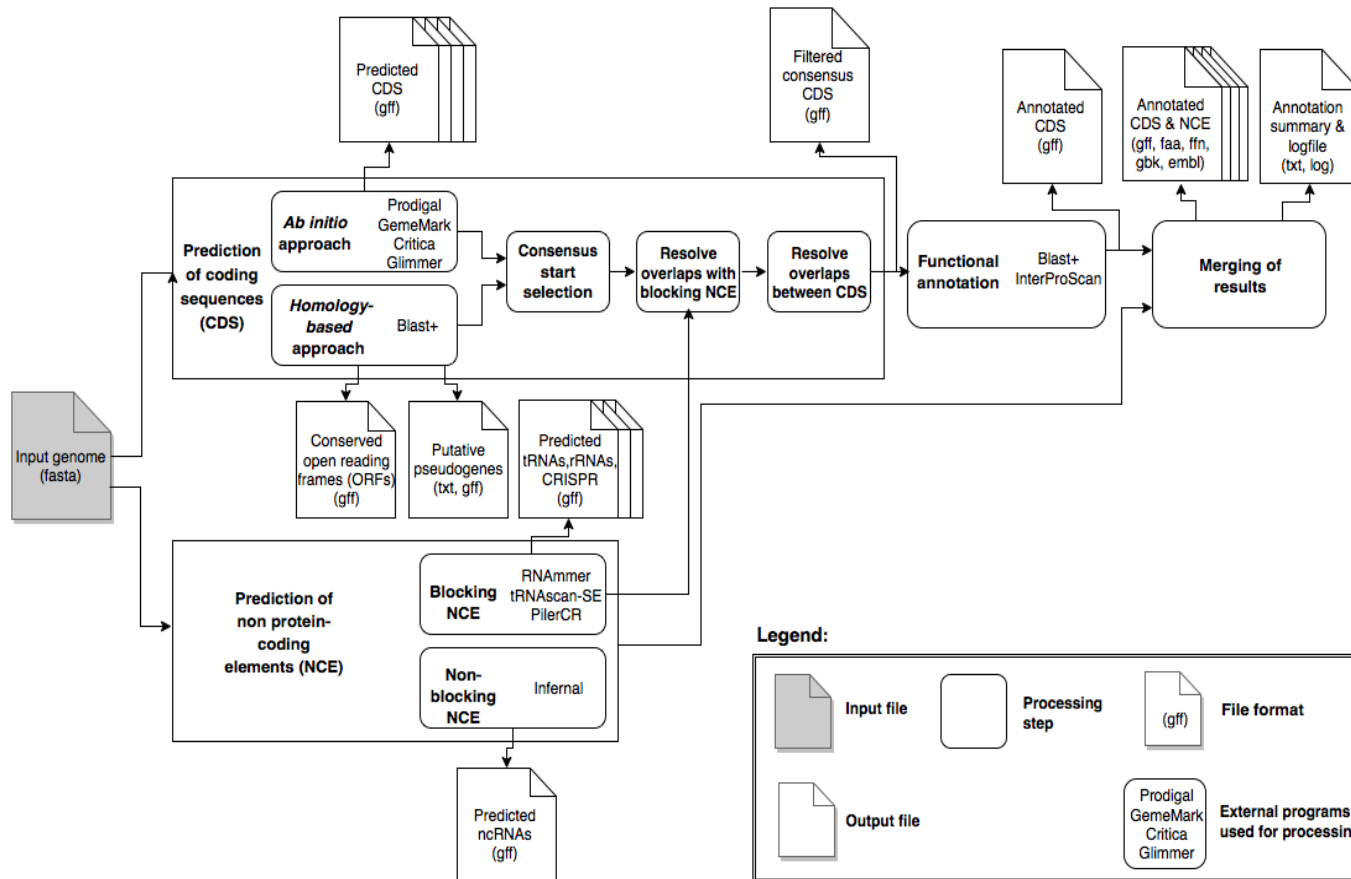
✓ **ConsPred** (Bioinformatics. 2016, 32:3327-29)

DESCRIPTION:

We present ConsPred, a prokaryotic genome annotation framework that performs intrinsic gene predictions, homology searches, predictions of non-coding genes as well as CRISPR repeats and integrates all evidence into a consensus annotation. ConsPred achieves comprehensive, high-quality annotations based on rules and priorities, similar to decision-making in manual curation and avoids conflicting predictions. Parameters controlling the annotation process are configurable by the user.

Environnements intégrés pour l'annotation des génomes procaryotes

Exemple de ConsPred (Weinmaier *et al.*, 2016, Bioinformatics, 32:3327-29)



(Extrait de Weinmaier *et al.*)

Figure S1. ConsPred workflow

Coding sequences (CDS) are predicted by combining different *ab initio* gene predictions, and conserved open reading frames (ORFs) detected by homology search against the NCBI nr database. Database entries from closely related taxa are excluded to prevent possible misannotations due to low phylogenetic distance. Putative pseudogenes are exported for user inspection. From all predicted non-protein-coding elements (NCE) those that biologically must not overlap with CDS are considered blocking NCE. CDS overlapping with blocking NCE are removed. Filtered consensus CDS are obtained from predicted CDS and conserved ORFs by using predefined weights and rules and subsequent removal of CDS that overlap with blocking NCEs. Filtered consensus CDS are functionally annotated and then merged with the NCE into the final annotation files.

Environnements intégrés pour l'annotation des génomes prokaryotes et eucaryotes

GenSAS v6.0

<https://www.gensas.org/>

Home

Use GenSAS

Available Tools

Available Libraries

New Features

Help

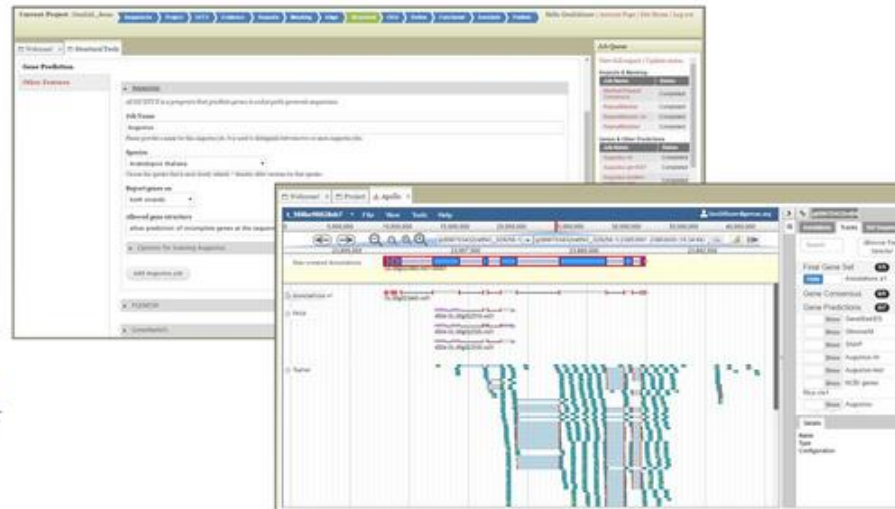
Cite GenSAS

GenS

JL Humann et al. "Structural and Functional Annotation of Eukaryotic Genomes with GenSAS." Gene Prediction: Methods and Protocols. Ed. Martin Kollmar. New York: Humana Press, 2019. 29-51.

