UNIVERSITÉ DU QUÉBEC À MONTRÉAL

RECONSTRUCTION DES ANCÊTRES DE SÉQUENCES PARTAGEANT LEUR HISTOIRE DUPLICATIVE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

PHILIPPE LAVOIE-MONGRAIN

JUILLET 2014

REMERCIEMENTS

En premier lieu, je tiens à remercier ma directrice de maîtrise, Anne Bergeron. Son savoir, sa rigueur scientifique et son sens de la pédagogie sont véritablement hors du commun. Elle a été d'une aide et d'un support inestimables pour la rédaction de ce mémoire.

J'aimerais remercier Étienne Gagnon, directeur du programme de maîtrise en informatique, pour sa compréhension et pour m'avoir accordé les prolongations dont j'avais besoin pour peaufiner la rédaction du présent ouvrage.

Merci à Robert Godin pour avoir été mon mentor lors de ma première incursion dans le monde de la recherche. Merci à tous les professeurs de l'UQAM qui ont su susciter mon intérêt pour l'informatique et parfaire ma formation.

Merci à ma mère, Martine, qui a été le support constant dont j'ai eu besoin tout au long de ma vie. Je tiens également à remercier ma complice, Che, pour sa force de caractère et son soutien précieux.

Finalement, j'aimerais offrir mes remerciements à mes amis, Marc-André Bergeron, Benoît Garceau, Alexandre et Nicolas Saint-Aubin pour avoir su aérer mes idées lorsque j'en avais besoin.

TABLE DES MATIÈRES

LISTE DES FIGURES vii
LISTE DES TABLEAUX xi
RÉSUMÉ xiii
INTRODUCTION 1
CHAPITRE I
PRÉLIMINAIRES BIOLOGIQUES 5
1.1 Séquences d'ADN 5
1.1.1 Les alignements 7
1.2 Les protéines 8
1.3 Le rôle de l'ADN dans la synthèse des protéines 10
1.4 Les mutations 11
1.5 Les mutations sur les segments codants 12
1.6 Les phages et la TMP 14
CHAPITRE II
LES RÉPÉTITIONS EN TANDEM 17
2.1 Les répétitions en tandem 17
2.2 Modèle à frontières fixes et modèle à frontières variables 19
2.3 À la recherche des segments ancestraux 20
2.4 La recherche de la duplication la plus récente 22
2.5 Première solution 23
2.6 L'expérience de Belcaid et al. 26
2.6.1 Résultats et discussion 26
CHAPITRE III
MUTATIONS PRÉ-SPÉCIATION 29
3.1 Arbres de spéciations/duplications 29
3.2 Partage de l'histoire duplicative 34
3.3 Étiquetages parcimonieux 36

3.4 Algorithmes de Fitch 39
3.5 Solution pour 4 feuilles 42
3.5.1 Résultats des calculs de la moyenne des mutations pré-spéciation de Belcaid et al. (2011) 46
3.6 Algorithmes de Sankoff 46
CHAPITRE IV
IMPLÉMENTATIONS ET RÉSULTATS 63
4.1 Algorithme standard 63
4.2 Algorithme avec fenêtre coulissante 66
4.3 Algorithme avec antémémoire pour classes de nucléotides 70
4.4 Combinaison des deux optimisations 71
4.5 Comparaison des performances 73
4.6 Résultats avec des séquences codantes de TMP 76
CONCLUSION 81
BIBLIOGRAPHIE 83

LISTE DES FIGURES

Figure		Page
1.1	Une molécule d'ADN	6
1.2	Un alignement multiple obtenu à l'aide de l'outil Clustal W.	9
1.3	Structure typique d'un bactériophage	14
1.4	Liaison du ruban à mesurer et de la protéine adjuvante.	15
1.5	Séquence d'acides aminés de la TMP. Figure tirée de Siponen et al. (2009).	16
2.1	Alignements de deux TMP	27
2.2	Courbes des distances normalisées obtenues dans l'expérience de Belcaid et al. (2011).	28
3.1	Phylogénies des espèces ancestrales V et W et des espèces observées X, Y et Z.	31
3.2	Exemples d'arbres avec des mutations.	32
3.3	Arbre de spéciations/duplications.	33
3.4	Phylogénie de trois espèces.	34
3.5	Arbre de l'évolution des segments issus de la duplication la plus récente des espèces de la Figure 3.4.	35
3.6	Exemple d'arbre de partage d'histoire duplicative.	36
3.7	Un arbre de partage d'histoire duplicative où les feuilles sont annotées avec des nucléotides.	38
3.8	Étiquetages possibles de l'arbre de la Figure 3.7.	38
3.9	Arbre-F de l'arbre de la Figure 3.7.	40
3.10	Étiquetage manqué par la procédure de Fitch.	41
3.11	Exemple de segments issus de la duplication la plus récente sur 2 séquences	43
3.12	Arbre-F correspondant à la première liste de nucléotides des segments de la Figure 3.11.	43

3.13 Les 5 étiquetages parcimonieux de la liste de nucléotides `AACT`. 44

3.14 Courbes des distances de Hamming et des moyennes du nombre de mutations pré-spéciation normalisées obtenues par Belcaid et al. (2011). . . 46

3.15 Exemple de topologie où les feuilles sont annotées. 47

3.16 Exemple d'étiquetages sous le noeud n_1 de la Figure 3.15 à gauche et sous le noeud n_2 à droite. 48

3.17 Étiquetage parcimonieux sous le noeud n_2 de l'arbre de la Figure 3.15 étiqueté par le nucléotide `T`. 49

3.18 Étiquetages parcimonieux sous le noeud n_2 de l'arbre de la Figure 3.15 pour chaque nucléotide. 49

3.19 Première étape de l'exemple de l'Algorithme 4. 51

3.20 Deuxième étape de l'exemple de l'Algorithme 4. 53

3.21 Troisième étape de l'exemple de l'Algorithme 4. 55

3.22 Arbre de Sankoff correspondant à l'arbre de la Figure 3.15 où tous les coûts ont été calculés. 56

3.23 Exemple où les racines ne peuvent pas être étiquetées par n'importe quel nucléotide dans un étiquetage parcimonieux. 58

3.24 Exemple d'étiquetage parcimonieux généré avec l'Algorithme 5. 59

4.1 Temps d'exécution pour l'algorithme standard. Graphique en surface et courbes pour chaque nombre de séquences en fonction de leur taille. . . 73

4.2 Temps d'exécution pour l'algorithme avec fenêtre coulissante. Graphique en surface et courbes pour chaque nombre de séquences en fonction de leur taille. 74

4.3 Temps d'exécution pour l'algorithme avec antémémoire. Graphique en surface et courbes pour chaque nombre de séquences en fonction de leur taille. 74

4.4 Temps d'exécution pour l'algorithme avec les deux optimisations. Graphique en surface et courbes pour chaque nombre de séquences en fonction de leur taille. 75

4.5 Accélérations des optimisations par rapport à l'implémentation standard. 76

4.6 Courbes de distances de Hamming normalisées pour une longueur de segments de 120 nucléotides. 77

4.7 Courbes de distances de Hamming normalisées pour une longueur de segments de 60 nucléotides. 78

4.8 Courbes des coûts moyens de mutations pré-spéciation pour une longueur de segments de 120 nucléotides. 79

LISTE DES TABLEAUX

Tableau Page

1.1 Les acides aminés associés à la synthèse des protéines. 9

1.2 Codons de l'ADN. 11

RÉSUMÉ

La reconstruction de l'histoire duplicative d'une séquence génomique est un problème où on tente d'expliquer l'évolution d'un segment avec des mutations ponctuelles et des duplications. Une heuristique intéressante est la réduction récursive de la duplication la plus récente : à chaque étape, on identifie la duplication la plus récente, on la réduit et on recommence sur la séquence résultante.

L'identification des segments issus de la duplication la plus récente peut se faire avec une mesure de coûts sur cette paire de segments, comme la distance de Hamming par exemple.

Lorsque des séquences partagent leur histoire duplicative, il est possible d'élaborer des mesures de distance qui traitent l'ensemble des segments issus de la duplication la plus récente de ces séquences en parallèle.

On voit, dans ce mémoire, des algorithmes basés sur les travaux de Fitch (1971), Sankoff (1975), Benson et Dong (1999) et Belcaid et al. (2011). Ces algorithmes mesurent le coût moyen des mutation pré-spéciation des segments issus de la duplication la plus récente. Nous développons ces algorithmes pour permettre le traitement de plusieurs séquences en parallèle et présentons diverses optimisations.

MOTS CLÉS : Bioinformatique, répétitions en tandem approximatives, duplications en tandem, reconstruction parcimonieuse de l'histoire duplicative, partage d'histoire duplicative.

INTRODUCTION

L'ADN est une molécule vitale au bon fonctionnement de tout organisme cellulaire vivant. Elle contient, entre autres, l'information nécessaire à la synthèse des protéines.

Cette molécule évolue de façon dynamique par le biais des mutations. Certaines mutations vont survivre au processus de sélection naturelle, se répandant à travers les âges, et vont donner lieu aux molécules d'ADN qu'on observe aujourd'hui.

Il existe plusieurs sortes de mutations et quelques unes d'entre elles engendrent le phénomène qui est au coeur de ce travail : les répétitions en tandem approximatives.

La présence de telles répétitions sur des segments d'ADN a donné lieu au problème de reconstruction parcimonieuse de l'histoire duplicative d'une séquence. Dans ce problème, on cherche à déterminer la suite des mutations et des états ancestraux qui ont produit l'état actuel.

Plusieurs travaux (Benson et Dong, 1999; Jaitly et al., 2002; Tang, Waterman et Yooseph, 2002; Elemento, Gascuel et Lefranc, 2002; Belcaid, Bergeron et Poisson, 2011) étudient des solutions à ce problème.

Nous étendons, dans ce mémoire, les algorithmes développés par Belcaid et al. (2011) pour traiter plusieurs séquences en parallèles. Leurs travaux utilisent d'abord une heuristique de Benson et Dong (1999) et la testent sur des séquences de phages. Leurs résultats montrent qu'on peut intégrer le contexte biologique des séquences analysées, particulièrement le fait qu'elles partagent leur histoire duplicative, pour calculer des résultats plus significatifs.

Leurs algorithmes se basent sur les travaux de Fitch (1971) et traitent 2 séquences en parallèle. Nous généralisons ces algorithmes pour permettre le traitement d'un nombre

quelconque de séquences en parallèle en nous basant sur les travaux de Sankoff (1975).

L'algorithme résultant qui calcule le coût moyen de mutations pré-spéciation a une complexité temporelle qui grandit de façon cubique en fonction de la longueur des séquences. Nous définissons des optimisations pour accélérer les temps de calcul. Au final, nous obtenons des performances, pour nos implémentations, qui sont de 10 à 275 fois plus rapides que l'algorithme de base.

