HAL
open science

Calculs d'inférence dans les arbres phylogénétiques

Laurent Guéguen

To cite this version:

Laurent Guéguen. Calculs d'inférence dans les arbres phylogénétiques. Gilles Didier; Stéphane Guindon. Modèles et méthodes pour l'évolution biologique, ISTE Group, pp.177-202, 2022, 9781789480696. 10.51926/ISTE.9069.ch7 . hal-04059719

HAL Id: hal-04059719

https://hal.science/hal-04059719v1

Submitted on 7 Apr 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Calculs d'inférence dans les arbres phylogénétiques

Laurent Guéguen

LBBE, Claude Bernard University Lyon 1, Villeurbanne, France
Swedish Collegium for Advanced Study, Uppsala, Sweden

Chapitre 7 de « Modèles et Méthodes pour l'évolution biologique »
Gilles Didier , Stéphane Guindon, ISTE Edition. Chapitre 7, pp.177-202, 2022.
doi :10.51926/ISTE.9069.ch7

Table des matières

1 Inférences **2**
1.1 Maximum de parcimonie 3
1.2 Vraisemblance 3

2 Programmation dynamique **4**
2.1 Réduction de la complexité 7
2.2 Gestion de la racine 8

3 Maximum de parcimonie **10**
3.1 Inférence ancestrale 12

4 Vraisemblance **12**
4.1 Gestion de la racine 14
4.2 Calcul aux nœuds 14
4.3 Maximisation - Dérivation 15
4.3.1 Dérivation numérique 16
4.3.2 Dérivation analytique 17
4.4 Inférence ancestrale 17

1 Inférences

L'objectif de l'**inférence phylogénétique** est d'expliquer, à partir de séquences homologues actuelles, le processus évolutif qui les a générées. Généralement, on considère que ce processus s'est déroulé selon une dynamique arborescente.

Il est possible que la topologie de cet arbre soit inconnue, et doive être inférée. Cela peut-être fait grâce à des méthodes de distances entre séquences (voir chapitre ...), ou lors de l'estimation du processus évolutif. Dans ce chapitre, pour poser les méthodes de calcul, nous considérerons que cette topologie est donnée.

Comme cette étude fait appel à des outils mathématiques et informatiques, il est nécessaire de modéliser les processus évolutifs. Usuellement, il existe deux types de modélisation : le maximum de parcimonie et la vraisemblance. Chercher le maximum de parcimonie signifie poser l'hypothèse que les données sont issues d'un scénario qui est le plus court possible, en terme de nombre d'événements, ou plus généralement le moins coûteux possible, si des coûts sont associés aux événements. Calculer la vraisemblance de ces données s'inscrit dans une modélisation probabiliste (voir chapitre...), et il s'agit alors de calculer la probabilité des données d'après un certain modèle évolutif. On pourra chercher à maximiser cette vraisemblance, ou alors à l'utiliser dans une démarche bayésienne (voir chapitre ...).

Au-delà du calcul d'un score à partir de séquences, on voudra aussi utiliser cette démarche pour reconstruire l'histoire qui a abouti à ces séquences : quelles étaient les séquences ancestrales, par quelles étapes ont-elles évolué vers les séquences actuelles ? Ce que l'on appelle **inférence ancestrale**. Nous verrons dans ce chapitre par quels calculs on peut effectuer l'inférence phylogénétique puis l'inférence ancestrale.

Prenons un exemple simple : les données sont trois séquences $\mathcal{D} = \{\mathtt{S}, \mathtt{T}, \mathtt{U}\}$ issues d'un processus évolutif le long d'un arbre $\mathcal{A}$ (Fig. 1). Comme ce processus est inconnu, pour calculer un score sur $\mathcal{D}$ et $\mathcal{A}$ il faut considérer l'ensemble des processus possibles ($\mathbb{P}$) qui ont pu engendrer $\mathcal{D}$, et calculer un minimum ou une somme sur cet ensemble.

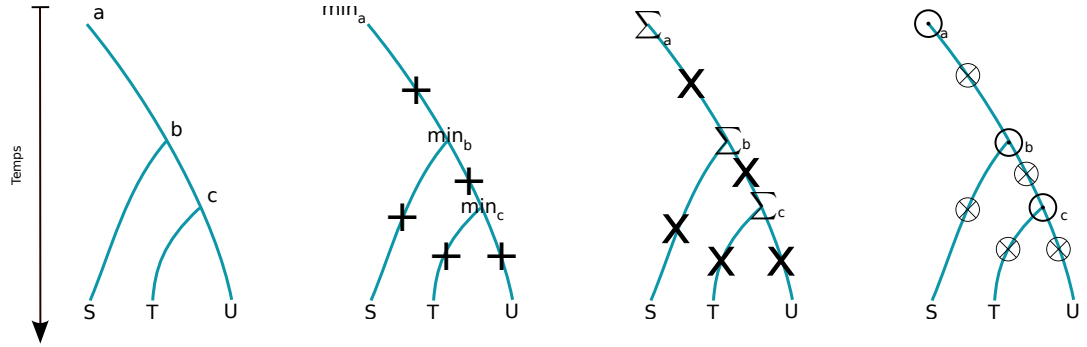

FIGURE 1 – Exemples de calculs le long d'un arbre. De gauche à droite : maximum de parcimonie, vraisemblance, calcul formel.

Par souci de généralité, nous dirons qu'un processus permet de passer d'**état**

en **état**, et qu'un changement d'état est une **substitution**. Un état peut être une séquence, un site, ou un ensemble de sites. On note $\mathbb{E}$ l'ensemble des états possibles.

1.1 Maximum de parcimonie

Le maximum de parcimonie, $M(\mathcal{D})$, est le coût minimum sur tous les processus : $M(\mathcal{D}) = min_{p\in\mathbb{P}}\text{cout}(\mathcal{D}|p,\mathcal{A}) := \text{mcout}(\mathcal{D}|\mathcal{A})$

Pour simplifier, on note **mcout** le coût minimum sur l'ensemble des processus.

Pour ce calcul, nous considérons que l'**évolution n'a pas de mémoire**. Le coût de passage d'un état a vers un état b dépend de a et b, mais a étant donné, ce coût ne dépend pas des états dont a est issu. Ainsi, pour calculer le coût le long d'un processus il suffit de calculer les coûts de passage indépendants entre tous les états successifs de ce processus.

Si on note $a, b, c \in \mathbb{E}$ les états à la racine et aux nœuds de l'arbre, avec $a \to b$ le passage de a à b :

$$\begin{aligned}
M(\mathcal{D}) &= \min_{a,b,c}[\text{mcout}(\mathcal{D}|a,b,c,\mathcal{A})] \\
&= \min_{a,b,c}\Big[\text{mcout}(a\to b) + \Big(\text{mcout}(b\to \mathtt{S}) + \text{mcout}(b\to c) \\
&\qquad + \big(\text{mcout}(c\to \mathtt{T}) + \text{mcout}(c\to \mathtt{U})\big)\Big)\Big] \\
&= \min_{a}\Big[\min_{b}\text{mcout}(a\to b) + \Big(\text{mcout}(b\to \mathtt{S}) + \min_{c}\text{mcout}(b\to c) \\
&\qquad + \big(\text{mcout}(c\to \mathtt{T}) + \text{mcout}(c\to \mathtt{U})\big)\Big)\Big]
\end{aligned}$$

On voit qu'il est possible de séparer les différents minimums le long de l'arbre : le minimum sur a ne dépend pas des c, car il y a le nœud b entre eux. Cela est dû à la distributivité $\min(a+b, a+c) = a + \min(b,c)$.

Sur chaque branche, il faut calculer le coût minimal, sur l'ensemble des processus, du passage entre les deux états qui sont aux extrémités de la branche. Dans l'absolu, cela nécessite de considérer explicitement tous les états possibles qui sont apparus le long de la branche, ce qui est prohibitif, surtout quand la branche est longue. Au lieu de cela, on va considérer que l'**on peut calculer directement le coût minimum sur une branche à partir des deux états qui sont aux extrémités de cette branche**, et que l'inconnu est maintenant réduit à l'ensemble de ces états aux nœuds. Ce qui permet de simplifier la fonction de coût :

$\forall a, b \in \mathbb{E}, \text{mcout}(a\to b) = \text{cout}(a,b)$

Pour maintenir la cohérence de la modélisation, cette fonction de coût associée aux couples d'états aux extrémités des branches devra toujours être associée à un minimum sur l'ensemble des processus.