Welcome to GenSAS

The Genome Sequence Annotation Server (GenSAS) is an online platform that provides a pipeline for whole genome structural and functional annotation for eukaryotes and prokaryotes. Users can upload genome sequences and select from a variety of tools for repeat masking, prediction of gene models and other structural features as well as functional annotation tools. GenSAS integrates with **JBrowse** and **Apollo** to provide visualization and editing. For more details about GenSAS, please see our **User's Guide**.



Méthodes disponibles dans GenSAS



Category	Tool Name	Version	Description
Repeat Masking	RepeatMasker	4.1.1	Masks repeated sequences
	RepeatModeler	2.0.1	<i>de novo</i> repeat masker
Gene Prediction	Augustus	3.4.0	Eukaryotic gene predictor
	BRAKER2	2.1.5	Eukaryotic gene predictor
	GeneMarkES	4.48	Eukaryotic gene predictor
	GeneMarkS	4.32	Prokaryotic gene predictor
	Genscan	1.0	Eukaryotic gene predictor
	Glimmer3	3.0.2	Prokaryotic gene predictor
	GlimmerM	2.5.1	Eukaryotic gene predictor
	SNAP	11/29/2013	Semi-HMM-based Nucleic Acid Parser, gene predictor
Alignments	BLAST+	2.12.0	Alignment tool for transcripts and proteins
	pBLAT	2.5	Alignment tool for transcripts
	DIAMOND	2.0.6	Alignment tool for protein evidence
	HISAT2	2.2.1	Aligns RNA-seq reads to genome
	PASA	2.4.1	Aligns EST or transcript evidence
	TopHat	2.1.1	Aligns RNA-seq reads to genome
Genomic Features	Getorf	EMBOSS-6.6.0	Open reading frame finder
	RNAmmer	1.2	Finds rRNA sequences
	SSR Finder	N/A	Finds user defined simple sequence repeats
	tRNAScan-SE	2.0.7	Identifies tRNA sequences
Consensus Gene Models	EVidenceModeler	1.1.1	Creates consensus gene structures for eukaryotes
Functional Annotation	BLAST+	2.12.0	Alignment tool for proteins
	DIAMOND	2.0.6	Alignment tool for proteins
	InterProScan	5.53-87.0	Classifies proteins by families and identifies functional domains
	Pfam	1.6	Uses Pfam database to identify functional domains and families in proteins
	SignalP	5.0	Predicts signal peptide cleavage sites in protein sequences
	TargetP	2.0	Predicts the subcellular location of eukaryotic proteins based on signal peptide sequences
Genome Browser	Apollo	2.0.7	Annotation editor
	JBrowse	1.12.1	Genome browser
Assembly/annotation metrics	BUSCO	5.2.2	Determines genome assembly and annotation completeness based on orthologs
	One code to find them all	1.0	Reports repeat types from RepeatMasker
	PRINSEQ-lite	0.20.4	Used to analyze assembly file for sequence number and size

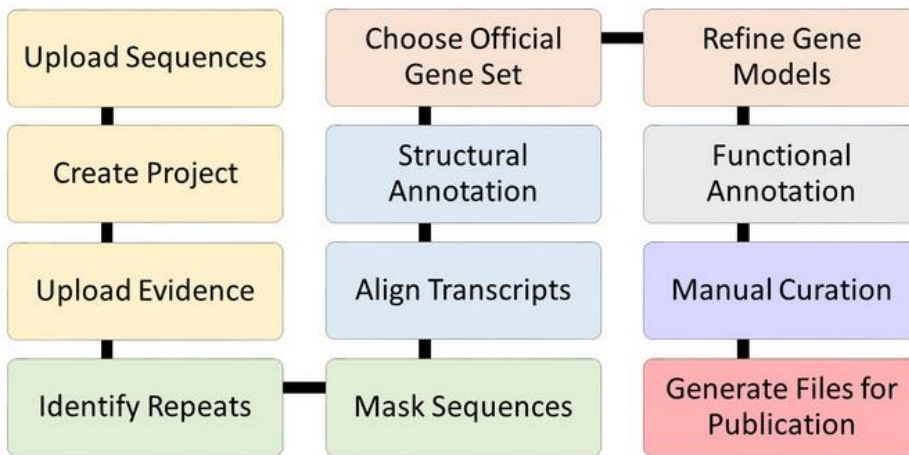
Bases de données disponibles dans GenSAS

GenSAS provided library	RNA number	Protein number	Release date
refseq_archaea	1,354	2,936,792(nr)	5/4/2023
refseq_bacteria	28,459	210,034,561(nr)	5/4/2023
refseq_fungi	5,154,195	5,163,994	5/5/2023
refseq_invertebrate	9,708,846	8,686,644	5/4/2023
refseq_mitochondrion	NA	240,117	5/4/2023
refseq_plant	8,805,618	8,192,225	5/4/2023
refseq_plasmid	7	2,080,794(nr)	5/4/2023
refseq_plastid	44	1,064,130	5/5/2023
refseq_protozoa	1,083,003	1,164,421	5/4/2023
refseq Vertebrate_mammalian	12,211,560	10,215,066	5/4/2023
refseq Vertebrate_other	15,510,388	13,672,291	5/4/2023
refseq_viral	NA	655,246	5/4/2023
sprot	NA	569,793	6/28/2023
trembl	NA	248,272,897	6/28/2023
RepBase library	NA	NA	12/24/2018