Une version préliminaire de ces travaux a été présentée sous forme d'affiche à la conférence RECOMB 2012 à Barcelone (Belcaid et al. 2012).

Nous présentons dans le premier chapitre une introduction aux aspects biologiques nécessaires à la compréhension du problème traité. Particulièrement, nous présentons l'ADN, les protéines, la synthèse des protéines, les phages et la protéine TMP.

Au deuxième chapitre, nous présentons les répétitions en tandem approximatives, la problématique, l'heuristique de Benson et Dong et les résultats qu'on obtenus Belcaid et al. avec cette heuristique sur des séquences codantes de TMP. On termine le chapitre en discutant de ces résultats, notamment de la différence entre les attentes et les observations.

Au troisième chapitre, nous montrons comment on peut représenter les phénomènes évolutifs avec des arbres binaires. Ces arbres vont servir à isoler les mutations pré-spéciation des segments issus de la duplication la plus récente. On présente les algorithmes de Fitch et la méthodologie de Belcaid et al. basée sur ces algorithmes. On présente leurs résultats obtenus avec cette méthode. On termine finalement en étendant leur solution pour supporter plusieurs séquences en parallèle en se basant sur les travaux de Sankoff (1975).

Le dernier chapitre présente les détails des implémentations et les optimisations utilisées pour accélérer les calculs. Nous verrons qu'on réduit la complexité temporelle de façon appréciable avec ces optimisations. Les performances de chacune des implémentations sont comparées en fonction de plusieurs valeurs de longueur et de nombre de séquences.

Finalement, on termine avec une expérience sur un ensemble de séquences de phages partageant leur histoire duplicative identifié dans le travail de Nicolas Massoulier (2013). Nous comparons les résultats obtenus par l'heuristique de Benson et Dong sur chacune des séquences avec ceux de notre algorithme intégrant l'ensemble des séquences en parallèle.

CHAPITRE I

PRÉLIMINAIRES BIOLOGIQUES

Ce chapitre a pour but de couvrir les notions d'ordre biologique nécessaires à la compréhension du problème traité dans ce mémoire. Dans un premier temps, nous introduisons les séquences d'ADN et leur représentation en informatique.

Dans un deuxième temps, nous présentons ce que sont les protéines et le rôle de l'ADN dans leur construction. Nous discutons ensuite des processus qui permettent l'évolution de l'ADN.

Les séquences traitées dans ce mémoire appartiennent à des virus appelés *phages*. Nous terminons ce chapitre en présentant ce que sont les phages et les particularités des segments analysées.

1.1 Séquences d'ADN

La majorité des notions biologiques introduites dans ce travail sont des définitions élémentaires qui se trouvent dans la plupart des manuels de biologie moléculaire. Le lecteur intéressé peut consulter l'oeuvre de Alberts (2007), par exemple.

L'ADN est une molécule formée de deux brins unis entre eux par des liens chimiques. La Figure 1.1 montre un schéma d'une molécule d'ADN. Chacun des brins de l'ADN est en fait une suite de *nucléotides*. Les nucléotides de l'ADN sont les 4 molécules suivantes : l'adénine (`A`), la guanine (`G`), la cytosine (`C`) et la thymine (`T`).

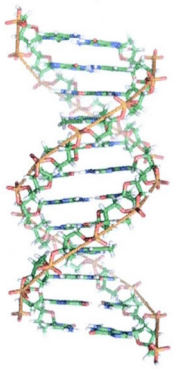


Figure 1.1: Une molécule d'ADN. Cette figure est tirée et modifiée de http ://en.wikipedia.org/wiki/DNA.

Il existe des méthodes (Sanger et Coulson, 1975; Staden, 1979) pour reconstruire la suite des nucléotides d'un segment d'ADN. Une fois cette suite déterminée, on dit du segment qu'il a été *séquencé*. Un segment d'ADN séquencé est souvent stocké dans une base de données publiques telles que celles disponibles sur le site du National Center of Biotechnology Information (NCBI). Ce segment se voit assigner un identifiant unique appelé un *numéro d'accession*.

Ces séquences sont stockées sous forme de chaînes de caractères. Chacun des caractères correspond à l'abréviation d'un nucléotide. Voici quelques exemples :

- `AAACCGAGATTTAGTGAGATGCACA` est une séquence de nucléotides valide.
- `GATTACA` est une séquence de nucléotides valide.
- `AAACCCCBNITTAGAGTGAGCACCCA` n'est pas une séquence valide, car `B`, `N` et `I` ne correspondent pas à des nucléotides.

Il n'est pas rare qu'on veuille faire ressortir les différences ou les similarités entre deux séquences. On utilise alors un *alignement*.

1.1.1 Les alignements

Un alignement est un traitement qui s'effectue sur des chaînes de caractères et qui ne se limite pas qu'aux séquences d'ADN. Les séquences sont alignées parallèlement, une par-dessus l'autre et, lorsqu'il y a débordement par la droite, l'alignement continue sur une nouvelle ligne.

Il en existe plusieurs types. Un *alignement pair à pair* implique deux séquences. Lorsqu'on aligne plus de deux séquences, on utilise le terme d'*alignement multiple*. Il est aussi possible qu'on aligne une séquence sur elle-même, le terme utilisé est alors un *alignement sur soi*.

Les séquences n'ont pas à être de longueurs égales pour être alignées. On réserve un caractère spécial pour les *écarts*, soit '-'. Pour toutes les colonnes de l'alignement, il y a un caractère associé à chaque séquence.

Définition 1. Soient $S_1 = s_{11} \dots s_{1n}$ et $S_2 = s_{21} \dots s_{2m}$, deux séquences de longueurs n et m respectivement, $k \leq m+n$ et '-' un symbole qui représente un écart. Un *alignement pair à pair* est un ensemble de paires $\{(x_1, y_1) \dots (x_k, y_k)\}$, présentées verticalement, tel que :

1. $x_i = s_{1j}$ pour $1 \leq j \leq n$, ou $x_i =$ '-';
2. $y_i = s_{2j}$ pour $1 \leq j \leq m$, ou $y_i =$ '-';
3. Pour une colonne i donnée, il y a au moins un caractère différent de '-';
4. La séquence $x_1 \dots x_k$ où les '-' sont retirés est S_1;
5. La séquence $y_1 \dots y_k$ où les '-' sont retirés est S_2;

Par exemple, si on aligne les séquences `CTTACAT` et `CTTAACGT` de longueur 7 et 8 respectivement, on peut obtenir le résultat suivant :

```
CTT-ACAT
CTTAACGT
```

Dans un alignement multiple de n séquences, on présente des tuples de n caractères verticalement au lieu de paires. Par exemple, si on aligne 4 séquences, on a 4 caractères par colonne. Un alignement sur soi est comme un alignement multiple à la différence que chacune des séquences est en fait un segment de la même séquence.

On appelle un *alignement sans écart* un alignement où aucun écart n'a été inséré.

On donne souvent un *coût* à un alignement. Il y a plusieurs façons de mesurer le coût d'un alignement. Un exemple de coût pour un alignement pair à pair est la *distance de Hamming* (Hamming, 1950) :

Définition 2. Soient S_1 et S_2, deux chaînes de caractères de longueurs égales k. La *distance de Hamming* se définit comme étant le nombre de caractères qui diffèrent sur les deux chaînes, c'est-à-dire le nombre de positions i tel que $s_{1i} \neq s_{2i}$ pour $1 \leq i \leq k$.

Par exemple, l'alignement suivant produit un coût de 4, calculé avec la distance de Hamming :

```
CTTACAT-
CTTAACGT
```

Par contre, l'alignement suivant a un coût de 2, avec les deux mêmes chaînes :

```
CTT-ACAT
CTTAACGT
```

La Figure 1.2 montre un exemple d'alignement multiple sans écart des séquences identifiées 1, 2 et 3. Cette figure montre un alignement sur deux lignes de 60 caractères. Il n'est pas rare que les outils d'alignements annotent les colonnes pour qualifier la conservation ou l'altération d'un caractère et, dans l'exemple présenté, les colonnes contenant un nucléotide identique dans les 3 séquences sont étoilées.

1.2 Les protéines

Une *protéine* est un type de molécule qui peut jouer plusieurs rôles dans un organisme ; une protéine peut être une enzyme, un anticorps, du matériel structurel, etc.

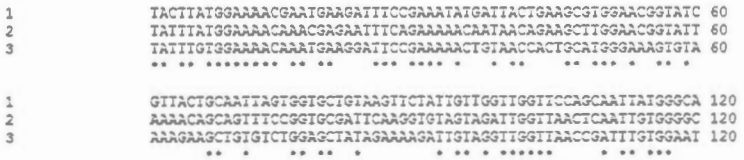


Figure 1.2: Un alignement multiple obtenu à l'aide de l'outil Clustal W.

Ces molécules sont construites par un processus appelé la *synthèse des protéines*. Elles sont assemblées à partir d'une séquence d'*acides aminés*. Cette séquence est unique à chaque protéine et elle détermine la structure et la fonction de la protéine.

Vingt-deux acides aminés sont utilisés lors de la synthèse des protéines. Ils ont une représentation abréviée à une lettre et le Tableau 1.1 donne la liste de celles-ci.

Acide Aminé	Abbréviation
Alanine	A
Cystéine	C
Acide aspartique	D
Acide glutamique	E
Phénylalanine	F
Glycine	G
Histidine	H
Isoleucine	I
Lysine	K
Leucine	L
Méthionine	M
Asparagine	N
Pyrrolysine	O
Proline	P
Glutamine	Q
Arginine	R
Sérine	S
Thréonine	T
Sélénocystéine	U
Valine	V
Tryptophane	W
Tyrosine	Y

Tableau 1.1: Les acides aminés associés à la synthèse des protéines.

À l'instar des séquences d'ADN, on représente une protéine comme une chaîne de caractères, où chaque caractère correspond à l'abréviation d'un des acides aminés du Tableau 1.1. Voici quelques exemples :

- **NRRTPPRSLGP** est une séquence d'acides aminés valide.
- **ECOLE** et **UNIVERSITE** sont deux séquences d'acides aminés valides.
- ZEBRE n'est pas une séquence valide, car **Z** et **B** ne sont pas des acides aminés associés à la synthèse des protéines.

Tout comme les segments d'ADN, les séquences protéines sont souvent stockées dans les bases de données publiques, et on leur assigne un numéro d'accession unique.