1.2 Vraisemblance

Considérons maintenant le calcul de la vraisemblance, en notant P la probabilité. La vraisemblance des données $P(\mathcal{D})$ est la somme des vraisemblances de $\mathcal{D}$ selon

tous les processus possibles. En prenant l'exemple de la Figure 1 :

$$\begin{aligned} P(\mathcal{D}) &= \sum_{p\in\mathbb{P}} P(\mathcal{D}|p)P(p) \\ &= \sum_{p\in\mathbb{P}} \sum_{a,b,c\in\mathbb{E}} P(\mathcal{D}|a,b,c,p)P(a,b,c|p)P(p) \end{aligned}$$

En posant aussi que l'évolution n'a pas de mémoire, et que chaque branche a une histoire indépendante, un processus sur un arbre peut être découpé en processus indépendants sur les différentes branches[1] :

$$\begin{aligned} P(\mathcal{D}) = \; & \sum_{a,b,c\in\mathbb{E}} (\sum_p P(b \to \mathtt{S}|p)P(p))(\sum_p P(c \to \mathtt{T}|p)P(p)) \\ & \times(\sum_p P(c \to U|p)P(p)) \\ & \times(\sum_p P(a|p)P(p))(\sum_p P(a \to b|p)P(p))(\sum_p P(b \to c|p)P(p)) \end{aligned}$$

Par définition, la probabilité de transition entre deux séquences a et b est la somme des probabilités sur l'ensemble des processus qui mènent de a à b, si bien que, par exemple : $\sum_{p\in\mathbb{P}} P(a \to b|p)P(p) = P(b|a)$:

$$\begin{aligned} P(\mathcal{D}) &= \sum_{a,b,c} \Big[P(a) \times P(b|a) \times \Big(P(\mathtt{S}|b) \times P(c|b) \times \big(P(\mathtt{T}|c) \times P(\mathtt{U}|c) \big) \Big) \Big] \\ &= \sum_a P(a) \Big[\sum_b P(b|a) \times \Big(P(\mathtt{S}|b) \times \sum_c P(c|b) \times \big(P(\mathtt{T}|c) \times P(\mathtt{U}|c) \big) \Big) \Big] \end{aligned}$$

Ici encore, il est possible de séparer les différentes sommes le long de l'arbre. La somme sur a ne dépend pas de c, car il y a le nœud portant b entre eux. Cela est dû à la distributivité : $a \times b + a \times c = a \times (b + c)$.

2 Programmation dynamique

La méthode pour le calcul du score étant la même pour les deux approches, notons-la d'une façon plus formelle, pour se concentrer sur l'algorithme sans se soucier du calcul lui-même. On identifie les opérateurs de calcul sur l'arbre, ainsi que l'opérateur sur l'ensemble des histoires. Indépendamment des histoires, à chaque transition entre deux états est associé un coût σ, et éventuellement un coût sur les états à la racine π. Le lien entre ces opérateurs est décrit dans le tableau suivant :

	Type de calcul		
	Formel	Parcimonie	Vraisemblance
nœud	$\oplus$	min	$\sum$
nœud	$\odot$	$\sum$	$\times$
branche	σ	cout	P
racine	π	1	P
branche	$M\otimes$	min cout $\sum$	$\sum P\times$

Les calculs reviennent à calculer un score global S en fonction des scores σ sur les branches :

$$S(\mathcal{D}) = \bigoplus_{a\in\mathbb{E}} \pi(a) \Big[\bigoplus_{b\in\mathbb{E}} \sigma(a,b) \odot \Big(\sigma(b,S) \odot \bigoplus_{c\in\mathbb{E}} \sigma(b,c) \odot \big(\sigma(c,\mathtt{T}) \odot \sigma(c,\mathtt{U}) \big) \Big) \Big]$$

1. Pour simplifier, on note toujours par p les processus sur chaque branche.

Comme le calcul de $\bigoplus_x$ en chaque nœud dépend du nœud dont il est issu (dans notre exemple le calcul sur c dépend de b), il n'est pas possible de résoudre l'opérateur $\bigoplus$ directement au nœud où il est appliqué. Il doit être calculé au début de la branche dont il est issu, après avoir calculé $\sigma\odot$ de cette branche pour tous les états possibles. Pour cette raison, le calcul va s'effectuer des feuilles vers la racine, en remontant l'arbre, et il sera nécessaire dans cette remontée de garder en mémoire les valeurs calculées pour l'ensemble des états. Ces valeurs seront stockées dans des vecteurs v, et le calcul sera une suite d'actions sur ces vecteurs.

Construisons donc les opérateurs qui seront appliqués aux vecteurs.

Sur les branches :

Sur chaque branche interne, on a toujours la suite $\bigoplus\sigma\odot$, on va donc regrouper ces opérateurs, et nous notons par le produit $M\otimes$ leur action :
$M \otimes v = (\bigoplus_{x\in\mathbb{E}} \sigma(x,y) \odot v(y))_{y\in\mathbb{E}}$.

Le résultat de ce produit est un nouveau vecteur sur l'ensemble des états possibles, et cela permet de remonter les branches de l'arbre.

En ce qui concerne les branches terminales, il faut définir en chaque feuille un vecteur v pour lequel $M \otimes v$ égale le vecteur des coûts de transition vers la valeur observée (dans notre exemple, on veut que $M \otimes v_{\mathtt{U}} = \begin{pmatrix} \sigma(c_1,\mathtt{U}) \\ \vdots \\ \sigma(c_z,\mathtt{U}) \end{pmatrix}$).

Pour cela, le vecteur en chaque feuille est constitué de l'élément neutre de $\oplus$ (noté 0), excepté pour la valeur correspondant à l'état observé, qui sera l'élément neutre de $\otimes$ (noté 1). Rappelons que la propriété nécessaire pour que notre calcul fonctionne ainsi est la distributivité : $(a \odot b) \oplus (a \odot c) = a \odot (b \oplus c)$. Cela entraine que $\forall x \in E, x = x \odot (0 \oplus 1) = (x \odot 0) \oplus (x \odot 1) = (x \odot 0) \oplus x$ et donc $x \odot 0 = 0$: 0 est élément absorbant de $\odot$.

Si par exemple $\mathbb{E} = \{\mathtt{A},\ \mathtt{C},\ \mathtt{G},\ \mathtt{T}\}$, et en une feuille l'état $\mathtt{A}$ est observé, le calcul sur la branche terminale qui mène à cette feuille est :

$$M \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} (\sigma(\mathtt{A},\mathtt{A}) \odot 1) \oplus (\sigma(\mathtt{A},\mathtt{C}) \odot 0) \oplus (\sigma(\mathtt{A},\mathtt{G}) \odot 0) \oplus (\sigma(\mathtt{A},\mathtt{T}) \odot 0) \\ (\sigma(\mathtt{C},\mathtt{A}) \odot 1) \oplus (\sigma(\mathtt{C},\mathtt{C}) \odot 0) \oplus (\sigma(\mathtt{C},\mathtt{G}) \odot 0) \oplus (\sigma(\mathtt{C},\mathtt{T}) \odot 0) \\ (\sigma(\mathtt{G},\mathtt{A}) \odot 1) \oplus (\sigma(\mathtt{G},\mathtt{C}) \odot 0) \oplus (\sigma(\mathtt{G},\mathtt{G}) \odot 0) \oplus (\sigma(\mathtt{G},\mathtt{T}) \odot 0) \\ (\sigma(\mathtt{T},\mathtt{A}) \odot 1) \oplus (\sigma(\mathtt{T},\mathtt{C}) \odot 0) \oplus (\sigma(\mathtt{T},\mathtt{G}) \odot 0) \oplus (\sigma(\mathtt{T},\mathtt{T}) \odot 0) \end{pmatrix}$$
$$= \begin{pmatrix} \sigma(\mathtt{A},\mathtt{A}) \\ \sigma(\mathtt{C},\mathtt{A}) \\ \sigma(\mathtt{G},\mathtt{A}) \\ \sigma(\mathtt{T},\mathtt{A}) \end{pmatrix}$$

Ce qui correspond aux valeurs souhaitées en haut de la branche terminale qui mène à $\mathtt{A}$.

Sur les nœuds :

Après avoir établi le calcul des vecteurs le long des branches, définissons celui aux nœuds. À chaque nœud, l'opérateur $\odot$ est appliqué entre des valeurs qui concernent les mêmes états (par exemple $\sigma(c,T)\odot\sigma(c,\mathtt{U})$ pour chaque état c), si bien qu'il suffit d'étendre $\odot$ au produit terme à terme des vecteurs :

$v \odot w = (v_i \odot w_i)_{i \in \mathbb{E}}$.

Le résultat de ce produit est un nouveau vecteur qui permet de remonter les nœuds de l'arbre.

À la racine :

Les valeurs du vecteur radical doivent être cumulées pour obtenir une valeur réelle. Ce cumul étant fait sur l'ensemble des processus possibles, il est fait de la même manière que les opérations aux nœuds, par l'opérateur $\odot$. Pour permettre un poids aux différents états à la racine, on intègre un coût π, et on définit l'opérateur à la racine :

$\pi \otimes v = \bigoplus_{x \in \mathbb{E}} \pi(x) \odot v(x)$.