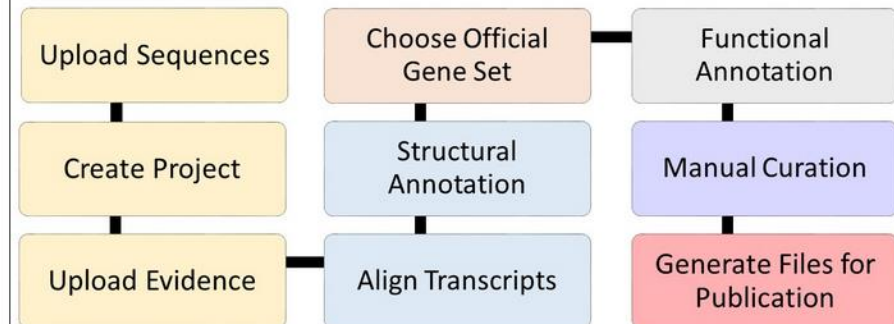
The steps on the flowcharts with the same color represent the major steps of annotation:

1. Project creation/uploading sequences and evidence
2. Repeat identification and masking (for eukaryotes)
3. Structural annotation
4. Selection of official gene set (and refinement for eukaryotes)
5. Functional annotation
6. Manual curation of the annotation
7. Preparing files for publication

GenSAS eukaryotic annotation process



GenSAS prokaryotic annotation process



Environnement pour des génomes eucaryotes

Ensembl (Howe *et al.*, *Nucleic Acids Res.* 2021, vol. 49(1):884–891)

Tools

[All tools](#)

BioMart >

Export custom datasets from Ensembl with this data-mining tool

BLAST/BLAT >

Search our genomes for your DNA or protein sequence

Variant Effect Predictor >

Analyse your own variants and predict the functional consequences of known and unknown variants

Search

All species for

e.g. BRCA2 or rat 5:62797383-63627669 or rs699 or coronary heart disease

All genomes

-- Select a species --



Pig breeds

Pig reference genome and 12 additional breeds

[View full list of all species](#)

Favourite genomes



Human

GRCCh38.p13

[Still using GRCh37?](#)



Mouse

GRCm39



Zebrafish

GRCz11

Ensembl is a genome browser for vertebrate genomes that supports research in comparative genomics, evolution, sequence variation and transcriptional regulation. Ensembl annotate genes, computes multiple alignments, predicts regulatory function and collects disease data. Ensembl tools include BLAST, BLAT, BioMart and the Variant Effect Predictor (VEP) for all supported species.

Ensembl Release 104 (May 2021)

- Update to the Ensembl Canonical transcript set.
- Human and mouse gene sets updated to GENCODE 38 and GENCODE M27, respectively.
- Retirement of gene names derived from BAC clones.

[More release news](#) on our blog

Ensembl Rapid Release

New assemblies with gene and protein annotation every two weeks.

Note: species that already exist on this site will continue to be updated with the full range of annotations.

The Ensembl Rapid Release website provides annotation for recently produced, publicly available vertebrate and non-vertebrate genomes from biodiversity initiatives such as Darwin Tree of Life, the Vertebrate Genomes Project and the Earth BioGenome Project.

[Rapid Release news](#) on our blog

Other news from our blog

- 23 Aug 2021: [Job: Outreach Officer](#)
- 16 Aug 2021: [Free Ensembl Browser and REST API virtual workshops in September](#)
- 10 Aug 2021: [Job: Bioinformatician](#)

Compare genes across species



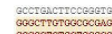
Find SNPs and other variants for my gene



Gene expression in different tissues



Retrieve gene sequence



Find a Data Display



Use my own data in Ensembl



<https://www.ensembl.org/index.html>

C'est un projet conjoint de l'European Bioinformatics Institute (EBI) et du Wellcome Trust Sanger Institute dont l'idée centrale est d'organiser de vastes champs d'information biologique autour de séquences génomiques.

Diagramme des étapes d'annotation de CDS dans Ensembl

Targetted stage: species-specific proteins aligned to the genome to produce transcript models

Similarity stage: proteins from closely-related species aligned to the genome to produce transcript models

Species-specific cDNA and ESTs aligned to the genome to produce transcript models and add UTRs to protein-derived models

RNA-seq reads aligned to the genome to produce transcript models and add UTRs to protein-derived models

Multi-transcript genes produced by merging where exons match

species-specific proteins are aligned to the genome and Genewise and/or Exonerate is used to build a transcript structure for the protein on the genome.

proteins from closely related species are used to build transcript structure in regions where a Targeted transcript structure is absent. For species with fewer species-specific protein sequences the Similarity stage plays an important role in predicting gene structures

Alignment of species-specific cDNA and EST sequences to the genome. Where cDNA alignments overlap transcripts predicted in the preceding stages, any non-translated region from the cDNA is spliced onto the transcript prediction as UTR.

Where available, RNA-seq reads are aligned to the genome and build RNA-seq-based transcript models from these data. These models may be used to add new genes or transcript isoforms into the gene set, and to add UTR to protein-coding models.

The final set: merging identical transcripts built from different proteins sequences to produce multi-transcript gene predictions, each with a non-redundant set of transcripts models. Where transcripts have identical exons, these are combined to form a gene

Artemis: genome browser and annotation tool




Artemis Entry Edit: subtilin.seq

File Entries Select View Goto Edit Create Write Run Graph Display

Selected feature: bases 1326 amino acids 441 CDS (/note="none")

Entry: subtilin.seq ORFS_300+ Patscan_RBS.gff Patscan_Promoter.gff Patscan_term.gff