1.3 Le rôle de l'ADN dans la synthèse des protéines

Une protéine est construite dans un organisme à partir d'un *segment codant* de l'ADN. Ce segment encode la séquence d'acides aminés de la protéine. L'information codante est une suite de *codons*. Un codon est un groupement de trois nucléotides consécutifs. Par exemple, `TCA` est un codon. À chaque codon correspond un acide aminé, mais à chaque acide aminé correspond au moins un codon. Le codon `TCA` correspond à l'acide aminé **S**, mais cet acide aminé peut aussi être encodé par les codons `TCT`, `TCC`, `TCG`, `AGT` et `AGC`.

Seulement vingt-deux acides aminés entrent dans la synthèse des protéines alors qu'il existe $4^3 = 64$ codons différents. Plusieurs codons peuvent encoder le même acide aminé. Le Tableau 1.2 montre la table du code génétique.

Les codons d'une séquence d'ADN dépendent du *cadre de lecture*. La lecture d'un segment codant débute sur une position donnée et cette position détermine le cadre de lecture. Lorsqu'on lit une séquence de bout en bout, il existe trois cadres de lecture différents. Considérons la séquence `AAACCCGGGTTT` :

- Si la lecture débute sur le premier nucléotide, alors les codons sont `AAA`, `CCC`, `GGG`, `TTT` et la séquence d'acides aminés encodés est **KPGF**.
- Si la lecture débute sur le deuxième nucléotide, alors les codons sont `AAC`, `CCG`, `GGT`

Première base	Deuxième base								Troisième base
	T		C		A		G		
T	TTT	Phenylalanine (F)	TCT	Sérine (S)	TAT	Tyrosine (Y)	TGT	Cystéine (C)	T
	TTC	Phenylalanine (F)	TCC	Sérine (S)	TAC	Tyrosine (Y)	TGC	Cystéine (C)	C
	TTA	Leucine (L)	TCA	Sérine (S)	TAA	Stop (Ochre)	TGA	Stop (Opal)	A
	TTG	Leucine (L)	TCG	Sérine (S)	TAG	Stop (Amber)	TGG	Tryptophane (W)	G
C	CTT	Leucine (L)	CCT	Proline (P)	CAT	Histidine (H)	CGT	Arginine (R)	T
	CTC	Leucine (L)	CCC	Proline (P)	CAC	Histidine (H)	CGC	Arginine (R)	C
	CTA	Leucine (L)	CCA	Proline (P)	CAA	Glutamine (Q)	CGA	Arginine (R)	A
	CTG	Leucine (L)	CCG	Proline (P)	CAG	Glutamine (Q)	CGG	Arginine (R)	G
A	ATT	Isoleucine (I)	ACT	Thréonine (T)	AAT	Asparagine (N)	AGT	Sérine (S)	T
	ATC	Isoleucine (I)	ACC	Thréonine (T)	AAC	Asparagine (N)	AGC	Sérine (S)	C
	ATA	Isoleucine (I)	ACA	Thréonine (T)	AAA	Lysine (K)	AGA	Arginine (R)	A
	ATG	Méthionine (M)	ACG	Thréonine (T)	AAG	Lysine (K)	AGG	Arginine (R)	G
G	GTT	Valine (V)	GCT	Alanine (A)	GAT	Acide aspartique (D)	GGT	Glycine (G)	T
	GTC	Valine (V)	GCC	Alanine (A)	GAC	Acide aspartique (D)	GGC	Glycine (G)	C
	GTA	Valine (V)	GCA	Alanine (A)	GAA	Acide glutamique (E)	GGA	Glycine (G)	A
	GTG	Valine (V)	GCG	Alanine (A)	GAG	Acide glutamique (E)	GGG	Glycine (G)	G

Tableau 1.2: Codons de l'ADN.

et la séquence d'acides aminés encodés est **NPG**.

- Si la lecture débute sur le troisième nucléotide, alors les codons sont `ACC`, `CGG`, `GTT` et la séquence d'acides aminés encodés est **TRV**.

L'impact d'un changement du cadre de lecture sur les acides aminés encodés par un segment d'ADN codant est donc important.

1.4 Les mutations

L'ADN change dynamiquement par le biais de processus biologiques appelés *mutations*. Ces événements ont lieu de façon spontanée et permettent l'évolution de l'ADN. Une mutation est une altération de la séquence de nucléotides.

Une *mutation ponctuelle* est un changement qui n'affecte qu'un seul nucléotide. Il en existe trois sortes :

- La *substitution* est une mutation où un nucléotide est changé pour un autre. Exemple : `AAATTT` devient `AACTTT`.
- L'*insertion* est une mutation où un nucléotide est inséré dans la séquence. Exemple :

AAATTT devient AAACTTT.

- La *délétion* est une mutation où un nucléotide est retiré de la séquence. Exemple : AAATTT devient AATTT.

Une *amplification en tandem* (Rivals, 2004) est une mutation dans laquelle un groupe de nucléotides, appelé *motif*, est copié et où les copies et le segment original se retrouvent adjacents. Par exemple, voici une amplification qui génère 3 nouvelles copies du motif ACGT :

$$\underline{\texttt{ACGT}} \rightarrow \underline{\texttt{ACGTACGTACGTACGT}}$$

Les *frontières du motif* sont l'endroit où débute et termine ce dernier. Dans l'exemple ci-dessus, les frontières du motif sont les bases A et T.

L'*arité* d'une amplification est le nombre de copies générées plus 1. L'arité de l'amplification dans l'exemple précédent est 4. Dans ce travail, **nous nous concentrons sur les amplifications d'arité** 2, appelées les *duplications en tandem.*

Une amplification d'arité supérieure à 2 peut être représentée par une suite de duplications. L'exemple précédent peut s'expliquer avec le scénario suivant :

$$\begin{aligned} \underline{\texttt{ACGT}} &\rightarrow \underline{\texttt{ACGTACGT}} \\ \texttt{ACGT}\underline{\texttt{ACGT}} &\rightarrow \texttt{ACGT}\underline{\texttt{ACGTACGT}} \\ \texttt{ACGTACGT}\underline{\texttt{ACGT}} &\rightarrow \texttt{ACGTACGT}\underline{\texttt{ACGTACGT}} \end{aligned}$$

Ou encore :

$$\begin{aligned} \underline{\texttt{ACGT}} &\rightarrow \underline{\texttt{ACGTACGT}} \\ \underline{\texttt{ACGTACGT}} &\rightarrow \underline{\texttt{ACGTACGTACGTACGT}} \end{aligned}$$

1.5 Les mutations sur les segments codants

Une substitution n'affecte qu'un seul nucléotide. Pour un cadre de lecture donné, il se peut même que le codon résultant de la mutation soit associé au même acide aminé, ce qui rend la mutation *silencieuse.*

Par contre, une insertion ou une délétion **déplace** les nucléotides suivants. Imaginons, par exemple, le segment AAACCCGGGTTT codant pour la séquence d'acides aminés **KPGF**. Si on insère un C juste avant le premier G, alors l'impact se répercute sur tous les codons qui suivent : AAACCCCGGGTTT donne maintenant la protéine **KP**RV. L'impact d'une délétion est similaire.

Les chances que la nouvelle protéine puisse jouer le rôle de l'ancienne sont très minces, et ce genre de mutations ne survit généralement pas au processus de sélection naturelle.

Les séquences dont nous traitons dans ce travail sont des segments codants. Ainsi, il est raisonnable, dans un premier temps, d'ignorer les insertions/délétions :

Hypothèse 1. Les séquences traitées par nos algorithmes n'ont pas subi d'insertions ni de délétions.

Une duplication peut, ou non, avoir un impact similaire aux insertions/délétions. Si la longueur du motif est un multiple de 3, alors, pour n'importe quel cadre de lecture, les codons suivant ne sont pas affectés. Sinon, l'impact est analogue à une insertion. Par exemple :

ACGACGACG code pour **TTT**
ACGACGACG → ACGAACGACG code pour TND
ACGACGACG → ACGACACGACG code pour **TT**R
ACGACGACG → ACGACGACGACG code pour **TTTT**
ACGACGACG → ACGACGAACGACG code pour **TT**ND
...

De la même façon, pour nos séquences il est logique de ne considérer que les longueurs de segments dupliqués qui sont des multiples de 3 :

Hypothèse 2. Les séquences traitées par nos algorithmes n'ont subi que des duplications dont la longueur du motif est un multiple de 3.

1.6 Les phages et la TMP

Les séquences qui nous intéressent dans ce travail appartiennent à des *bactériophages*. Un bactériophage, ou *phage*, est un virus qui infecte les bactéries. Les phages représentent la forme de vie la plus abondante sur terre (Brüssow et Hendrix, 2002). Ce fait est reflété par l'important nombre de séquences d'ADN de phages disponibles dans les bases de données publiques.

On peut diviser la structure du phage en trois composants : la tête, la queue et l'embase. La Figure 1.3 montre la structure typique d'un phage.

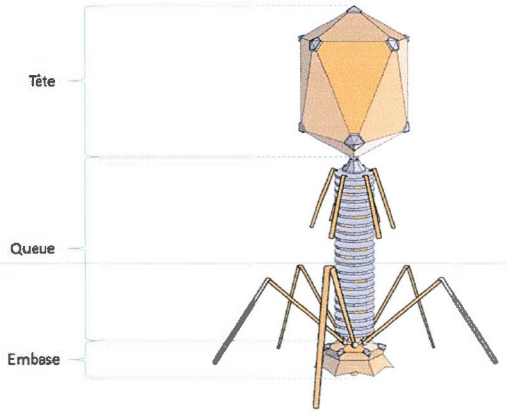


Figure 1.3: Structure typique d'un bactériophage. Cette figure est tirée et modifiée de http ://en.wikipedia.org/wiki/Bacteriophage.

La partie centrale du phage, soit la queue, est formée d'une protéine nommée *ruban à mesurer*, ou *tape measure protein* (TMP) en anglais. Dans les travaux de Katsura et Hendrix (1984), on démontre que la longueur de cette dernière détermine la longueur de la queue du phage, d'où son nom.

Lorsque le segment d'ADN codant pour la TMP est raccourci ou allongé, la TMP est elle-même proportionnellement raccourcie ou allongée respectivement : la longueur du segment d'ADN codant détermine la longueur de la TMP qui détermine la longueur de la queue du phage.

Des analyses crystallograhiques (Siponen et al., 2009) ont permis de mieux comprendre le rôle de la TMP dans la construction du phage. Elle offre des points d'ancrage à des protéines adjuvantes qui vont venir dérouler et étirer la TMP, formant ainsi la structure de la queue.

L'espacement régulier, ou la *période*, de ces points d'ancrage est un point critique à la construction du phage. Il est raisonnable de se questionner sur l'existence d'une unité de mesure de la TMP, similaire à un centimètre sur une règle, par exemple.