Sur l'arbre :

Pouvant remonter les branches et les nœuds de l'arbre, on peut calculer le score le long de l'arbre jusqu'à la racine. Notons v_n le vecteur des valeurs au nœud n, et $v_{n \to m}$ le vecteur des valeurs en tête de la branche $n \to m$. Dans notre exemple (voir Fig. 2), le calcul du score est :

$$S(\mathcal{D}) = \pi \otimes \left[M_{6\to5} \otimes \left(M_{5\to4} \otimes v_4 \odot \left(M_{5\to3} \otimes \left((M_{3\to2} \otimes v_2) \odot (M_{3\to1} \otimes v_1) \right) \right) \right) \right]$$

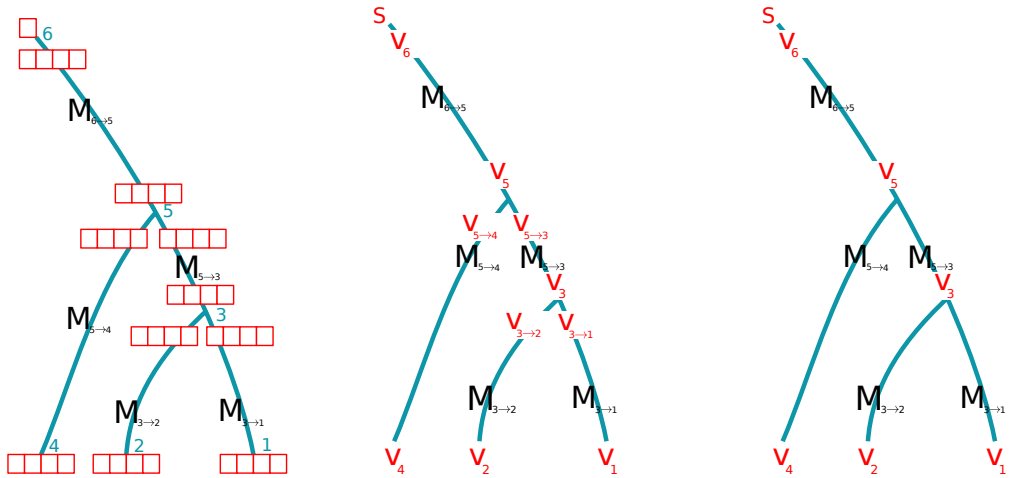

FIGURE 2 – Calcul vectoriel du score. À gauche, représentation des vecteurs impliqués dans le calcul. Au milieu, calcul complet avec les noms des vecteurs aux nœuds et en haut des branches. À droite, calcul simplifié avec les noms des vecteurs aux nœuds.

Si on note $e(n)$ l'ensemble des nœuds qui sont directement sous le nœud n, les vecteurs sont calculés successivement ainsi :

$$\begin{cases} v_n & = \bigodot_{m \in e(n)} v_{n \to m} \\ v_{n \to m} & = M_{n \to m} \otimes v_m \\ \\ S(\mathcal{D}) & = \pi \otimes v_{\text{racine}} \end{cases}$$

Toujours dans notre exemple, les différentes étapes du calcul sont donc :

$$
\begin{aligned}
S(\mathcal{D}) &= \pi \otimes \left[M_{6\to5} \otimes \left((M_{5\to4} \otimes v_4) \odot \left(M_{5\to3} \otimes \left((M_{3\to2} \otimes v_2) \odot (M_{3\to1} \otimes v_1) \right) \right) \right) \right] \\
&= \pi \otimes \left[M_{6\to5} \otimes \left(v_{5\to4} \odot \left(M_{5\to3} \otimes \left(v_{3\to2} \odot v_{3\to1} \right) \right) \right) \right] \\
&= \pi \otimes \left[M_{6\to5} \otimes \left(v_{5\to4} \odot \left(M_{5\to3} \otimes v_3 \right) \right) \right] \\
&= \pi \otimes \left[M_{6\to5} \otimes \left(v_{5\to4} \odot v_{5\to3} \right) \right] \\
&= \pi \otimes \left[M_{6\to5} \otimes v_5 \right] \\
&= \pi \otimes v_6
\end{aligned}
$$

Comme il est effectué dans un arbre, ce calcul peut très facilement être implémenté de façon récursive en suivant la topologie de l'arbre.

Pour que la méthode soit moins coûteuse en mémoire, on peut enlever les vecteurs en haut des branches ($v_{n\to m}$) et regrouper les calculs aux nœuds :

$v_n = \bigodot_{m\in e(n)} M_{n\to m} \otimes v_m$

Dans notre exemple, les étapes sont alors :

$$
\begin{aligned}
S(\mathcal{D}) &= \pi \otimes \left[M_{6\to5} \otimes \left((M_{5\to4} \otimes v_4) \odot \left(M_{5\to3} \otimes \left((M_{3\to2} \otimes v_2) \odot (M_{3\to1} \otimes v_1) \right) \right) \right) \right] \\
&= \pi \otimes \left[M_{6\to5} \otimes \left((M_{5\to4} \otimes v_4) \odot \left(M_{5\to3} \otimes v_3 \right) \right) \right] \\
&= \pi \otimes \left[M_{6\to5} \otimes v_5 \right] \\
&= \pi \otimes v_6
\end{aligned}
$$

Cependant, la première implémentation peut être utile lorsque l'on a besoin de manipuler les scores en haut des branches (par exemple, si on veut tester le positionnement d'une branche (ou d'un clade) en différents endroits de l'arbre, on peut « promener »ce vecteur au nœud où on veut tester sa position, sans avoir à recalculer le score qui est sous cette branche). Aussi, dans le cas de la vraisemblance, les vecteurs $v_{n\to m}$ sont utiles pour le calcul aux nœuds (voir 4.2).

La méthode présentée ici peut s'appliquer à n'importe quel type d'opérateurs, du moment que la règle de distributivité entre $\odot$ et $\oplus$ est respectée. Aussi, pour que la démarche de découper l'arbre en branches soit cohérente, il faut le score σ soit du même type que l'opérateur $\oplus$, c'est-à-dire tous deux un miminum, une somme, etc. Ceci n'est pas toujours possible. Par exemple, lorsque les mécanismes évolutifs considérés sont trop complexes pour pouvoir être analytiquement cumulés sur une distance quelconque (transferts de gènes, réarrangements de génomes, ...), ou bien lorsque la fonction de coût ne se cumule pas simplement le long de la branche. Dans ce cas, chaque branche devra être découpée en petits bouts sur lesquels il sera supposé que zéro ou un événement a eu lieu, et les coûts de tous ces petits événements seron cumulés : ceci est très coûteux en temps et mémoire.

2.1 Réduction de la complexité

Comme $M\otimes$ est appliqué à tous les couples d'états en chaque branche, la complexité temporelle est linéaire avec le nombre de branches et quadratique avec le

nombre d'états[2]. Quels états considérer ? Si l'on considère les séquences sur toute leur longueur, le nombre d'états possibles croît de façon exponentielle avec cette longueur, et il devient prohibitif de manipuler l'ensemble de ces états.

L'approche commune pour diminuer le nombre d'états est de segmenter le processus évolutif sur les séquences en un ensemble de processus évolutifs indépendants sur des morceaux de ces séquences. Ces morceaux sont appelés **sites**. Dans la segmentation la plus fine, un site est une position de chaque séquence, le nombre d'états étant alors la taille de l'alphabet des séquences (nucléotides ou protéines). Dans le cas de séquences codantes, on aura aussi tendance à considérer les codons comme sites autonomes d'évolution. Grâce à l'hypothèse que tous les sites sont indépendants, les calculs dans l'arbre peuvent être effectués indépendamment sur chaque site, puis « simplement »cumulés le long des séquences, si bien que la complexité est désormais proportionnelle au nombre de sites multiplié par la taille de l'alphabet au carré. D'un point de vue modélisation, sur de tels sites les substitutions considérées sont généralement les substitutions au sens biologique (voir le chapitre sur les modèles de substitution), mais dans l'absolu rien n'empêche de considérer des opérateurs différents, du moment que l'indépendance entre sites est préservée.