>>



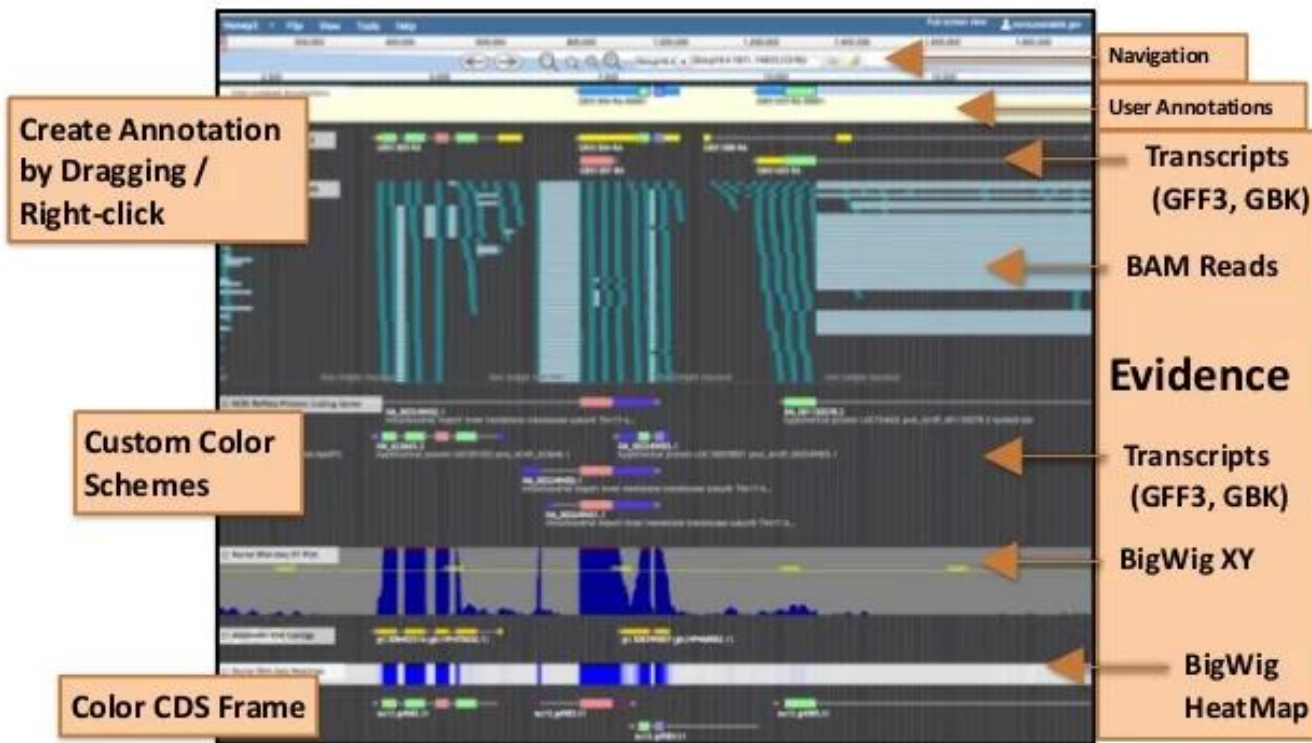
Gene annotation: subtilin.seq

Feature	Start	End	Notes
CDS	5395	6720	none
CDS	10946	12349	none
RBS	97	114	
RBS	193	210	
RBS	439	451	
RBS	478	497	
RBS	1367	1384	c

https://www.sanger.ac.uk/tool/artemis/

Apollo: genome annotation editor

Apollo Genome Editor



Référence : Dunn NA, Unni DR, Diesh C, Munoz-Torres M, Harris NL, Yao E, et al. (2019) Apollo: Democratizing genome annotation. PLoS Comput Biol 15(2): e1006790. <https://doi.org/10.1371/journal.pcbi.1006790>

Documentation : <https://genomearchitect.readthedocs.io/en/latest/>

HMM et construction de profil

Alignement multiple

Construction de profils

Problème : déterminer si une séquence appartient à une famille connue.

L'alignement de deux séquences capture la relation entre ces deux séquences alors qu'un alignement multiple montre comment les séquences d'une même famille sont apparentées les unes aux autres, permettant d'identifier des régions plus conservées que d'autres. Quand on recherche si une nouvelle séquence appartient à la famille, il serait souhaitable de détecter si ces régions plus conservées sont présentes. Comment obtenir et utiliser ces informations.

Ceci est réalisé en construisant des profils.

Existence de différentes méthodes basées sur l'utilisation de méthodes probabilistes différentes

En commun, matrice dérivée d'un alignement multiple

Quelques méthodes

- Modèle de Gribskov (1987, *Proc. Natl. Acad. Sci. USA*, **84**, 4355-4358)
- PsiBlast (Altschul *et al.* (1997), *NAR*, **25**, 3389-3402)
- Gibbs sampling/algorithm génétique : PROBE (Neuwald *et al.* (1997), *NAR*, **25**, 1665-1677)
- Modèle de Markov caché (HMM, suite HMMER) (<http://hmmer.wustl.edu>)

Construction de profils : modèle de Gribskov



A partir d'un alignement multiple on construit une matrice à 21 colonnes (une par acide aminé plus une pour la pondération des indels) et N lignes (N longueur de l'alignement).

Calcul du score de l'acide aminé a à la position p :

$$M(p, a) = \sum_{b=1}^{20} W(b, p) Y(a, b)$$

avec $Y(a, b)$ poids dans la matrice de substitution pour les acides aminés a et b (PAM250) et $W(b, p)$ poids de l'apparition de l'acide aminé b à la position p

$$W(b, p) = \frac{n(b, p)}{N_R}$$

$n(b, p)$ nombre observé de b à la position p
 N_R nombre de séquences

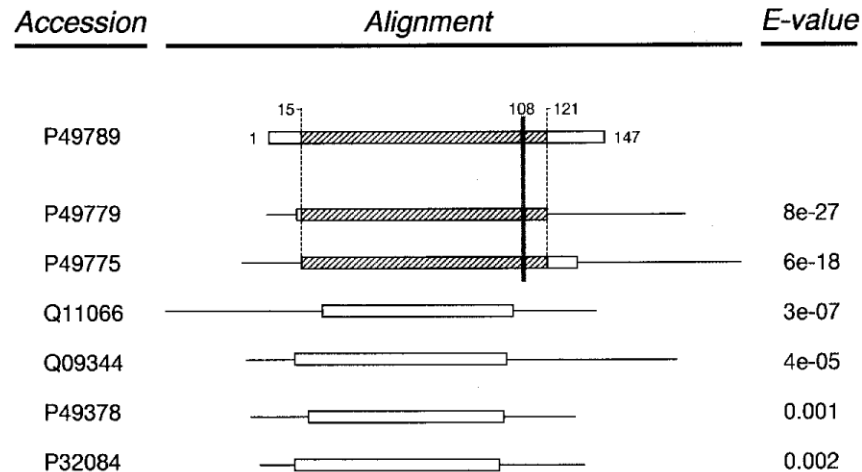
Exemple : position i donnée dans un alignement (4 séquences) : **L L S P**

La valeur $M(i, A)$ sera donnée par :

$$M(i, A) = \frac{1}{4} Y(A, S) + \frac{1}{4} Y(A, P) + \frac{1}{2} Y(A, L)$$

Construction de profils : PsiBlast

- recherche avec une séquence sonde des séquences similaires dans une base de données
- parmi les séquences identifiées, celles dont l'alignement à une *e-value* inférieure à un seuil donné sont utilisées pour construire une matrice de scores position-spécifique
- la sonde est utilisée comme référence pour construire l'alignement multiple (identifier les segments des différentes séquences de la banque s'alignant sur la sonde)



Pour simplifier, le profil aura la longueur du segment de la séquence sonde (délétions dans la sonde pas prises en compte), mais toutes les positions ne seront pas forcément représentées par le même nombre de séquences (ex: calcul pour la position 108, les 3 séquences grisées).