La Figure 1.4 est tirée du travail de Siponen et al. (2009). Elle montre une protéine TMP, au centre, qui accueille une protéine adjuvante, en haut. Les deux protéines se lient entre elles par le biais des liens biomoléculaires entre les acides aminés en mauve et en rose. La protéine résultante est en bas sur la figure.

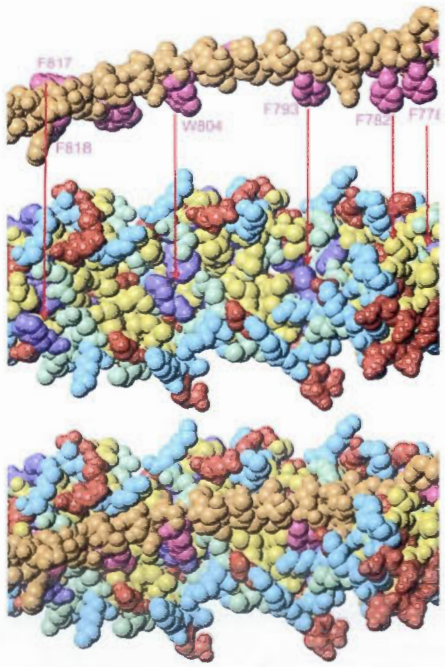


Figure 1.4: Liaison du ruban à mesurer et de la protéine adjuvante. Figure tirée de Siponen et al. (2009).

La Figure 1.5 montre la séquence d'acides aminés de la TMP présentée dans la Fi-

gure 1.4. La section surlignée en jaune correspond au segment présenté. Des segments d'acides aminés sont alignés pour mettre en évidence la périodicité des points d'ancrage, soit les tryptophanes (**W**) et les phénylalanines (**F**).

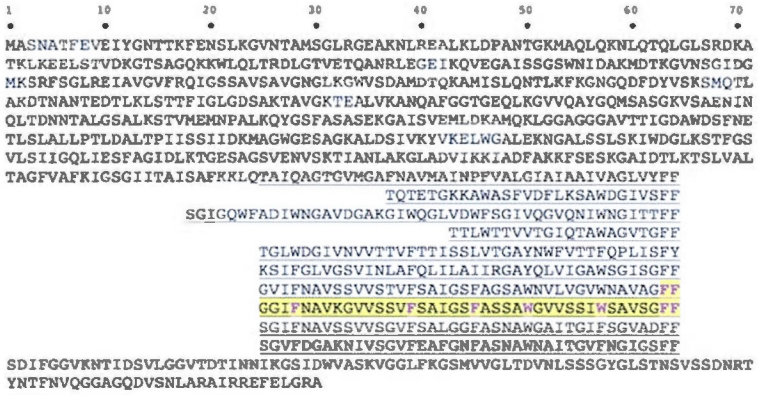


Figure 1.5: Séquence d'acides aminés de la TMP. Figure tirée de Siponen et al. (2009).

La périodicité de ces acides aminés de la TMP laisse croire que **son segment codant a subi plusieurs duplications en tandem au cours de son évolution**, ce qui en fait un candidat idéal pour nos algorithmes, dont le premier est présenté au chapitre suivant.

CHAPITRE II

LES RÉPÉTITIONS EN TANDEM

Le but de ce chapitre est de présenter la problématique et une première solution. On montre ensuite les performances de cette solution sur des séquences codantes de TMP.

Dans un premier temps, nous montrons ce que sont les répétitions en tandem et les particularités de leur évolution. Ensuite, nous présentons le problème de reconstruction de l'histoire duplicative et discutons d'approches existantes.

Nous montrons l'algorithme d'une première solution pour l'identification de la duplication la plus récente. L'expérience de Belcaid et al. (2011) utilise cet algorithme sur des séquences codantes de TMP. Nous terminons ce chapitre en discutant leurs résultats et les raisons des différences entre les attentes et les observations.

2.1 Les répétitions en tandem

Le produit d'une duplication en tandem s'appelle une *répétition en tandem*. En fait, une duplication engendre une *répétition exacte*, c'est-à-dire des copies identiques du motif. Par exemple, si on duplique le motif `AAT`, on obtient la répétition exacte `AATAAT`.

Au fil du temps, des mutations ponctuelles vont s'accumuler sur les copies et vont changer les répétitions exactes en *répétitions approximatives*.

En reprenant l'exemple précédent, si on substitue le premier `A` par un `C` on obtient la répétition approximative : `CATAAT`.

Plus une duplication est ancienne, plus elle accumule de mutations ponctuelles. L'inverse est aussi vrai. Plus elle est récente, plus ses copies sont similaires. On utilise le nombre de mutations qui différencient deux copies d'une duplication comme mesure pour quantifier l'âge de celle-ci.

Notons que n'importe quelle séquence peut être considérée comme une répétition approximative d'un motif quelconque. Par exemple, la séquence ACAATTTT peut être considérée comme une répétition approximative du motif CCCC. Il est pertinent de se demander d'abord si une séquence est issue d'une duplication avant d'identifier son motif.

Lorsqu'on pense qu'une séquence est issue d'une duplication on dit de cette séquence qu'elle est une *répétition significative*. L'identification des répétitions significatives est un domaine de recherche à part entière (Benson, 1999; Delgrange, Dauchet et Rivals, 1999; Kolpakov et Kucherov, 1999; Kolpakov et Kucherov, 2001; Rivals et al., 1997; Rivals et al., 1996; Stoye et Gusfield, 2002; Sagot et Myers, 1998).

Plusieurs duplications et plusieurs mutations ponctuelles entrelacées peuvent survenir au cours de l'évolution d'un segment. Voici un exemple de scénario évolutif :

ACGT	→ ACGTACGT	1 duplication
ACGTACGT	→ ACGGACGT	1 substitution
ACGGACGT	→ ACGGACGTACGGACGT	1 duplication
ACGGACGTACGGACGT	→ ACTGACGTCCGGAAGT	3 substitutions
ACTGACGTCCGGAAGT	→ ACTGACGTCCGGACGGAAGT	1 duplication

Les longueurs des répétitions approximatives sont plus difficiles à déterminer que celles des répétitions exactes, particulièrement pour l'oeil humain. Par exemple, il est difficile de cerner les segments répétés d'une séquence telle que :

ACTGTAAACGTCGTACGAGTAACGTCGTACTAGGAACCGTCGTCCTAGTAAACGTCGGACTAGTAACCGGCG

Par contre, il devient trivial de les identifier lorsqu'on utilise un alignement. L'exemple ci-dessous présente un alignement sur soi de la séquence précédente, où chaque ligne correspond à une répétition de longueur 15 nucléotides.

```
1  ACT-GTAAACGTCGT
2  ACGAGTAA-CGTCGT
3  ACTAGGAACCGTCGT
4  CCTAGTAAACGTCGG
5  ACTAGTAACCGGCG-
```

2.2 Modèle à frontières fixes et modèle à frontières variables

Il existe des cas où on peut supposer raisonnablement que les frontières du motif dupliqué restent toujours les mêmes. C'est ce qui s'appelle le *modèle à frontières fixes*. Dans l'exemple ci-dessous, les frontières du motif sont fixées à `A` et `T`.

```
ACGTACGT       →  ACGTACGTACGT
ACGTACGTACGT   →  ACGTACGTACGTACGT
```

Les cas où les frontières du motifs varient suivent le *modèle à frontières variables*. Dans l'exemple suivant, les frontières du motif de la première duplication sont `G` et `C` et celles de la deuxième duplication sont `C` et `A`.

```
ACGTACGT       →  ACGTACGTACGT
ACGTACGTACGT   →  ACGTACGTACGTACGT
```

Le résultat final est identique à celui où on a fixé les frontières à `A` et `T`. Par contre, en incorporant les mutations ponctuelles, toutes les frontières d'un motif ne produisent plus nécessairement le même résultat. L'exemple suivant montre une duplication sur le segment `ACGTTCGTACGT` ayant subi une substitution par rapport au segment `ACGTACGTACGT` avec deux ensembles de frontières différents.

```
ACGTTCGTACGT   →  ACGTTCGTTCGTACGT
ACGTTCGTACGT   →  ACGTTCGTACGTACGT
```

Les 2 segments produits sont divergents. Les mutations ponctuelles viennent donc ajouter de l'information quant aux frontières du motif dupliqué.

Les répétitions contenues dans les séquences de TMP sont intégrées à l'intérieur de la protéine et ne constituent pas son entièreté. Idéalement, à chaque duplication correspondrait un point d'ancrage, mais il arrive que la longueur des répétitions varient sporadiquement. Ces deux particularités laissent croire que les duplications sur le segment codant d'une TMP suivent le modèle à frontières variables (Belcaid, Bergeron et Poisson, 2011) :

Hypothèse 3. Les duplications des séquences analysées suivent le modèle à frontières variables.

2.3 À la recherche des segments ancestraux

Dans la majorité des cas, on ne dispose que de séquences actuelles observées. On ne dispose pas des segments *ancestraux* de leur évolution, contrairement à nos exemples. Un des problèmes en bioinformatique est de retrouver les segments ancestraux correspondant à l'évolution d'un segment.

Par exemple, voici une reconstruction évolutive du segment `ACGTACCTACGTACCT` où on retrouve 4 états ancestraux et où chacun d'entre eux subit une mutation.

```
ACGT              → ACGTACGT
ACGTACGT          → ACGTACGTACGTACGT
ACGTACGTACGTACGT  → ACGTACCTACGTACGT
ACGTACCTACGTACGT  → ACGTACCTACGTACCT
```

Voici une autre reconstruction du même segment où on retrouve 3 mutations.

```
ACGT      → ACGTACGT
ACGTACGT  → ACGTACCT
ACGTACCT  → ACGTACCTACGTACCT
```

En général, on favorise certaines reconstructions aux dépens des autres en suivant le *principe de parcimonie.*

Définition 3. Le *principe de parcimonie* (Fitch, 1971) est une application du rasoir d'Ockham (Thorburn, 1915) par lequel on favorise les explications qui minimisent le nombre d'hypothèses. Il consiste à favoriser les scénarios où on retrouve un minimum de mutations.

La *reconstruction parcimonieuse de l'histoire duplicative d'une séquence* est le problème dans lequel on tente d'expliquer, avec un minimum de mutations ponctuelles et de duplications, l'histoire évolutive d'une séquence.

Dans les travaux qui tentent de solutionner ce problème, on suppose généralement que la longueur du premier motif dupliqué est connue (Benson et Dong, 1999; Jaitly et al., 2002; Fitch, 1977; Tang, Waterman et Yooseph, 2002; Elemento, Gascuel et Lefranc, 2002).

Problème 1 (Histoire duplicative)**.** Étant donné une longueur k et une séquence S, déterminer une reconstruction parcimonieuse qui transforme un segment de longueur k en la séquence S.