Sur l'ensemble des données, on dispose alors de plusieurs processus indépendants, dont les scores devront être regroupés pour former le score global. Dans le calcul, c'est l'opérateur $\odot$ qui cumule les scores indépendants (d'une façon similaire, les calculs de par et d'autre des nœuds de l'arbre sont indépendants). Ainsi, si la donnée $\mathcal{D}$ est divisée en sites $\mathcal{D}_1, ..., \mathcal{D}_l$ considérées indépendants :

$S(\mathcal{D}) = \bigodot_{i \in <1,l>} S(\mathcal{D}_i)$

Au niveau de l'implémentation, en chaque nœud les valeurs seront stockées sous la forme de tableaux à deux dimensions « Nombre de sites $\times$ Nombre d'états ». Aussi, il est possible que les données en plusieurs sites soient semblables. Ceci sera d'autant plus fréquent que l'alphabet sera petit et les sites nombreux. Dans ce cas, il n'est pas nécessaire de gérer les calculs pour ces différents sites, mais simplement un seul d'entre eux, et le calcul de la valeur globale tiendra compte du nombre de copies de ce site.

Cependant, l'hypothèse de sites évoluant indépendamment peut être considérée comme trop forte pour l'étude de certaines séquences, ou certains mécanismes, voire même être complètement inapplicable. Par exemple, si dans l'évolution de protéines on veut tenir compte des relations entre acides aminés. De la même manière, les mutations qui font intervenir plusieurs nucléotides voisins (comme les CpG) contredisent cette hypothèse. De fait, la dynamique évolutive de la génomique des populations (par la transmission d'haplotypes) contredit cette hypothèse posée au niveau de la phylogénie.

2.2 Gestion de la racine

Dans l'exemple de la figure 1, une branche mène de la racine au nœud le plus haut de l'arbre. Cette branche correspond au passé du processus, avant le premier

2. $\odot$ est appliqué à tous les états, donc de complexité linéaire avec le nombre d'états.

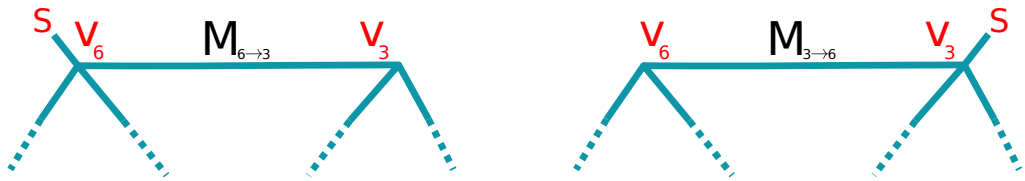

FIGURE 3 – Exemple du calcul du score selon que la racine est au nœud 6 ou au nœud 3 de l'arbre.

événement de spéciation. Le principe de l'inférence phylogénétique en un point de l'arbre est d'utiliser l'information issue de la comparaison entre les séquences qui sont de part et d'autre de ce point. En l'absence d'information sur ce qu'il y avait en amont de cette branche radicale, il n'est pas possible d'y reconstruire le processus. Ainsi, pour l'inférence phylogénétique, cette branche est enlevée de la modélisation et la racine est portée par le nœud le plus ancestral de l'arbre.

Dans le chapitre qui traite des méthodes de reconstruction basées sur les distances (voir chapitre...), les arbres inférés ne sont pas racinés. Ceci signifie que nous ne disposons pas de l'orientation du processus évolutif. Dans ce cas, sous quelles conditions est-il possible de calculer un score, ou bien faut-il toujours disposer d'une racine (*i.e.* d'une orientation du processus) ?

Considérons un arbre où le processus n'est pas orienté. Le sens de parcours d'une branche n'importe donc pas dans l'algorithme de calcul. Dans l'exemple de la figure 3, cela signifie que $M_{3\to 6} = M_{6\to 3} := M$. Dans ce cas, si on place la racine au nœud 6, le score est $S_6(\mathcal{D}) = \pi \otimes (v_6 \odot (M \otimes v_3))$ et si on la place au nœud 3, il est : $S_3(\mathcal{D}) = \pi \otimes (v_3 \odot (M \otimes v_6))$.

Il faut que les scores soient égaux. Comme le calcul est linéaire, on peut regarder pour toutes les composantes de v_3 et v_6. En notant δ_i le vecteur dans lequel tous les termes valent 0, sauf le i^{e}élément qui vaut 1, avec $v_3 = \delta_i$ et $v_6 = \delta_j$:

$$\begin{aligned} S_6(\mathcal{D}) &= \pi \otimes (\delta_j \odot (M \otimes \delta_i)) \\ &= \pi \otimes (\delta_j \odot \begin{pmatrix} M_{1i} \\ \vdots \\ M_{ji} \\ \vdots \\ M_{ni} \end{pmatrix}) = \pi \otimes \begin{pmatrix} 0 \\ \vdots \\ M_{ji} \\ 0 \\ \vdots \end{pmatrix} \quad \leftarrow \text{j}^{\text{e}}\text{ligne} \\ &= \pi_j \otimes M_{ji} \end{aligned}$$

et $S_3(\mathcal{D}) = \pi_i \otimes M_{ij}$.

Les scores sont donc égaux si $\forall i, j \in \mathbb{E}, \pi_i \otimes M_{ij} = \pi_j \otimes M_{ji}$. Sous cette condition, de proche en proche on peut déplacer la racine sur tous les points de l'arbre, et donc le score peut être calculé sur un arbre non orienté. Ceci est appelé **principe de la poulie**. En pratique, pour effectuer le calcul, la racine est placée en un point quelconque

En l'absence de cette propriété, l'orientation du processus a une importance dans le calcul du score, et il sera important d'identifier la position de la racine dans

l'arbre.

Dans les parties suivantes, nous appliquerons ces calculs aux approches spécifiques du maximum de parcimonie et de la vraisemblance. Nous présenterons des spécificités de chaque approche, et comment elles peuvent être utilisées pour l'inférence ancestrale.

3 Maximum de parcimonie

Le maximum de parcimonie vise à construire le scénario le plus parcimonieux – c'est-à-dire avec le moins de changements – pour expliquer les données. Les algorithmes qui permettent de calculer ce miminum ont été proposés par Walter FITCH en 1971 [1] et David SANKOFF en 1975 [2]. Nous présentons l'approche de SANKOFF, qui s'inscrit dans le calcul général de la programmation dynamique, et est plus généralisable que celle de FITCH. Cet algorithme peut être étendu de façon efficace quand l'espace des états est grand et la fonction de coût possède certaines propriétés (voir [3] pour une revue).

L'algorithme de calcul présenté dans la partie précédente s'effectue ici avec les opérateurs $(\min, +)$ pour $(\odot, \otimes)$ [3]. Il faut définir la matrice des coûts de substitution entre les états. Le coût le plus courant est d'établir qu'une substitution coûte 1 :

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Aux feuilles, les vecteurs valent ∞ (élément neutre de min) pour tous les états, hormis pour les états observés, où la valeur est 0 (élément neutre de $+$).

Dans l'exemple de la figure 4, si les états sont pris dans l'ordre `A`,`C`,`G`,`T` :

$$v_1 = \begin{pmatrix} 0 \\ \infty \\ \infty \\ \infty \end{pmatrix} \text{ et } v_2 = \begin{pmatrix} \infty \\ 0 \\ \infty \\ \infty \end{pmatrix}$$

$$v_{3\to1} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ \infty \\ \infty \\ \infty \end{pmatrix} = \begin{pmatrix} \min(0,\infty,\infty,\infty) \\ \min(1,\infty,\infty,\infty) \\ \min(1,\infty,\infty,\infty) \\ \min(1,\infty,\infty,\infty) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$v_{3\to2} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} \infty \\ 0 \\ \infty \\ \infty \end{pmatrix} = \begin{pmatrix} \min(1,\infty,\infty,\infty) \\ \min(0,\infty,\infty,\infty) \\ \min(1,\infty,\infty,\infty) \\ \min(1,\infty,\infty,\infty) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Si bien que

3. Par rapport aux calculs usuels dans $(\mathbb{R}, +, \times)$, min remplace « $+$ », « $+$ »remplace « $\times$ ».

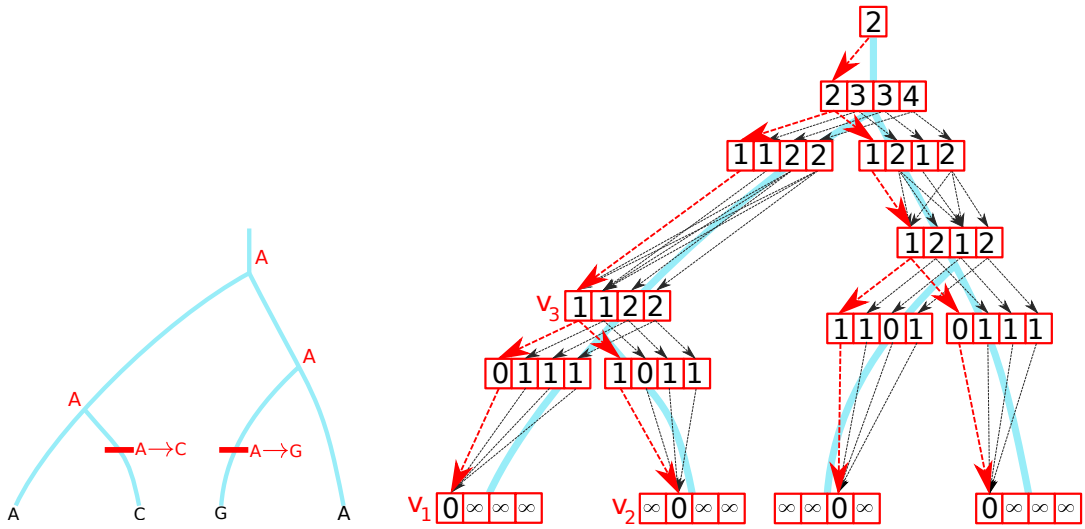

FIGURE 4 – Exemple du calcul de la parcimonie, et de la reconstruction du scénario le plus parcimonie. Gauche : les lettres aux feuilles, et les états inférés par le scénario le plus parcimonieux. Les deux barres rouges représentent les substitutions. Droite : les vecteurs avec les valeurs calculées, et en pointillés les liens de des arguments minimums des cellules.