Des poids sont également donnés aux séquences en fonction de leur pourcentage de similarité (plus faible pour séquences proches).

Construction de profils : PsiBlast

- Le profil ainsi dérivé est utilisé dans une nouvelle recherche sur la banque dans le but d'identifier de nouvelles séquences plus faiblement conservées en séquence.
- Même principe, les séquences présentant un alignement avec une *e-value* inférieure à un seuil donné sont conservées pour établir une nouvelle matrice de scores position-spécifique.

On arrête le processus d'itération (recherche/construction du profil) quand il y a convergence, *i.e.*, quand une nouvelle itération ne permet pas d'identifier de nouvelles séquences (le groupe de séquences est stable) ou si le nombre maximum d'itération fixé par l'utilisateur est atteint.

- Autres utilisations :
- On peut utiliser le profil dérivé par PsiBlast pour une nouvelle recherche sur une autre banque de données par exemple
- On peut aussi utiliser PsiBlast en lui fournissant un profil pré-calculé (dérivé d'un alignement multiple) et une séquence sonde

Construction de profils : HMM

Alignement sans indels

Quand on considère un alignement multiple on s'aperçoit que les indels ont tendances à se localiser dans une même région des séquences permettant de définir des blocs alignés sans indels.

Dans ce cas, la probabilité qu'une nouvelle séquence x suive le modèle est donnée par :

$$P(x | M) = \prod_{i=1}^L e_i(x_i)$$

Avec L longueur du bloc, x_i l'acide aminé présent à la position i de la séquence x et $e_i(x_i)$ la probabilité d'observer l'acide aminé x_i à la position i

En général, on est plus intéressé par le rapport de cette probabilité sur la probabilité de x suivant un modèle aléatoire, soit :

$$S = \sum_{i=1}^L \log \frac{e_i(x_i)}{q_{x_i}}$$

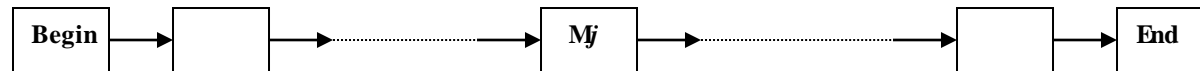
Une telle approche est connue sous le nom de position specific scoring matrix (PSSM). Une PSSM contient les valeurs :

$$\log \frac{e_i(x_i)}{q_{x_i}}$$

Une PSSM peut être utilisée pour chercher un match dans une séquence x de longueur N en calculant S_j pour chaque point de départ j dans x , j allant de 1 à $N-L+1$.

Construction de profils : HMM

Une PSSM peut être vue comme un HMM avec une structure répétitive d'états, mais différentes probabilités d'émission à chaque position. Cela fournira un modèle probabiliste complet pour une famille de séquences. Les probabilités de transition entre états seront alors de 1 (pas de choix).



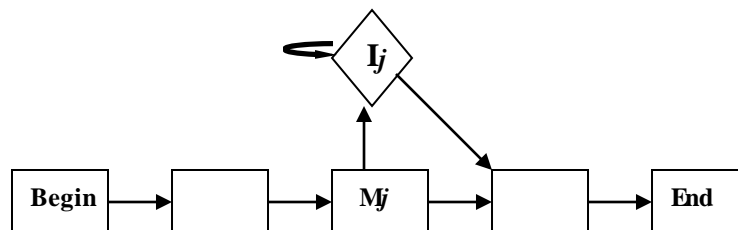
Les états sont appelés match M , et la probabilité d'émission de l'acide aminé a à l'état M_i est appelée $e_{M_i}(a)$

Ceci ne permet cependant pas de capturer toutes les informations d'un alignement multiple et notamment la localisation des indels. Il nous faut donc prendre en compte ces informations.

Un alignement multiple peut être décrit par trois états :

- match M
- délétion D
- insertion I

Insertion, nouveaux états I_i utilisé pour représenter une insertion après le résidu dans l'état M_i



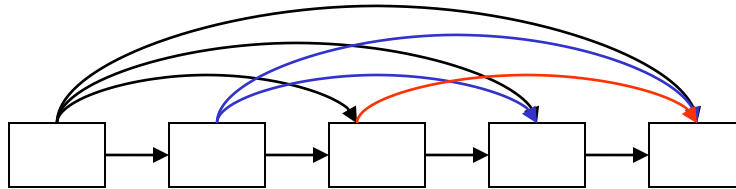
Probabilité d'émission pour l'état I_i : $e_{I_i}(a)$

En général, considérée comme distribution du bruit de fond q_a .

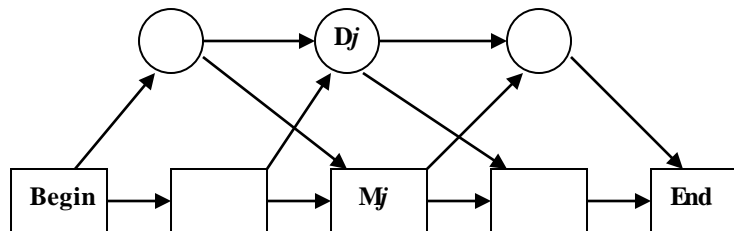
Construction de profils : HMM

Représentation des délétions.

Une délétion dans la séquence x correspond à un segment de séquence présent dans l'alignement multiple mais absent dans x . Elle pourrait donc être représentée comme un saut entre deux états non voisins :