Certains travaux (Benson et Dong, 1999; Jaitly et al., 2002) tiennent compte des insertions et des délétions qui surviennent après les duplications, changeant ainsi la longueur de ses copies. Leurs algorithmes s'attendent généralement à recevoir des alignements avec écarts. D'autres travaux, dont ceux de Tang et al. (2002), font abstraction des insertions et des délétions, et c'est notre cas ici, comme nous avons vu au chapitre précédent. Étant donné cette hypothèse, les segments issus d'une duplication gardent leur longueur. Pour un motif de longueur k, la longueur de la paire de segments résultante est $2k$.

Pour retracer l'histoire duplicative d'une séquence, on utilise une opération nommée la *réduction* (Rivals, 2004). On prend une paire de segments adjacents de longueurs égales, les *segments réduits*, et on les fusionne en un, le *segment fusionné*.

Dans l'exemple suivant, on réduit la paire de segments adjacents `ACGTACGT` vers le segment `ACGT` :

ACGTACGT → ACGT

Reconstruire l'histoire duplicative d'une séquence est un problème qui peut se résoudre avec une suite de réductions qui se termine sur un segment de longueur k.

Par exemple, étant donné une longueur $k = 4$ et une séquence $S =$ ACGTACGTACGTACGT, voici une suite de réductions qui reconstruit l'histoire du segment S :

ACGTACGTACGTACGT → ACGTACGT
ACGTACGT → ACGT

On y est arrivé avec 2 réductions. Le scénario évolutif correspondant est son inverse, soit :

ACGT → ACGTACGT
ACGTACGT → ACGTACGTACGTACGT

Les réductions se compliquent lorsqu'on traite les répétitions approximatives.

ACCTACGTACGTACCT → AC?TAC?T
AC?TAC?T → AC?T

Il existe des méthodes, dont celle de Benson et Dong (1999), pour produire des segments fusionnés qui tiennent compte des mutations ponctuelles.

2.4 La recherche de la duplication la plus récente

Le problème de reconstruction de l'histoire duplicative énoncé précédemment est NP-diffile (Rivals, 2004). Une heuristique qui nous intéresse particulièrement est celle de Benson et Dong (1999).

L'algorithme prend en entrée une séquence S. À chaque étape, on teste l'ensemble des paires de segments adjacents. Pour une position p et une longueur k données, une paire de segments adjacents est représentée par :

Segment 1	Segment 2
$S[p, p+k-1]$	$S[p+k, p+2k-1]$

Pour chacune des valeurs de p et k possibles, on calcul le coût suivant :

$$D(S[p, p+k-1], S[p+k, p+2k-1])/k$$

Où $D(S_1, S_2)$ est la distance de Hamming. Ce coût représente la moyenne des mutations qui séparent les 2 segments. La première paire de segment qui **minimise** ce coût est sélectionnée pour être réduite. Cette paire correspond aux segments issus de la duplication la plus récente de l'étape courante. On effectue la réduction sur cette paire et on recommence la procédure sur la séquence résultante de façon récursive jusqu'à ce qu'on termine avec un segment dont la longueur est fournie par l'utilisateur.

2.5 Première solution

Nous présentons ci-dessous l'adaptation de l'heuristique gloutonne de Benson et Dong pour identifier la duplication la plus récente.

Trois boucles imbriquées forment l'algorithme. La boucle externe génère toutes les longueurs k de segment dupliqué possibles. La longueur maximale qu'un segment dupliqué peut avoir sur une séquence de longueur l est la moitié de celle-ci, soit de $\lfloor l/2 \rfloor$.

Nous avons vu précédemment avec l'Hypothèse 2 que seules les longueurs k qui sont des multiples de 3 nous intéressent. Ainsi, la valeur minimale de k est 3, le pas de la boucle est 3 et la valeur maximale de k est $\lfloor l/2 \rfloor - (\lfloor l/2 \rfloor \mod 3)$.

Pour chacune des longueurs de segment k générées par la première boucle, la boucle intermédiaire va itérer sur toutes les positions p où une paire de segments adjacents de longueur k peut se trouver. La position maximale est $l - (2k)$, en supposant que la première position est 0.

Par exemple, pour la séquence : `ACGTACGTAAAA` de longueur $l = 12$ et pour $k = 2$, la

boucle intermédiaire va itérer sur les positions allant de 0 à $l - (2k) = 12 - (2 \cdot 2) = 8$, générant en séquence les 9 paires de segments candidats à la réduction :

```
(AC, GT), (CG, TA), (GT, AC), ..., (AA, AA)
```

Finalement, la boucle interne normalise sur k la distance de Hamming de la paire de segments. Pour notre algorithme, nous nous contentons d'appeler une fonction qui calcule la distance de Hamming avant de normaliser cette valeur.

La distance normalisée est comparée à la valeur minimale observée jusqu'à maintenant et, si elle lui est inférieure, on garde les valeurs de p et k en mémoire. L'algorithme se termine en retournant les premières valeurs de p et k qui ont minimisé ce coût, ainsi que ce coût minimal, parce que c'est intéressant de le connaître.

L'Algorithme 1 détaille la procédure.

Proposition 1. La complexité temporelle de l'Algorithme 1 est $O(l^3)$, où l est la longueur de la séquence.

Preuve. Le calcul de la distance de Hamming d'une paire de segments de longueur k implique k comparaisons. Nous allons calculer le nombre total de comparaisons effectuées par l'algorithme. La boucle externe génère des valeurs de k allant de 3 à $maxK = \lfloor l/2 \rfloor - (\lfloor l/2 \rfloor \mod 3)$ avec un pas de 3.

Pour chacune de ces valeurs, on teste les paires de segments allant de la position 0 à la position $l - 2k$. On a donc l'expression suivante qui calcule le nombre de comparaisons :

$$3 \cdot (\sum_{p=0}^{l-2k} k) + 6 \cdot (\sum_{p=0}^{l-2k} k) + 9 \cdot (\sum_{p=0}^{l-2k} k) + \ldots + maxK \cdot (\sum_{p=0}^{l-2k} k)$$

$$= 3 \cdot \sum_{k=1}^{maxK/3} \sum_{p=0}^{l-2k} k$$

En supposant que l est pair et multiple de 3, donc un multiple de 6, la valeur exacte de

Algorithme 1 *duplicationRecente*(S)

```
1: l ← longueur de la séquence
2: minK = 3
3: maxK = ⌊l/2⌋ − (⌊l/2⌋ mod 3)
4: coutMin ← ∞
5: resultat ← (∞, ∞, ∞)                          ▷ Meilleur résultat connu.
6: pour k ← minK à maxK faire
7:     pour p ← 0 à l − (2k) faire
8:         d ← Hamming(S[p, p + k − 1], S[p + k, p + 2k − 1])
9:         d_n ← d/k
10:        si d_n < coutMin alors
11:            coutMin ← distanceNormalisee
12:            resultat ← (p, k, coutMin)
13:        fin si
14:    fin pour
15: fin pour
       retourner resultat
```

cette expression est :

$$
\begin{aligned}
3 \cdot \sum_{k=1}^{l/6} \sum_{p=0}^{l-2k} k &= 3 \cdot \sum_{k=1}^{l/6} (l - 2k + 1) \cdot k \\
&= 3 \cdot \sum_{k=1}^{l/6} (lk - 2k^2 + k) \\
&= 3 \cdot \left(l \cdot \sum_{k=1}^{l/6} k - 2 \cdot \sum_{k=1}^{l/6} k^2 + \sum_{k=1}^{l/6} k \right) \\
&= 3 \cdot \left((l+1) \cdot ((l/6 \cdot (l/6+1))/2) - 2 \cdot ((l/6)^3/3 + (l/6)^2/2 + (l/6)/6) \right) \\
&= 3 \cdot \left((l+1) \cdot (l^2/72 + l/12) - (l^3/324 + l^2/2 + l/18) \right) \\
&= 3 \cdot \left(l^3/72 + l^2/12 + l^2/72 + l/12 - l^3/324 - l^2/108 - l/18 \right) \\
&= 3 \cdot \left(l^3 \cdot (1/72 - 1/324) + l^2 \cdot (1/12 + 1/72 - 1/108) + l \cdot (1/12 - 1/18) \right) \\
&= 3 \cdot \left(7l^3/648 + 19l^2/216 + l/36 \right)
\end{aligned}
$$

Les calculs lorsque l n'est pas un multiple de 6 sont similaires. On a donc une complexité de $O(l^3)$. □

2.6 L'expérience de Belcaid et al.

Dans les travaux de Belcaid et al. (2011), on utilise l'Algorithme 1 sur des séquences codantes de TMP pour identifier la duplication la plus récente. Les TMP de deux souches de phages infectant la bactérie *Clostridium botulinum* sont analysées. Leurs numéros d'accession sont YP_002803860 (A2_Kyoto) et YP_002862700 (Ba4_657). Les auteurs montrent, avec un bon niveau de confiance, que ces séquences *partagent leur histoire duplicative*. Cela signifie que l'ensemble des duplications ont eu lieu dans leur ancêtre commun, et pas de façon indépendante sur chacune des séquences.

Cette caractéristique fait en sorte qu'on s'attend à ce que l'algorithme identifie la même paire de segments comme étant issus de la duplication la plus récente pour les deux séquences, c'est-à-dire les mêmes valeurs de p et de k.

La Figure 2.1 montre les deux séquences d'acides aminés encodées alignées sur elle-même une à côté de l'autre. Les segments dupliqués sont de taille 11 acides aminés, soit 33 nucléotides sur le segment codant. Les acides aminés hautement conservés et vitaux à la fonction de la TMP sont mis en évidence en rouge et en bleu.

2.6.1 Résultats et discussion

Dans l'expérience de Belcaid et al., les duplications les plus récentes identifiées par l'algorithme pour les deux séquences ont toutes deux une longueur de segment dupliqué de $k = 33$ nucléotides. Cependant, leurs positions p sont divergentes, **contrairement à ce qui était attendu**.