$$v_3 = v_{3\to 1} \odot v_{3\to 2} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 2 \end{pmatrix}$$

Comme une substitution coûte 1, chaque valeur du vecteur v_3 représente le nombre de substitutions nécessaires pour expliquer les données à partir d'un état. Par exemple si au nœud 3 l'état est `G`, il faut deux substitutions (une sur chaque branche) pour expliquer les données sour ce nœud. S'il y a l'état `C`, une substitution suffit (sur la branche qui mène à la feuille 1).

Comme illustré dans la figure, on calcule le coût nécessaire à expliquer les données, en fonction de l'état présent en chaque nœud. Enfin, si à la racine il n'y a *a priori* pas de préférence sur l'état $\pi = 0$, et le minimum des valeurs de v_{racine} est gardé.

Il est tout à fait possible de mettre d'autres valeurs dans M, de manière à tenir compte de différents coûts de substitution entre les états. Néanmoins, il faut se rappeler que le coût sur une branche entière est le minimum de tous les coûts des processus possibles sur cette branche. Par conséquent, dans le cas où ce coût est le même quelque soit la branche (par exemple il ne dépend pas d'une longueur), si M est la matrice des coûts, $M \otimes M = M$ car introduire le calcul d'un vecteur intermédiaire à l'intérieur d'une branche ne doit pas modifier le résultat. Cela signifie que :
$\forall a, b \in \mathbb{E}, \text{cout(a,b)} = \min_{c\in\mathbb{E}}(\text{cout(a,c)} + \text{cout}(c, b))$.
En particulier, le coût doit respecter l'inégalité triangulaire :
$\forall a, b, c \in \mathbb{E}, \text{cout(a,b)} \leqslant \text{cout(a,c)} + \text{cout}(c, b)$.

Généralement, le coût correspond à une distance entre états, et est symétrique ($\text{cout}(a, b) = \text{cout}(b, a)$). Dans ce cas, le coût ne fait pas de différence entre les orientations du processus. Aussi, il n'est généralement posé de préférence sur les états à la racine, si bien que le calcul de la parcimonie ne dépend pas de l'emplacement de la racine dans l'arbre. Dans ce cas, ce calcul peut être effectué sur des arbres non racinés.

3.1 Inférence ancestrale

Comme le calcul de la parcimonie construit un minimum, il est directement possible d'identifier le (ou les) scénario le plus parcimonieux, c'est-à-dire à la fois les séquences ancestrales et les événements qui minimisent le nombre de substitutions sur l'arbre. Il suffit, à chaque calcul d'un minimum, de garder en mémoire à partir de quel état ce minimum a été atteint (ou plusieurs états en cas d'égalité). Puis, à partir de la racine, de redescendre l'arbre en suivant le chemin inverse pour avoir la suite des états qui ont atteint le minimum.

Par exemple, dans la figure 4, le score minimal est 2. Lors du calcul, dans chaque case des vecteurs on stocke les références des cases qui ont fourni les minimums (flèches descendantes). Ensuite, on suit à partir du minimum gardé à la racine les flèches jusqu'aux feuilles (flèches rouges). Si une flèche pointe des cases différentes des vecteurs, cela signifie qu'il y a eu une substitution sur la branche correspondante. Par exemple, dans la figure 4, la flèche rouge sur la branche $3 \rightarrow 2$ correspond à une substitution de l'état `A` vers l'état `C`.

Ainsi, il est rapide de reconstruire la succession des séquences ancestrales, ainsi que les substitutions successives entre elles.

4 Vraisemblance

Dans cette partie, nous revenons sur le calcul du score de données dans le contexte d'un processus probabiliste : ce score est la probabilité que les données aient été engendrées par ce processus, ce qui est appelé vraisemblance.

Le calcul de la vraisemblance des données s'effectue de la même manière que la parcimonie, mais cette fois dans l'algèbre standard $(+, .)$.

Les coûts de transition entre états sont désormais des probabilités. En raison de l'absence de mémoire des processus, ils sont markoviens, et les matrices de scores sont des matrices de probabilités conditionnelles. De même que pour la parcimonie, dans le calcul de ces probabilités il faut respecter le fait que cette fois, la probabilité de transition entre deux états sur une branche entière est la somme des probabilités de tous les processus possibles qui relient ces deux états sur cette branche. En particulier, les calculs doivent être indépendants de l'introduction d'un vecteur intermédiaire à l'intérieur d'une branche. Cela signifie que si on découpe une branche de matrice de transition M en deux parties de matrices M_1 et M_2, alors $M = M_1.M_2$. L'approche usuelle pour cela est de modéliser l'évolution par des processus de Markov continus (voir chapitre...) : le long d'une branche de longueur l, si Q

est le générateur du modèle dans cette branche, la matrice de transition selon cette branche est $M(l) = e^{Q.l}$.

Comme présenté précédemment, le calcul de la vraisemblance d'un alignement $\mathcal{D}$ s'effectue des feuilles vers la racine. L'algorithme a été décrit par Joe Felsenstein en 1973 [4, 5]. À un nœud n, les valeurs dans le vecteur v_n sont calculées sur la base des données $\mathcal{D}_n$ qui sont sous ce nœud. En notant X_n la variable aléatoire de l'état au nœud n, si x est un état, $v_n(x)$ (resp. $v_{n\to m}$ est la vraisemblance de $\mathcal{D}_n$ (resp. $\mathcal{D}_m$) si l'état en n est x :

$v_n(x) = P(\mathcal{D}_n|X_n = x)$ et $v_{n\to m}(x) = P(\mathcal{D}_m|X_n = x)$.

On parle de **vraisemblance conditionnelle**. L'opérateur $\odot$ est le produit terme à terme des éléments des vecteurs.

Aux feuilles, les vecteurs valent 0 pour tous les états, hormis pour les états observés, où la valeur est 1.

Dans l'exemple de la figure 4, les branches $3 \to 1$ et $3 \to 2$ ont pour longueur respective 1 et 0.5. On pose le générateur du modèle de transition :

$$Q = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} -0.652 & 0.174 & 0.406 & 0.0725 \\ 0.609 & -1.52 & 0.652 & 0.261 \\ 0.261 & 0.652 & -1.52 & 0.609 \\ 0.0725 & 0.406 & 0.174 & -0.652 \end{pmatrix} \end{array}$$

Les matrices de transition dans ces branches sont respectivement :

$$M_{3\to1} = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 0.583 & 0.124 & 0.182 & 0.111 \\ 0.273 & 0.323 & 0.217 & 0.187 \\ 0.187 & 0.217 & 0.323 & 0.273 \\ 0.111 & 0.182 & 0.124 & 0.583 \end{pmatrix} \end{array} \text{ et } M_{3\to2} = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 0.742 & 0.074 & 0.132 & 0.052 \\ 0.198 & 0.512 & 0.179 & 0.112 \\ 0.112 & 0.179 & 0.512 & 0.198 \\ 0.052 & 0.132 & 0.074 & 0.742 \end{pmatrix} \end{array}$$

Pour calculer v_3 on définit d'abord $v_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ et $v_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$.

Puis :

$$v_{3\to1} = M_{3\to1}.v_1 = \begin{pmatrix} 0.583 \\ 0.273 \\ 0.186 \\ 0.111 \end{pmatrix} \text{ et } v_{3\to2} = M_{3\to2}.v_2 = \begin{pmatrix} 0.074 \\ 0.512 \\ 0.179 \\ 0.132 \end{pmatrix}$$

et enfin

$$v_3 = v_{3\to1} \odot v_{3\to2} = \begin{pmatrix} 0.583 \\ 0.273 \\ 0.186 \\ 0.111 \end{pmatrix} \odot \begin{pmatrix} 0.074 \\ 0.512 \\ 0.179 \\ 0.132 \end{pmatrix} = \begin{pmatrix} 0.043 \\ 0.140 \\ 0.033 \\ 0.015 \end{pmatrix}$$

En remontant l'arbre, on calcule ainsi à la racine le vecteur des vraisemblances conditionnelles de l'ensemble des données. Pour enfin calculer la vraisemblance

globale, il faut multiplier ce vecteur par le vecteur π des probabilités des états *a priori* (c'est-à-dire sans la connaissance des données) : $P(\mathcal{D}) = \pi^T \times v_{\text{racine}}$.