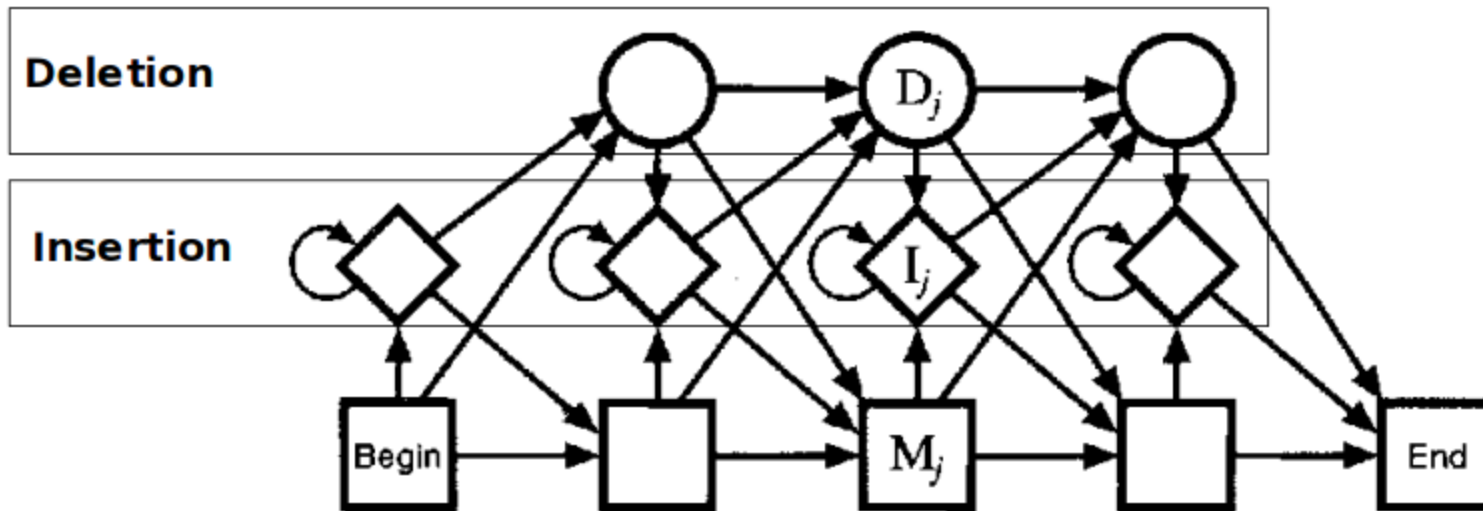


Cependant, si nous voulons permettre de longues délétions dans un modèle cela va générer un très grand nombre de transitions. Les délétions vont donc être décrites par des états D_j , appelés états silencieux car ils n'émettent aucun symbole (c'est aussi le cas des états Begin et End). Comme les états silencieux n'émettent pas de symbole, on peut donc aller de n'importe quel état « réel » à n'importe quel autre état « réel » postérieur sans émettre de symbole.



Construction de profils : HMM

HMM complet pour modéliser un alignement multiple

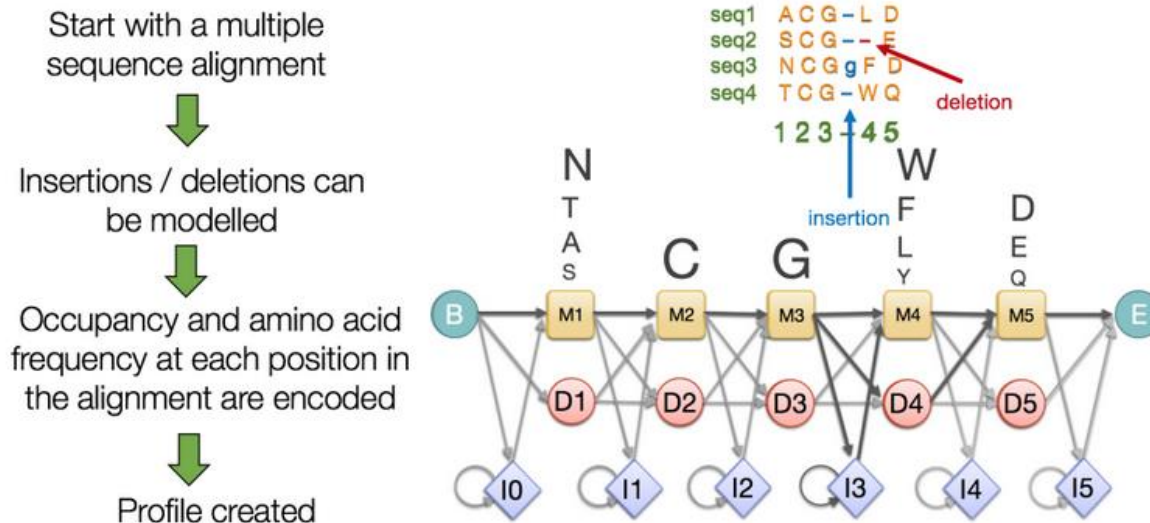


Les transitions entre les états d'insertion et de délétion ont été rajoutés même si en fait ils sont très improbables. Ne pas les prendre en compte a un effet négligeable sur le calcul du score, mais cela peut créer des problèmes lors de la construction du modèle.

Profils HMM : conclusion

Les profils HMM sont des modèles probabilistes qui intègrent les changements évolutifs survenus dans un ensemble de séquences apparentées (c'est-à-dire un alignement de séquences multiples). Pour cela :

- ils capturent des informations spécifiques à la position sur le degré de conservation de chaque acide aminé dans chaque colonne de l'alignement.
- Ils saisissent aussi des informations importantes comme le degré auquel les gaps et les insertions se sont produites.
- Contrairement à d'autres algorithmes de détection d'homologie de séquence, les profils HMM utilisent des pénalités de gap et des probabilités de substitution dépendant de la position, qui reflètent mieux la réalité biologique



Sachant maintenant modéliser un alignement multiple par un HMM, comment estimer les paramètres de celui-ci pour représenter une famille de séquences.

Dans un premier temps, il faut décider de la longueur du modèle, c'est à dire dans l'alignement multiple, quelles sont les positions qui vont être assignées à des états match.

Prenons l'exemple suivant (extrait de Durbin et al. (1998), *Biological sequence analysis*, Cambridge University Press) :

```
Seq1    VGA--HAGEY
Seq2    V----NVDEV
Seq3    VEA--DVAGH
Seq4    VKG-----D
Seq5    VYS--TYETS
Seq6    FNA--NIPKH
Seq7    IAGADNGAGV
      ***  *****
```

Dans cet alignement, les positions repérées par des * vont être modélisées comme des états match, soit un modèle de longueur 8. Une règle heuristique simple qui marche bien, considérer comme des insertions, les positions de l'alignement pour lesquelles plus de 50% des séquences présentent un gap. Il existe cependant des techniques plus sophistiquées (cf Durbin et al.).

Pour estimer les probabilités de transition et d'émission, nous utiliserons les formules vues précédemment :

$$a_{kl} = \frac{A_{kl} + r_{kl}}{\sum_{l'} A_{kl'} + r_{kl'}} \quad \text{et} \quad e_k(b) = \frac{E_k(b) + r_k(b)}{\sum_{b'} E_k(b') + r_k(b')}$$

On va appliquer ici la méthode la plus simple pour les pseudocounts, la méthode de Laplace qui consiste à ajouter 1 à chaque fréquence.