La Figure 2.2 montre les courbes de distances normalisées pour toutes les paires de segments de longueur 33 nucléotides. La courbe en bleu représente les résultats pour la séquence A2_Kyoto alors que la courbe en rouge représente ceux pour Ba4_657. Les deux

Alignement de A2_Kyoto	Alignement de Ba4_657
VAITAVIAIGL	VAITAIIAIGL
LLWKNWDKIKQ	LLWKNWDKIKQ
VAQTLWTAIKT	AAQSLWDKIKT
VFTNIWTTITT	VFTGIWTTITT
VFTNIWTTITT	VFTNIWTTITT
VASNIWTSITT	VASNIWTSITT
VFTNIWTTITT	VFTNIWTSITT
IFTAIWTTITT	IFTNIWTAIST
IATNIWSSIVS	VLTSIWTTIVT
IFTTIWNVIVT	VFTTIWNVIVA
IFTPIKLFIEA	ILTPIGLFIEA
VWKGILAVIII	VFKGILAVIIV
VGAFIWNAIVT	VGAWICNSIVT
MWTNVWNVIQP	MWTNVWSVIQP
ILTAIWNVITT	ILTAIWNVITT
VWTAIWTTITT	VWTAIWTTITT
IAMAIWNTIVN	IATAIWNTIVS
AWNTIAGVVSA	AWNTIAGVVST
VMSAIWGVISS	VMSAIWGVISS
IWSSIYGTVSG	IWSTIYGTVSG
IMSSIWSTITE	IMSSIWSTITD
IWNNIVSTVSD	IWNNIVSTVSD
IVGNIASTISD	IVGNIASTISD
GFNSLIGTCSD	GFNALIGICSD
IFNNIKNTVMD	IFNNIKNTVMG
IFQGIWQGIKD	IFEGIWDGIKS
IINGGIGMLNN	IINGGIDMLNN

Figure 2.1: Alignements des deux TMP de A2_Kyoto et Ba4_657

courbes ne se mettent pas d'accord sur la duplication la plus récente. Particulièrement, les courbes ont plusieurs minimums et maximums locaux, mais pas aux mêmes endroits.

En présumant que les répétitions partagent bel et bien leur histoire duplicative, les différences entre les courbes s'expliqueraient par les **mutations ponctuelles** qui se produisent de façon **indépendante** à l'intérieur des séquences.

Nous verrons, dans le prochain chapitre, des méthodes qui permettent de quantifier les

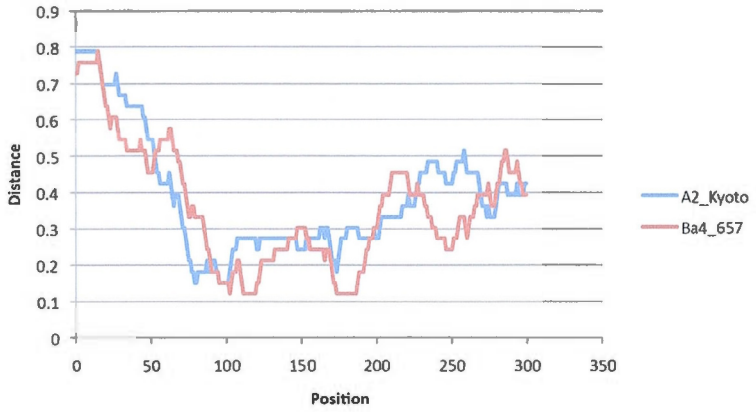


Figure 2.2: Courbes des distances normalisées obtenues dans l'expérience de Belcaid et al. (2011).

mutations ponctuelles partagées par les espèces et celles spécifiques aux espèces.

CHAPITRE III

MUTATIONS PRÉ-SPÉCIATION

Dans ce chapitre, nous raffinons la mesure de distance vue au Chapitre 2 pour isoler les mutations pré-spéciation.

Nous commençons d'abord par introduire les représentations arborescentes des spéciations, des duplications et du partage d'histoire duplicative.

Ensuite, nous voyons comment, à partir de ces représentations, on peut quantifier les mutations pré-spéciation des segments issus de la duplication la plus récente d'un ensemble de séquences partageant l'histoire duplicative.

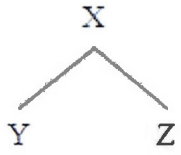
Pour ce faire, nous devons être en mesure de générer des étiquetages parcimonieux. Nous présentons les algorithmes de Fitch pour solutionner ce problème. Belcaid et al. (2011) s'en servent pour calculer le nombre moyen de mutations pré-spéciation et nous présentons leur méthodologie.

Finalement, on étend leur solution pour supporter plus de deux séquences en parallèle en utilisant les algorithmes de Sankoff (1975) présentés à la dernière section.

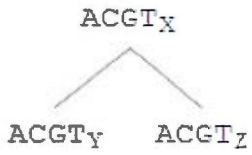
3.1 Arbres de spéciations/duplications

Le processus biologique par lequel de nouvelles espèces apparaissent se nomme la *spéciation*. La spéciation est la division d'une espèce en deux nouvelles espèces. Les spéciations peuvent se représenter sous forme d'arbre binaire. L'arbre suivant montre une spéciation

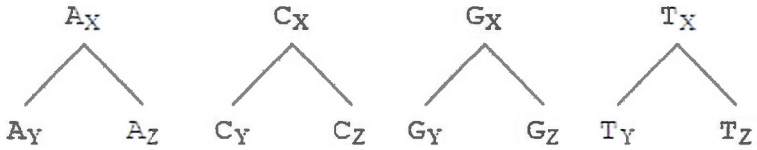
de l'espèce ancestrale X vers les espèces Y et Z.



On représente aussi les segments hérités par spéciation sous forme d'arbre binaire. Au lieu d'annoter les noeuds et les feuilles avec des espèces, on les annote avec des segments. L'exemple suivant montre le segment `ACGT` de l'ancêtre X hérité par Y et Z.



Chacun des nucléotides hérités peut aussi être représenté par un arbre de spéciation. Avec le segment de l'exemple précédent, on a les 4 arbres suivants.



On appelle la *phylogénie* d'un ensemble d'espèces les relations de parenté, exprimées par des arbres de spéciations, qui les relient entre eux.

Imaginons deux espèces ancestrales, V et W, et trois espèces modernes X, Y et Z issues de la spéciation des espèces ancestrales. Supposons que V est l'ancêtre de W. Les arbres de la Figure 3.1 représentent les phylogénies possibles.

Il existe plusieurs phylogénies possibles pour un ensemble d'espèces. Pour en connaître plus sur la phylogénie, le lecteur intéressé peut consulter l'oeuvre de Felsenstein (2004).

Hypothèse 4. La phylogénie des espèces auxquelles appartiennent les séquences ana-

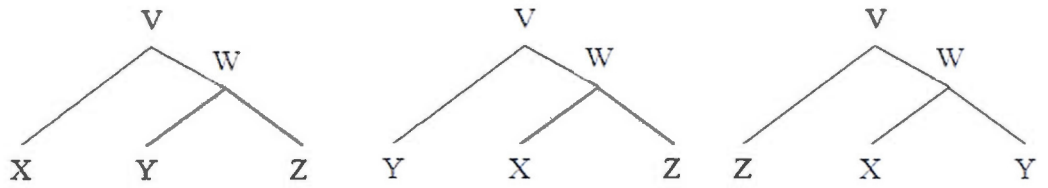
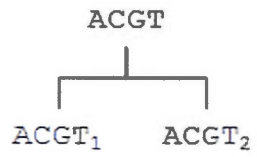


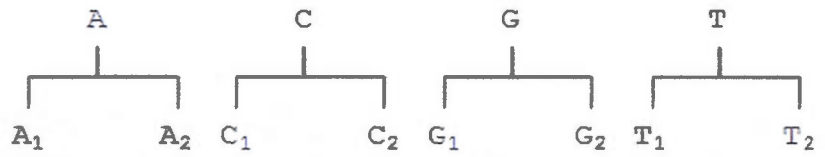
Figure 3.1: Phylogénies des espèces ancestrales V et W et des espèces observées X, Y et Z.

lysées nous est fournie.

Une duplication peut aussi se représenter sous forme d'arbre binaire. Pour différencier les duplications des spéciations, on utilise une arête à angle droit entre le noeud racine et ses enfants. Dans l'exemple suivant, le motif `ACGT` est dupliqué et chacun des noeuds enfants correspond à une copie.



Chacun des nucléotides dupliqué peut aussi être représenté sous forme d'arbre binaire. Avec la duplication de l'exemple précédent, on a les 4 arbres ci-dessous.



Les arêtes qui relient deux segments différents peuvent contenir une ou plusieurs mutations ponctuelles et les arêtes qui relient deux nucléotides différents contiennent une seule mutation ponctuelle. On met un trait sur l'arête pour faciliter son identification.

On présente deux arbres avec des mutations à la Figure 3.2. Dans l'exemple de spéciation,

à gauche, le nucléotide T de l'espèce X est hérité tel quel par les deux espèces. Au fil du temps, il devient G dans l'espèce Y. Dans l'exemple de duplication, le nucléotide T est dupliqué. La deuxième copie devient éventuellement C.

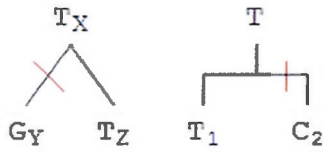


Figure 3.2: Exemples d'arbres avec des mutations.

Les deux formes d'arbres peuvent être utilisées conjointement pour expliquer l'histoire évolutive d'un segment ou d'un nucléotide. On appelle un tel arbre un *arbre de spéciations/duplications* :

Définition 4. Un *arbre de spéciations/duplications* est un arbre binaire tel que chaque noeud est annoté avec un segment et :

- Chaque noeud a obligatoirement 2 enfants, sauf si ce noeud est une feuille.
- Les spéciations sont représentées avec des arêtes directes entre un noeud et ses enfants. Chacun des enfants hérite du segment ancestral.
- Les duplications sont représentées avec des arêtes à angle droit entre un noeud et ses enfants. Chacun des enfants représente une copie du segment dupliqué.
- L'état d'un noeud représente l'état juste avant l'événement s'il a des enfants, l'état observé s'il s'agit d'une feuille.

Dans la Figure 3.3, nous suivons l'histoire évolutive du nucléotide A appartenant à l'espèce ancestrale S. Cette espèce subit une spéciation et devient les espèces U et V. Chacune d'entre elles hérite le nucléotide T. Ce nucléotide dans l'espèce U subit une substitution et devient le nucléotide T. L'espèce U subit une spéciation et devient les espèces W et X et le nucléotide T reste le même jusqu'au moment de l'observation. L'espèce V subit une duplication d'un motif dans lequel on trouve le nucléotide A hérité lors de la spéciation de l'espèce S. Au fil du temps, le nucléotide de la première copie issue

de la duplication devient C. L'espèce V subit une spéciation et devient les espèces Y et Z. Les deux copies de la duplication sont héritées par les deux espèces. Éventuellement, le nucléotide A de la deuxième copie de l'espèce Z devient le nucléotide G.