En pratique, sur un alignement trop long, la vraisemblance est trop proche de zéro et il peut y avoir des problèmes de précision dans son calcul. On préfèrera alors calculer la log-vraisemblance des données :
$\log P(\mathcal{D}) = \sum_{i \in <1,l>} \log P(\mathcal{D}i)$.

4.1 Gestion de la racine

Dans le cas de la vraisemblance, pour appliquer le principe de la poulie, il faut que pour chaque générateur Q utilisé par le processus : $\forall i, j, Q_{ij}\pi_i = Q_{ji}\pi_j$. Cela signifie que le processus est stationnaire, c'est-à-dire que π est la distribution d'équilibre des différents modèles le long du processus. En particulier, les séquences, aussi bien observées qu'ancestrales, sont toutes à cet équilibre, ce qui est une très forte hypothèse.

Aussi, cette égalité signifie que Q est réversible : l'équilibre global est maintenu par un équilibre des transitions au sein de toutes les paires d'états.

Pour un processus hors de l'équilibre, ou bien dont l'équilibre n'est pas maintenu par la réversibilité du générateur, l'emplacement de la racine aura une importance dans le calcul de la vraisemblance. En particulier il sera possible d'optimiser l'emplacement de cette racine pendant l'inférence phylogénétique.

4.2 Calcul aux nœuds

Il est possible de calculer la vraisemblance en n'importe quel nœud n de l'arbre. Pour cela, il faut différencier les données qui sont en-dessous de ce nœud ($\mathcal{D}_n$) des données qui en sont de l'autre coté (le reste des données), que l'on note $\mathcal{D}^n = \mathcal{D} \backslash \mathcal{D}_n$. Comme $\mathcal{D}^n$ et $\mathcal{D}_n$ sont indépendantes :

$$\begin{aligned} P(\mathcal{D}) &= \sum_{x \in \mathbb{E}} P(\mathcal{D}|X_n = x)P(X_n = x) \\ &= \sum_{x \in \mathbb{E}} P(\mathcal{D}_n|X_n = x)P(\mathcal{D}^n|X_n = x)P(X_n = x) \\ &= \sum_{x \in \mathbb{E}} P(\mathcal{D}_n|X_n = x)P(\mathcal{D}^n, X_n = x) \end{aligned}$$

il donc reste à calculer $P(\mathcal{D}^n, X_n = x)$. On note les vecteurs des probabilités **jointes** :

$$\begin{cases} v^n &= (P(\mathcal{D}^n, X_n = x))_{x \in \mathbb{E}} \\ v^{p \to n} &= (P(\mathcal{D}^n, X_p = x))_{x \in \mathbb{E}} \\ v(n) &= (P(\mathcal{D}|X_n = x))_{x \in \mathbb{E}} = v^n \odot v_n \end{cases}$$

Calculer v^n signifie parcourir l'arbre depuis les feuilles de $\mathcal{D}^n$ jusqu'à n, les branches allant de la racine jusqu'à n étant prises à rebours (par exemple figure 5).

Au nœud n, avec p le père de n :

$$\begin{aligned} \forall x \in \mathbb{E}, v^n(x) &= P(\mathcal{D}^n, X_n = x) = P(\mathcal{D}^n|X_n = x)P(X_n = x) \\ &= \sum_{y \in \mathbb{E}} P(\mathcal{D}^n|X_p = y)P(X_p = y|X_n = x)P(X_n = x) \\ &= \sum_{y \in \mathbb{E}} P(\mathcal{D}^n|X_p = y)P(X_n = x|X_p = y)P(X_p = y) \\ &= \sum_{y \in \mathbb{E}} P(\mathcal{D}^n, X_p = y)P(X_n = x|X_p = y) \\ &= \sum_{y \in \mathbb{E}} v^{p \to n}(y) M_{yx}^{p \to n} \end{aligned}$$

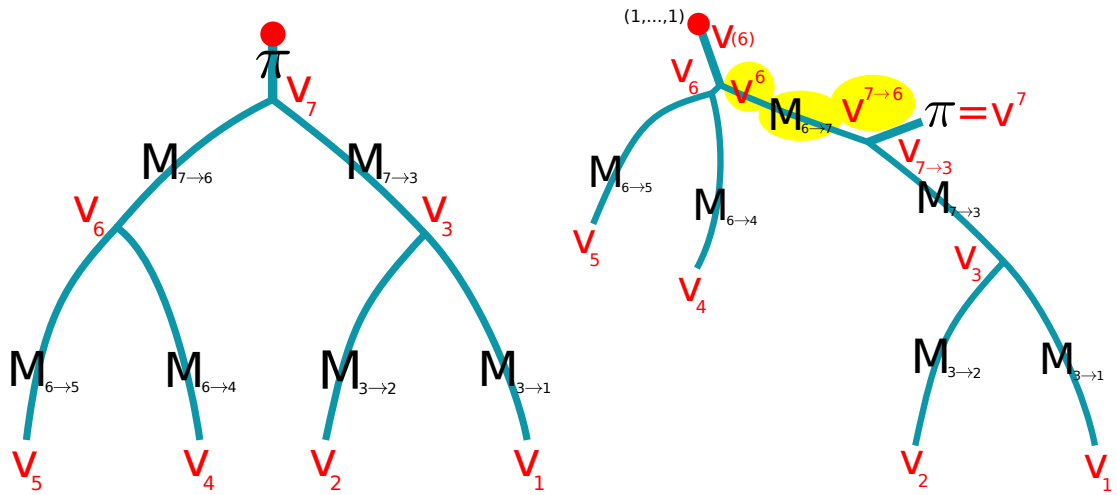

FIGURE 5 – Calcul de la vraisemblance à la racine (gauche), et au nœud 6 (droite). En jaune, les nouveaux éléments nécessaires au calcul des probabilités jointes ($M_{6\to 7} = {}^T M_{7\to 6}$).

Sur une branche $p \to n$:

$$\begin{aligned}
\forall x \in \mathbb{E}, v^{p\to n}(x) &= P(\mathcal{D}^n, X_p = x)P(X_p = x) \\
&= P(\mathcal{D}^p | X_p = x).\prod_{m\in e(p);m\neq n} P(\mathcal{D}_m | X_p = x)P(X_p = x) \\
&= P(\mathcal{D}^p, X_p = x).\prod_{m\in e(p);m\neq n} P(\mathcal{D}_m | X_p = x) \\
&= v^p(x).\prod_{m\in e(p);m\neq n} v_{p\to m}(x)
\end{aligned}$$

v^p est le vecteur des vraisemblances au-dela de p. Mais si p est la racine, il n'y a pas de données au-dessus de p, et donc v^p vaut la distribution *a priori* des états : π.

Donc le calcul des probabilités jointes s'effectue en parcourant l'arbre jusqu'au nœud d'intérêt, en appliquant les calculs :

$$\begin{cases} v^n &= M^T_{p\to n}.v^{p\to n} \\ v^{p\to n} &= v^p \odot \bigodot_{m\in e(p);m\neq n} v_{p\to m} \end{cases}$$

Une autre façon de représenter ce calcul est de considérer l'arbre de telle sorte que n est au sommet, avec une feuille fictive correspondant à la racine π. La vraisemblance est calculée comme précédemment, avec v_n et v^n de part et d'autre de n, en multipliant par la matrice M^T quand il s'agit de descendre une branche, et par la matrice M quand il s'agit de remonter une branche.

Finalement, $P(\mathcal{D}) = (1, \ldots, 1).v(n)$

4.3 Maximisation - Dérivation

Généralement, la vraisemblance $P(\mathcal{D})$ est utilisée pour trouver les processus qui modélisent « correctement »les données, soit selon une approche bayésienne, c'est-à-dire que les processus sont échantillonnés selon leur probabilité *a posteriori* des données (voir chapitre ...), soit selon une approche de maximum de vraisemblance, c'est-à-dire maximiser $P(\mathcal{D}|\Theta)$ sur un ensemble Θ de paramètres donnés qui suf-

fisent de définir les processus. Ces paramètres sont continus, tels que les paramètres des modèles, les longueurs de branches, ou bien discrets, telle que la topologie de l'arbre.