Si on considère l'état M_1 :

Probabilités d'émission

$$e_{M_1}(V) = \frac{5+1}{17+6+2+2} = \frac{6}{27} \quad \begin{array}{l} 17 \text{ aa ne sont pas présents} \\ 5V, 1F \text{ et } 1I \end{array}$$

$$e_{M_1}(F) = e_{M_1}(I) = \frac{2}{27} \quad \text{et} \quad e_{M_1}(a) = \frac{1}{27} \quad \begin{array}{l} \text{Pour tout acide aminé } a \\ \text{différent de F,V,I} \end{array}$$

Seq1	VGA--HAGEY
Seq2	V----NVDEV
Seq3	VEA--DVAGH
Seq4	VKG-----D
Seq5	VYS--TYETS
Seq6	FNA--NIPKH
Seq7	IAGADNGAGV
	*** *****

Probabilités de transition

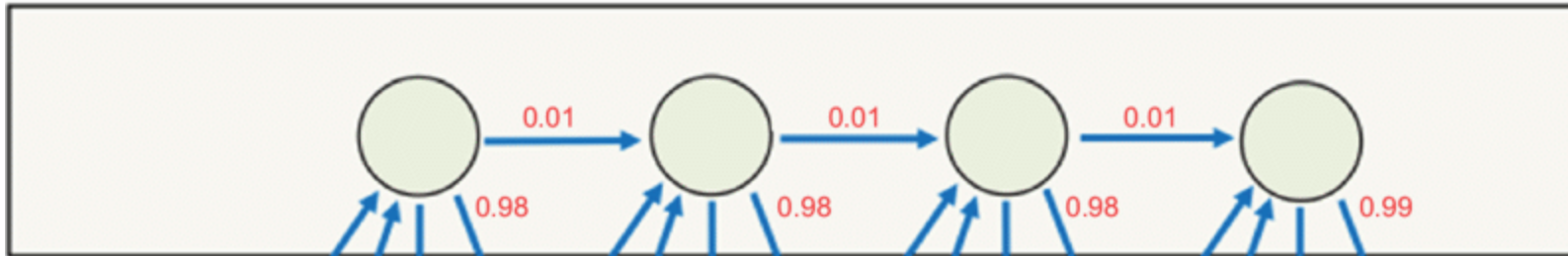
$$a_{M_1M_2} = \frac{6+1}{7+2+1} = \frac{7}{10} \quad \begin{array}{l} 6 \text{ transitions vers un match, } 1 \text{ transition vers une} \\ \text{délétion, } 0 \text{ transition vers une insertion} \end{array}$$

$$a_{M_1D_2} = \frac{2}{10} \quad \text{et} \quad a_{M_1I_2} = \frac{1}{10}$$

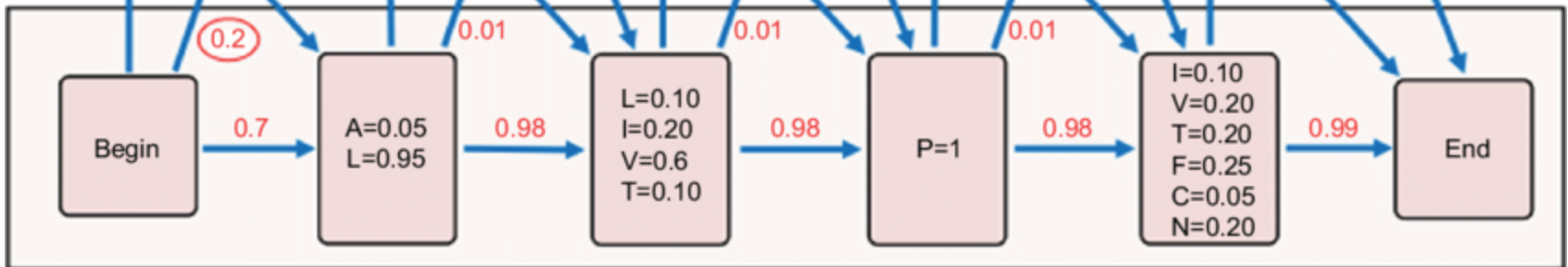
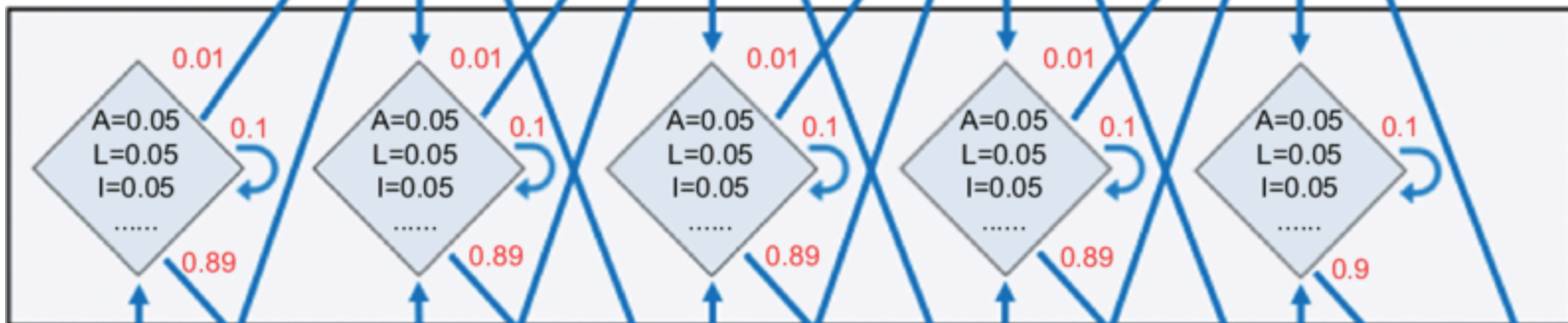
Profils HMM : estimation des paramètres

Exemple :

Deletion states



Insertion states



Match states

Recherche avec des profils HMM

Raison principale de construire des profils HMM  Recherche si une séquence appartient à la famille

Dans un premier temps, on suppose que l'on utilise un alignement global. On va rechercher si notre séquence possède des matches significatifs avec notre profil HMM.

Deux possibilités pour obtenir la valeur du match :

- utilisation des équations Viterbi pour trouver l'alignement le plus probable π^* de notre séquence x et sa probabilité $P(x, \pi^* | M)$,
- utilisation des équations forward pour calculer la probabilité complète de x sommée sur tous les chemins possibles $P(x | M)$.