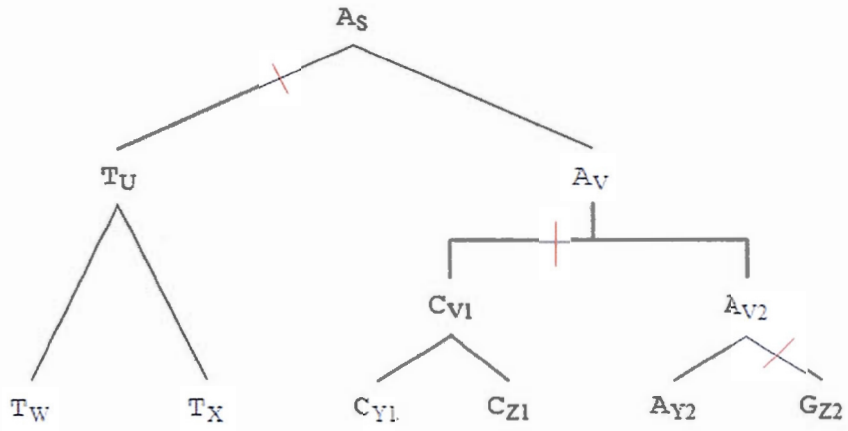


Figure 3.3: Arbre de spéciations/duplications.

Notons que les enfants de gauche des copies issues de la duplication dans l'espèce V correspondent aux nucléotides hérités par l'espèce Y. Les enfants droits correspondent aux nucléotides hérités par l'espèce Z.

Lorsque l'ancêtre commun **le plus récent** d'une paire de noeuds est un événement de duplication, alors on dit des segments contenus dans ces noeuds qu'ils sont *paralogues*. Lorsque l'ancêtre commun le plus récent d'une paire de noeuds est un événement de spéciation, alors on dit des segments contenus dans ces noeuds qu'ils sont *orthologues* (Fitch, 1970).

Sur l'arbre de la Figure 3.3, les nucléotides C_{V1} et A_{V2} sont paralogues car leur ancêtre commun le plus récent, A_V, s'est dupliqué. Les nucléotides C_{Y1} et G_{Z2} sont aussi paralogues, car leur ancêtre commun le plus récent est aussi A_V.

Par contre, les nucléotides C_{Y1} et C_{Z1} ne sont pas paralogues, car leur ancêtre commun le plus récent, C_{V1} s'est spécié. Ils sont donc orthologues. Les nucléotides T_W et A_{V2} sont aussi orthologues, car leur ancêtre commun le plus récent, A_S, s'est spécié.

Nous n'avons pas donné ici la liste exhaustive des relations de paralogie/orthologie de l'arbre de la Figure 3.3 . Chacun des noeuds est paralogue ou orthologue avec un autre, puisqu'ils ont un ancêtre commun.

3.2 Partage de l'histoire duplicative

Nous avons vu au chapitre précédent que les séquences de TMP utilisées dans l'expérience de Belcaid et al. partagent leur histoire duplicative. Nous voyons dans cette section la représentation d'un tel partage à l'aide d'arbres de spéciations/duplications.

Hypothèse 5. Les séquences traitées par nos algorithmes partagent leur histoire duplicative.

Supposons la phylogénie d'espèces de la Figure 3.4 :

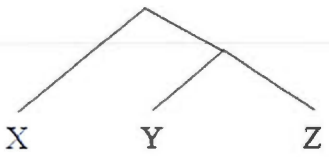


Figure 3.4: Phylogénie de trois espèces.

Supposons que ces espèces partagent leur histoire duplicative. Alors, on peut exprimer l'évolution des segments issus de la duplication la plus récente avec un arbre tel que celui de la Figure 3.5.

Nous appelons ce type d'arbre un *arbre de partage d'histoire duplicative* :

Définition 5. Un arbre de partage d'histoire duplicative pour un ensemble d'espèces S est un arbre de spéciations/duplications tel que :

- Le segment à la racine subit une duplication.
- Chacun des noeuds enfants de la racine représente l'évolution d'une copie à travers des espèces.

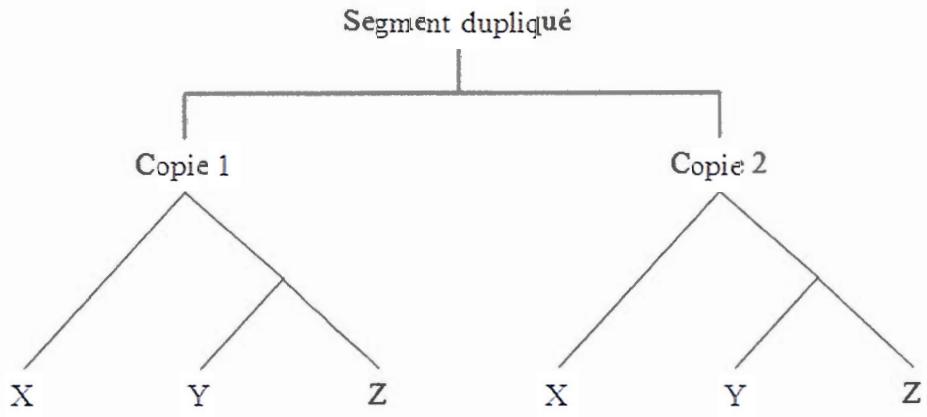


Figure 3.5: Arbre de l'évolution des segments issus de la duplication la plus récente des espèces de la Figure 3.4.

– Chacun des noeuds enfants de la racine est un sous arbre dont la topologie est identique à la phylogénie des espèces S.

Les mutations ponctuelles qui ont lieu avant la première spéciation sont appelées des mutations *pré-spéciation*. Ces mutations ont lieu entre la racine et ses enfants. Les mutations ponctuelles qui ont lieu après la première spéciation sont appelées des mutations *post-spéciation*. Toutes les mutations qui ont lieu sous les enfants de la racine sont des mutations post-spéciation.

L'arbre de la Figure 3.6 retrace l'évolution du nucléotide `T` appartenant au motif de la duplication la plus récente des trois espèces de la Figure 3.4. Le nucléotide `T` est d'abord dupliqué dans l'ancêtre commun des trois espèces X, Y et Z. Avant la première spéciation, la première copie du nucléotide change et devient `G`. L'espèce ancestrale se divise ensuite en deux espèces, soit X et l'ancêtre commun de Y et Z. Chacune de ces espèces hérite des deux copies. La première copie change dans l'ancêtre commun de Y et Z pour devenir `C`. Cette espèce subit ensuite une spéciation et devient les espèces observées Y et Z. Les deux copies changent finalement dans l'espèce Y pour devenir des `A`.

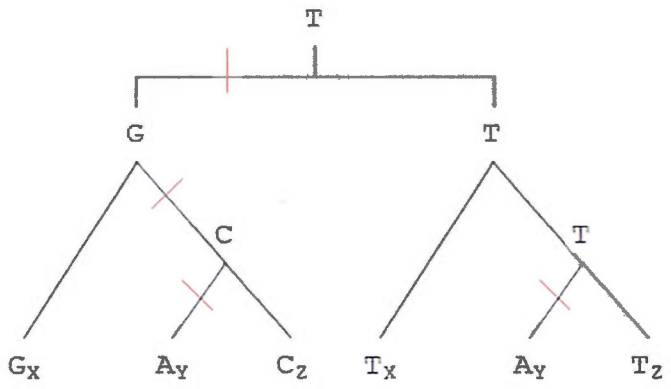


Figure 3.6: Exemple d'arbre de partage d'histoire duplicative.

Dans cet exemple, on compte un total de 4 mutations ponctuelles. La mutation de `T` vers `G` après la duplication est une mutation pré-spéciation. Toutes les mutations suivantes, soit de `G` vers `C`, de `C` vers `A` et de `T` vers `A`, sont post-spéciation.

3.3 Étiquetages parcimonieux

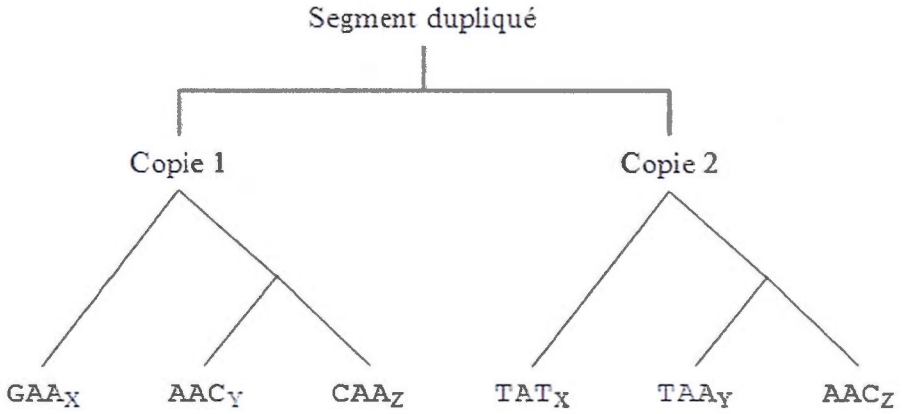
En réalité, nous ne disposons que de l'état observé, soit des segments dans les feuilles de l'arbre de partage d'histoire duplicative, et pas des segments ancestraux, soit ceux dans les noeuds intermédiaires. Or, nous avons vu qu'avec ces segments ancestraux, il est facile de calculer le nombre de mutations pré-spéciations, particulièrement lorsque ces segments sont des nucléotides. En effet, à une arête reliant deux nucléotides différents entre la racine et un de ses enfants correspond exactement une mutation pré-spéciation.

Supposons trois séquences, X, Y et Z dont la phylogénie est celle de la Figure 3.4. L'arbre de partage d'histoire duplicative pour ces séquences a donc la topologie de la Figure 3.5.

Supposons que les segments observés issus de la duplication la plus récente dans les espèces X, Y et Z sont ceux ci-dessous :

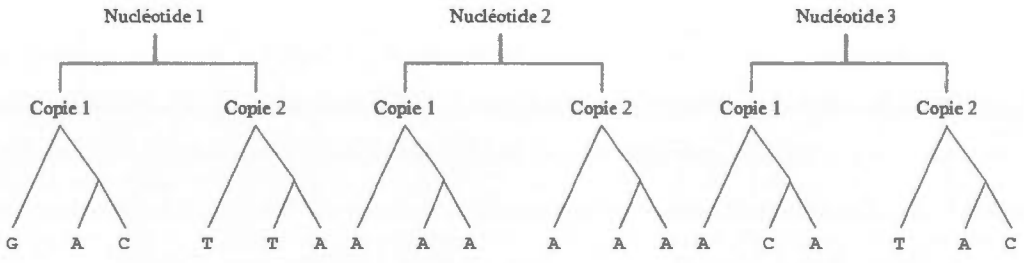
	Segment 1	Segment 2
X	GAA	TAT
Y	AAC	TAA
Z	CAA	AAC

Si on annote les feuilles de l'arbre de la Figure 3.5 avec ces segments, on obtient l'arbre de partage d'histoire duplicative suivant :



Ces segments ont une longueur de 3. Chaque nucléotide est paralogue avec le nucléotide situé 3 positions plus loin. Chaque nucléotide est orthologue avec les nucléotides situés à la même position sur les autres séquences.