À la différence de la parcimonie, où le calcul du score définit un optimum, le calcul de la vraisemblance ne suffit pas à cela. Trouver le maximum de vraisemblance est un problème d'optimisation, sur lequel de nombreuses techniques d'optimisation numérique peuvent être appliquées.

Plusieurs de ces techniques sont des méthodes de gradient, qui nécessitent de connaître les dérivées de la vraisemblance en fonction des paramètres d'intérêt. En général, calculer analytiquement ces dérivées est trop difficile ou coûteux en temps (par exemple pour les paramètres des modèles), et il est préférable de recourir à des dérivées numériques. Néanmoins, lorsque le paramètre de dérivation est la longueur des branches, il est possible de calculer simplement la dérivée analytique.

La vraisemblance sur un ensemble de sites est le produit des vraisemblances locales, et dériver un long produit est trop coûteux. On préfèrera dériver (et optimiser) la log-vraisemblance, qui est une somme :

$\frac{dlP(\mathcal{D})}{d\theta} = \sum_{i\in<1,l>} \frac{dP(\mathcal{D}i)}{d\theta} \cdot \frac{1}{P(\mathcal{D}i)}$ et $\frac{d^2lP(\mathcal{D})}{d\theta^2} = \sum_{i\in<1,l>} \frac{\frac{d^2P(\mathcal{D}i)}{d\theta^2}P(\mathcal{D}i) - \left(\frac{dP(\mathcal{D}i)}{d\theta}\right)^2}{P(\mathcal{D}i)^2}$

Il reste à calculer $\frac{dP(\mathcal{D}i)}{d\theta}$ et $\frac{d^2P(\mathcal{D}i)}{d\theta^2}$ en chaque site.

4.3.1 Dérivation numérique

Le principe de la dérivée numérique est de faire la différence entre les vraisemblances autour de la valeur d'intérêt. Par exemple, autour de la valeur θ_0 du paramètre θ, en se donnant une petite valeur $h > 0$:

$\frac{dP(\mathcal{D})}{d\theta}\Big|_{\theta_0} \simeq \frac{P(\mathcal{D}|\theta=\theta_0+h) - P(\mathcal{D}|\theta=\theta_0-h)}{2h}$

$\frac{d^2P(\mathcal{D})}{d\theta}\Big|_{\theta_0} \simeq \frac{P(\mathcal{D}|\theta=\theta_0+h) - 2P(\mathcal{D}|\theta=\theta_0) + P(\mathcal{D}|\theta=\theta_0-h)}{h^2}$

Il existe bien sûr d'autres méthodes numériques, mais celle-ci a l'avantage de ne pas calculer la vraisemblance trop souvent. En effet, étant donné que l'optimisation peut nécessiter de nombreuses étapes d'approche du maximum, et que le calcul de la vraisemblance est potentiellement coûteux en temps, il est critique de chercher à rendre les calculs auxiliaires efficaces. Cette question est aussi valable dans le cadre de l'inférence bayésienne, qui généralement nécessite de nombreux calculs de vraisemblance.

Dans l'absolu, il faudrait contenir le calcul aux endroits de l'arbre où le paramètre intervient directement. Par exemple, s'il intervient dans la construction d'un modèle, le paramètre va avoir une action sur les branches où le modèle est appliqué. Si ce modèle est utilisé partout, il faudra recalculer la vraisemblance sur l'ensemble de l'arbre, mais s'il est utilisé sur une seule branche, grâce aux vraisemblances conditionnelles et jointes on peut limiter le calcul. Dans l'exemple de la figure 5, si on veut calculer des vraisemblances en modifiant les probabilités de transition entre les nœuds 6 et 7, puisque v_6 et $v^{7\to6}$ ne dépendent pas de $M_{6\to7}$, il suffit d'utiliser :

$P(\mathcal{D}) = (1...1).v_6 \odot (M_{6\to7}.v^{7\to6})$

avec les différentes matrices $M_{6\to7}$ que l'on veut tester.

4.3.2 Dérivation analytique

D'une façon générale, pour un paramètre quelconque θ, en considérant le calcul à la racine :

$$\begin{cases} \frac{dP(\mathcal{D})}{d\theta} & = \pi.\frac{dv_{\text{racine}}}{d\theta} + \frac{d\pi}{d\theta}.v_{\text{racine}} \\ \frac{dv_n}{d\theta} & = \sum_{y\in e(n)} \left((M_{n\to y}.\frac{dv_y}{d\theta} + \frac{dM_{n\to y}}{d\theta}.v_y) \odot (\bigodot_{\substack{z\in e(n)\\ z\neq y}} M_{n\to z}.v_z) \right) \end{cases}$$

Ainsi, il est possible de calculer à partir des feuilles les dérivées de v_n par rapport à θ. D'une façon générale, l'expression de la dérivée de M par rapport à un paramètre du modèle n'est pas facile à exprimer (principalement à cause du facteur de normalisation du générateur). Néanmoins, un cas particulier est de dériver par rapport à la longueur d'une branche, longueur notée $t_{n\to m}$: $\frac{dM_{n\to m}}{dt_{n\to m}} = Q_{n\to m}.M_{n\to m}$ et pour les autres branches b, $\frac{dM_b}{dt_{n\to m}} = 0$, si bien que :

$$\begin{cases} \frac{dP(\mathcal{D})}{dt_{n\to m}} = \frac{dv_{\text{racine}}}{dt_{n\to m}}.\pi \\ \\ \text{Si } n' \text{ n'est pas au-dessus de } n : \\ \qquad \frac{dv_{n'}}{dt_{n\to m}} \quad = 0 \\ \\ \text{Si } n' \text{ est au-dessus de } n, \text{ en notant } m' \in e(n') \text{ qui mène à } n \ : \\ \qquad \frac{dv_{n'}}{dt_{n\to m}} \quad = \left(\bigodot_{\substack{z\in e(n')\\ z\neq m'}} M_{n'\to z} v_z \right) \odot (M_{n'\to m'}.\frac{dv_{m'}}{dt_{n\to m}}) \\ \\ \frac{dv_n}{dt_{n\to m}} = \left(\bigodot_{z\in e(n) z\neq m} M_{n\to z} v_z \right) \odot (Q_{n\to m}.M_{n\to m}.v_m) \end{cases}$$

Le calcul de $\frac{dP(\mathcal{D})}{dt_{n\to m}}$ s'effectue en fait exactement de la même manière que celui de $P(\mathcal{D})$, à ceci près qu'aux nœuds de l'arbre **au-dessus** de la branche dérivée, il faut gérer le calcul des vecteurs dérivés au lieu des vecteurs des vraisemblances conditionnelles (figure 6).

Nous avons présenté dans ce calcul à partir des équations économiques en mémoire, qui n'utilisent pas les vecteurs $v_{n\to m}$ en haut des branches. Cependant, la démarche est aussi valable en utilisant ces vecteurs, et sera même plus rapide. Dans l'exemple de la figure 6, utiliser les vecteurs $v_{7\to 6}$ et $v_{3\to 2}$ permet d'éviter de multiplier les vecteurs en bas de leur branche avec les matrices de transition.

Aussi, comme la longueur de branche intervient sur une seule branche, on peut utiliser les probabilités jointes pour économiser beaucoup de calculs, comme pour les dérivées numériques. Dans notre exemple :

$\frac{dP(\mathcal{D})}{dt_{3\to 1}} = (1...1).v^{3\to 1} \odot (Q_{3\to 1} M_{3\to 1}.v_1)$

La démarche sera la même pour calculer les dérivées secondes par rapport aux longueurs de branche.

4.4 Inférence ancestrale

En plus de calculer la vraisemblance globale, les vecteurs de vraisemblances aux nœuds sont surtout utilisés pour l'inférence ancestrale. En effet, il est possible de cal-

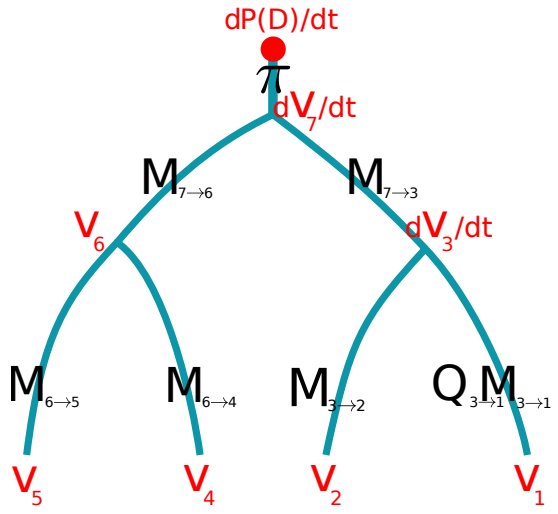

FIGURE 6 – Calcul de la dérivée de la vraisemblance en fonction de la longueur de branche $t_{3\to 1}$.

culer dans un nœud quelconque de l'arbre, les probabilités des différentes séquences ancestrales, ainsi que de décrire les événements évolutifs le long des branches.