Dans chaque cas, pour évaluer les matches potentiels, on va considérer le log-odds ratio de la probabilité résultante sur celle obtenue par cette même séquence x mais dans le cas d'un modèle aléatoire

$$P(x | R) = \prod_i q_{x_i}$$

Recherche avec des profils HMM

Algorithme de Viterbi établi spécifiquement pour les profils HMM avec le calcul direct des log-odds values. On va donc travailler dans l'espace des logs.

$$V_j^M(i) = \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}M_j} \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}M_j} \\ V_{j-1}^D(i-1) + \log a_{D_{j-1}M_j} \end{cases}$$

$$V_j^I(i) = \log \frac{e_{I_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_j^M(i-1) + \log a_{M_jI_j} \\ V_j^I(i-1) + \log a_{I_jI_j} \\ V_j^D(i-1) + \log a_{D_jI_j} \end{cases}$$

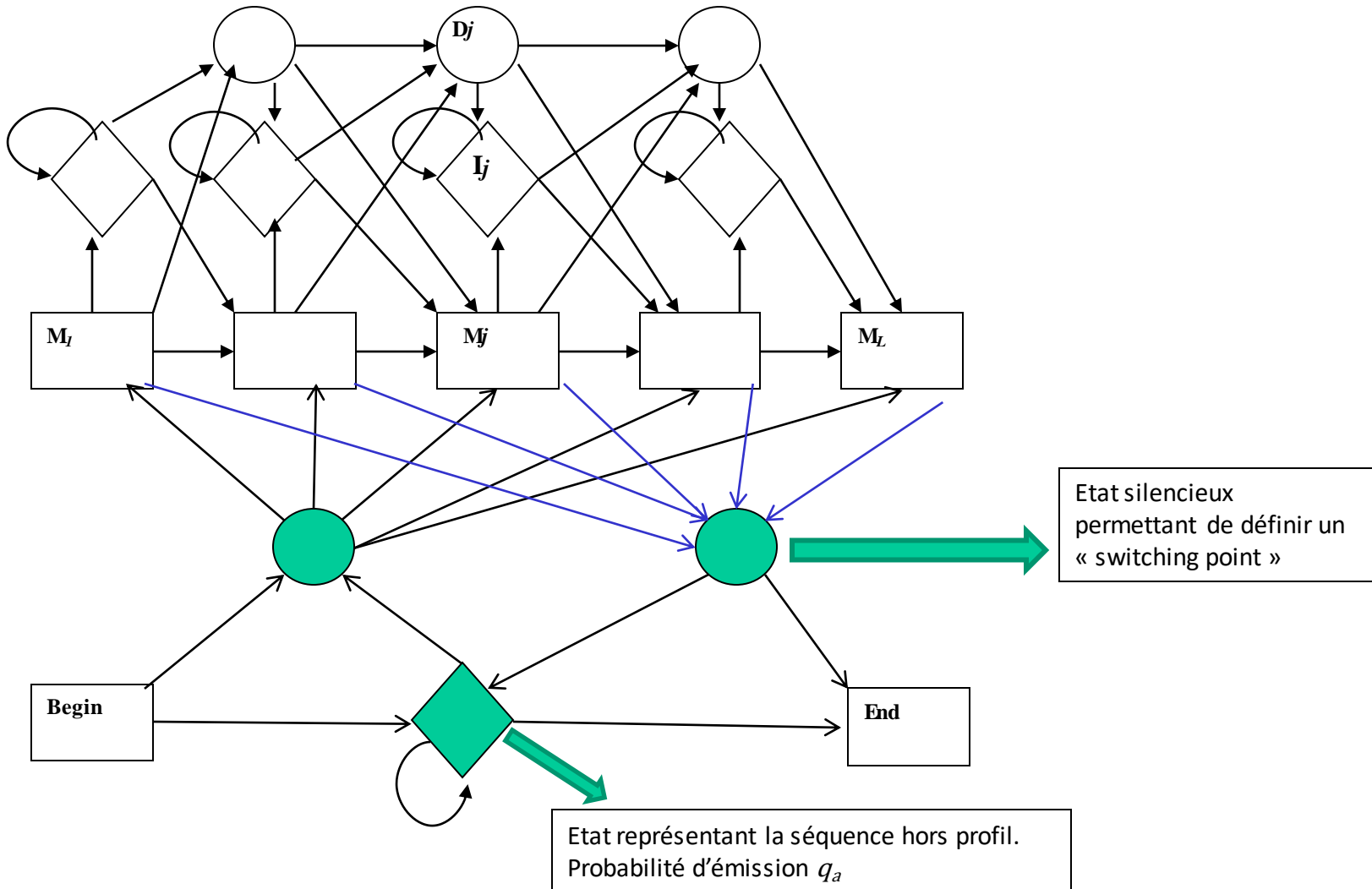
$$V_j^D(i) = \max \begin{cases} V_{j-1}^M(i) + \log a_{M_{j-1}D_j} \\ V_{j-1}^I(i) + \log a_{I_{j-1}D_j} \\ V_{j-1}^D(i) + \log a_{D_{j-1}D_j} \end{cases}$$

Equations générales mais on peut ne pas avoir de score d'émission dans l'équation $V_j^I(i)$

Si on suppose que la distribution des émissions dans les états I est la même que celle du bruit de fond ($\log 1 = 0$). De même, les termes de transitions $D \rightarrow I$ et $I \rightarrow D$ peuvent ne pas être présents.

Modèle pour alignement local

Permet en plus des matches répétés à des sous-parties du profil



La suite HMMER

Le package HMMER, version courante HMMER3, correspond à une collection de programmes qui réalisent différentes fonctions en utilisant des HMM, dont :

- La construction de profils HMM
 - *hmmbuild* - construct profile HMMs from multiple sequence alignments
- La recherche de similarité avec des profils
 - *hmmsearch* - search protein sequences against a profile HMM database
 - *hmmsearch* - search profile HMMs against a sequence database
- La recherche de similarité aussi avec des séquences
 - *phmmer* - search protein sequences against a protein database (like BlastP)
 - *jackhmmer* - iteratively search sequences against a protein database (like PsiBlast)
 - *nhmmer* - search DNA/RNA queries against a DNA/RNA sequence database
 - *nhmmscan* - search nucleotide sequences against a nucleotide profile
- Autres fonctions dont
 - *hmmalign* - align sequences to a profile HMM

Développé en C par Sean Eddy, Travis Wheeler et HMMER development team

Déposé dans GitHub : <https://github.com/EddyRivasLab/hmmer>

Site web : <http://hmmer.org/>