Soit un nœud n, on dispose du vecteur des vraisemblances conditionnelles : $v(n) = (P(\mathcal{D}|X_n = i))_{i\in\mathbb{E}}$. En utilisant la formule de Bayes,
$P(X_n = i|\mathcal{D}) = \frac{P(\mathcal{D}|X_n=i).P(X_n=i)}{P(\mathcal{D})} \propto P(\mathcal{D}|X_n = i)P(X_n = i)$.

Et si l'*a priori* sur les états est uniforme, en normalisant pour avoir des probabilités :
$P(X_n = i|\mathcal{D}) = \frac{v(n)_i}{\sum_{i\in\mathbb{E}} v(n)_i}$.

Cette approche simple permet de calculer les probabilités ancestrales sur une séquence. En prenant en chaque site l'état le plus probable, on construit ainsi la séquence la plus probable. Mais cette approche n'est pas valable pour construire l'ensemble le plus probable de séquences à différents nœuds, car ces séquences sont dépendantes. Une méthode pour cela est présentée dans [6].

En ce qui concerne les substitutions le long des branches, on peut calculer l'espérance (sur l'ensemble des processus) du nombre de substitutions d'un type quelconque entre deux nœuds, ainsi que l'espérance du temps passé dans un état quelconque entre deux nœuds [4]. Il s'agit de **comptage probabiliste**.

Nous ne détaillerons pas ici les calculs qui mènent à ces comptages, mais exposons le principe.

Il y a deux étapes.

En premier lieu, imaginons que l'on veuille mesurer l'espérance du nombre de substitutions de `A` vers `C` sur une branche $n \to m$ de longueur l qui commence par l'état d et termine par l'état f, selon un modèle de générateur Q (voir figure 7) : $E(N_{AC}|X(n) = d, X(m) = f)$.

4. Il est aussi possible de calculer les variances et autres moments de ces valeurs, voir [7].

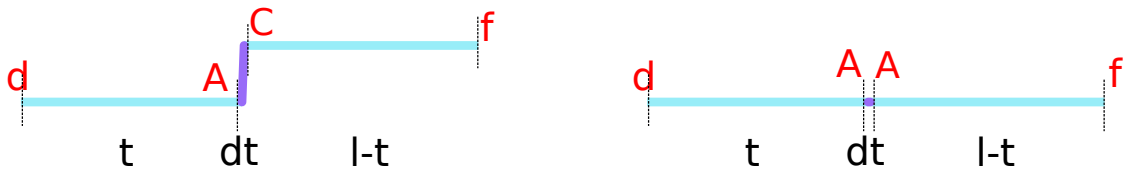

FIGURE 7 – Chemins élémentaires pour compter le nombre de transitions de A vers C (gauche), et le temps passé dans l'état A (droite).

Le calcul se fait sur l'ensemble des processus qui terminent en f sachant qu'ils commencent en d, et dont la probabilité est $(e^{Ql})_{df}$.

Considérons l'ensemble des processus dont l'état est A au temps t. Le nombre de ces processus qui passent de A à C pendant un petit temps dt est $Q_{\mathtt{AC}}dt$. Or, partant de d, la probabilité d'avoir A au temps t est $(e^{Qt})_{d\mathtt{A}}$, et la probabilité d'avoir un f à la fin de la branche à partir de C est $(e^{Q(l-t)})_{\mathtt{C}f}$. Donc, parmi les processus de d à f, l'espérance du nombre de transitions pendant dt de A vers C au temps t est $(e^{Qt})_{d\mathtt{A}}.Q_{\mathtt{AC}}(e^{Q(l-t)})_{\mathtt{C}f}dt$. Il suffit alors d'intégrer cette formule pour tous les t possibles entre 0 et l, et de diviser par la probabilité des processus de d à f :

$$E(N_{AC}|X(n)=d,X(m)=f)=\frac{Q_{\mathtt{AC}}}{(e^{Ql})_{df}}\int_{t=0}^{l}(e^{Qt})_{d\mathtt{A}}.(e^{Q(l-t)})_{\mathtt{C}f}dt.$$

On opère de même pour mesurer le temps moyen passé dans l'état A sur cette branche entre les états d et f : $E(T_{\mathtt{A}}|X(n)=d,X(m)=f)$. Pendant un temps dt, si l'état au temps t est A, le temps passé dans l'état A est dt. L'espérance du temps passé en A pendant dt est donc $(e^{Qt})_{d\mathtt{A}}.(e^{Q(l-t)})_{\mathtt{A}f}dt$, et donc :

$$E(T_A|X(n)=d,X(m)=f)=\frac{1}{(e^{Ql})_{df}}\int_{t=0}^{l}(e^{Qt})_{d\mathtt{A}}.(e^{Q(l-t)})_{\mathtt{A}f}dt.$$

Dans les deux cas, il faut calculer $I_{df}^{\mathtt{XY}}=\int_{t=0}^{l}(e^{Qt})_{d\mathtt{X}}.(e^{Q(l-t)})_{\mathtt{Y}f}dt$. Cela peut se faire de façon analytique, voir pour cela [8, 9].

En second lieu, nous intégrons ce calcul aux probabilités calculées dans l'arbre, sachant les données. Reprenons l'exemple des transitions de A vers C. Sur une branche $n\rightarrow m$ de longueur l, on souhaite calculer l'espérance du nombre de substitutions de A vers C, sachant les données $\mathcal{D}$: $E_{n\rightarrow m}(N_{AC}|\mathcal{D})$.

$$\begin{aligned}E_{n\rightarrow m}(N_{AC}|\mathcal{D}) &= \textstyle\sum_{d,f}E(N_{\mathtt{AC}}|X(n)=d,X(m)=f).P(X(n)=d,X(m)=f|\mathcal{D})\\ &= \textstyle\sum_{d,f}E(N_{\mathtt{AC}}|X(n)=d,X(m)=f).(e^{Ql})_{df}.P(X(n)=d|\mathcal{D})\\ &= Q_{\mathtt{AC}}\textstyle\sum_{d,f}I_{df}^{\mathtt{AC}}P(X(n)=d|\mathcal{D})\end{aligned}$$

Et de même, le temps de séjour dans l'état A est :

$$E_{n\rightarrow m}(T_A|\mathcal{D})\textstyle\sum_{d,f}I_{df}^{\mathtt{AA}}P(X(n)=d|\mathcal{D}).$$

En pratique, on ne calcule pas explicitement tous les $I_{df}^{\mathtt{XY}}$ car cela serait bien trop coûteux en temps, mais leur calcul peut être décomposé en morceaux qui seront assemblés d'une façon *ad hoc* lors du calcul des espérances [9].

Grâce à cette approche, on peut calculer l'espérance des différents types d'événements qui ont eu lieu au long du processus, en plus des différents états ancestraux. On accède ainsi au même type d'information que pour la parcimonie, mais dans un contexte probabiliste.

Références

[1] W.M. Fitch. Toward defining the course of evolution : minimum change for a specified tree topology. *Systematic Zoology*, 20(4) :406–416, 1971.

[2] D. Sankoff. Minimal mutation trees of sequences. *SIAM J. Appl. Math.*, 28 :35–42, 1975.

[3] Miklós Csűrös. *Models and Algorithms for Genome Evolution*, chapter How to infer ancestral genome features by parsimony : Dynamic programming over an evolutionary tree. Springer Verlag, 2013.

[4] J. Felsenstein. Maximum-likelihood estimation of evolutionary trees from continuous characters. *Am. J. Hum. Genet.*, 25(5) :471–492, Sep 1973.

[5] J. Felsenstein. Evolutionary trees from DNA sequences : a maximum likelihood approach. *J. Mol. Evol.*, 17(6) :368–376, 1981.

[6] T. Pupko, I. Pe'er, M. Hasegawa, D. Graur, and N. Friedman. A branch-and-bound algorithm for the inference of ancestral amino-acid sequences when the replacement rate varies among sites : Application to the evolution of five gene families. *Bioinformatics*, 18(8) :1116–1123, Aug 2002.

[7] A. Dhar and V. N. Minin. Calculating Higher-Order Moments of Phylogenetic Stochastic Mapping Summaries in Linear Time. *J. Comput. Biol.*, 24(5) :377–399, May 2017.

[8] V.N. Minin and M.A. Suchard. Fast, accurate and simulation-free stochastic mapping. *Phil. Trans. Roy. Soc. B*, 363 :3985–3995, 2008.

[9] P. Tataru and A. Hobolth. Comparison of methods for calculating conditional expectations of sufficient statistics for continuous time Markov chains. *BMC Bioinformatics*, 12 :465, Dec 2011.