Avec les segments précédents, on a les trois arbres suivants qui modélisent l'évolution de chacun des trois nucléotides du segment dupliqué :



Nous nous servirons de l'arbre représentant l'évolution du premier nucléotide tout au long du chapitre, soit celui de la Figure 3.7.

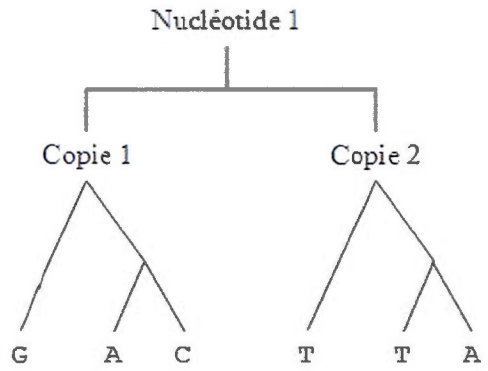


Figure 3.7: Un arbre de partage d'histoire duplicative où les feuilles sont annotées avec des nucléotides.

Nous appelons un *étiquetage* de l'arbre une instance où chacun des noeuds est étiqueté avec un nucléotide.

On présente trois étiquetages possibles de l'arbre de la Figure 3.7 à la Figure 3.8.

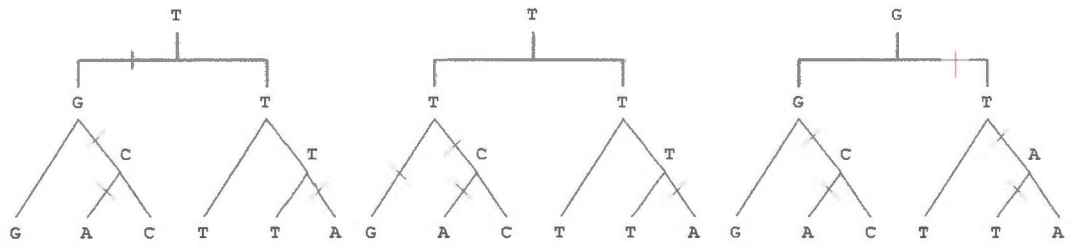


Figure 3.8: Étiquetages possibles de l'arbre de la Figure 3.7.

Les deux étiquetages de gauche comptent 4 mutations alors que celui de droite en compte 5. Puisque nous suivons le principe de parcimonie, l'étiquetage de droite est moins intéressant. Aussi, rien n'indique que les deux étiquetages de gauche sont parcimonieux, nous savons seulement qu'ils comptent moins de mutations que celui de droite.

En général, il y a plusieurs étiquetages qui comportent le même nombre de mutations ponctuelles. De surcroît, les étiquetages équivalents ne comptent pas nécessairement le même nombre de mutations pré-spéciation, comme on le voit dans les deux arbres de

gauche de la Figure 3.8.

Une étape cruciale à la quantification du nombre de mutations pré-spéciation est donc la résolution du problème suivant :

Problème 2 (Génération d'étiquetages parcimonieux)**.** Étant donné un arbre où les feuilles sont annotées avec des nucléotides, générer tous les étiquetages parcimonieux.

3.4 Algorithmes de Fitch

Dans les travaux de Fitch (1971), on retrouve une solution pour générer des étiquetages parcimonieux étant donné un arbre où les feuilles sont annotées avec des nucléotides.

Algorithme 2 (Remplissage de l'arbre-F, Fitch)**.** Soit un arbre A dont les feuilles sont annotées avec des ensembles contenant un nucléotide. On remplit récursivement les noeuds de l'arbre avec des ensembles formés de l'intersection des ensembles des noeuds enfants ou de leur union si l'intersection est vide. On appelle de tels ensembles des *ensembles de Fitch.* L'arbre résultant de cette procédure se nomme un *arbre-F.*

Exemple. On reprend l'arbre de la Figure 3.7 de la section précédente où les feuilles sont annotées par la liste de nucléotides `GACTAT` et on construit l'arbre-F correspondant. Le résultat est à la Figure 3.9. Les unions sont mises en rouge alors que les intersections sont mises en bleu. On compte 4 unions pour la construction de cet arbre.

Proposition 2. Le nombre d'unions effectuées par l'Algorithme 2 pour remplir l'arbre-F est le nombre de mutations ponctuelles d'un étiquetage parcimonieux.

La procédure de Fitch pour générer un étiquetage parcimonieux à partir d'un arbre-F est la suivante :

Algorithme 3 (Génération d'étiquetages parcimonieux, Fitch)**.** On choisit un nucléotide dans l'ensemble de la racine. Pour chacun de ses enfants, si ce nucléotide se retrouve dans l'ensemble de Fitch de cet enfant, alors cet enfant est étiqueté avec ce nucléotide. Sinon, on étiquette l'enfant avec n'importe lequel des nucléotides dans son ensemble de Fitch. On répète le processus pour chaque noeud en parcourant l'arbre de haut en bas.

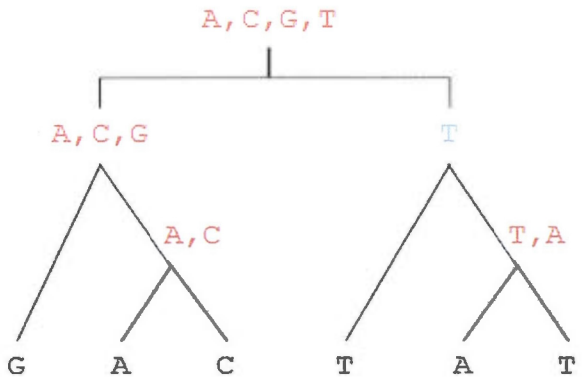
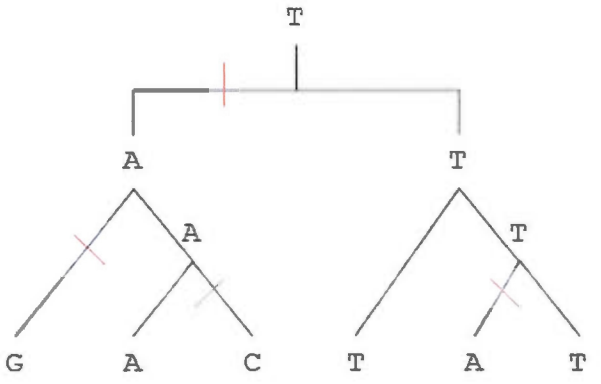


Figure 3.9: Arbre-F de l'arbre de la Figure 3.7.

Exemple. L'étiquetage suivant est produit avec l'Algorihtme 3 sur l'arbre-F de la Figure 3.9. L'ensemble de Fitch de la racine contient tous les nucléotides. On peut choisir n'importe lequel d'entre eux pour étiqueter la racine. On choisit `T` comme racine. Puisqu'il n'est pas dans l'ensemble de Fitch de son enfant gauche, on peut choisir n'importe quel nucléotide dans son ensemble {`A`, `C`, `G`} pour l'étiqueter. On choisit `A`. Par contre, `T` est dans l'ensemble de Fitch {`T`} de son enfant droit. Ainsi, on étiquette cet enfant avec `T`. Par la même règle, on étiquette l'enfant droit de l'enfant gauche de la racine avec `A` et l'enfant droit de l'enfant droit de la racine avec `T`.



Le nombre de mutations ponctuelles de l'étiquetage est 4, soit le nombre d'unions de l'arbre-F de la Figure 3.9.

Proposition 3. L'Algorithme 3 génère des étiquetages parcimonieux.

Malheureusement, la contraposée de la Proposition 3 n'est pas vraie : la procédure de Fitch ne génère pas **tous** les étiquetages parcimonieux. L'étiquetage de la Figure 3.10 compte aussi 4 mutations ponctuelles mais il ne peut pas être généré avec la procédure de Fitch.

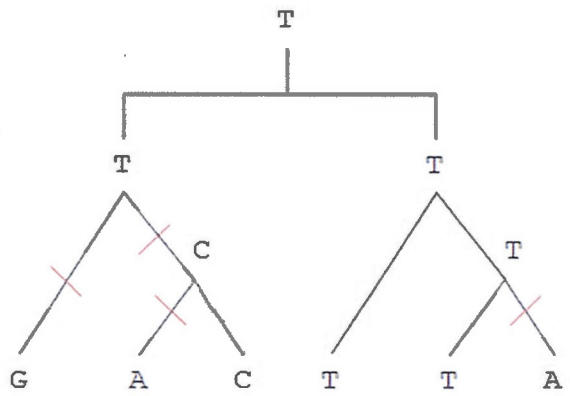


Figure 3.10: Étiquetage manqué par la procédure de Fitch.

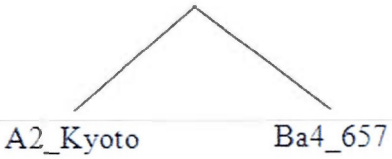
En effet, puisque l'ensemble de Fitch de l'enfant gauche de la racine de l'arbre-F de la Figure 3.9 ne contient pas le nucléotide T, alors ce nucléotide ne sera jamais sélectionné pour étiqueter ce noeud. Néanmoins, on peut se servir de l'Algorithme 3 pour calculer le nombre de mutations ponctuelles que contient un étiquetage parcimonieux, comme on voit dans la prochaine section.

Remarque (Felsenstein, 2004) : nous n'avons pas donné les preuves des Propositions 2 et 3 car elles n'existent pas dans l'article original. Les tentatives subséquentes des mathématiciens pour justifier les résultats de Fitch sont restées vaines : il semble que le problème soit difficile dans le cadre proposé par Felsenstein (2004). Néanmoins, la validité des algorithmes a été obtenue par David Sankoff (1975) comme conséquence d'un résultat beaucoup plus général, discuté dans la Section 3.6.

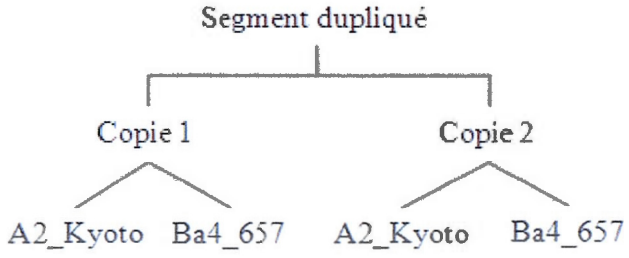
3.5 Solution pour 4 feuilles

Dans les travaux de Belcaid et al. (2011), on se sert des arbres-F pour calculer le nombre de mutations ponctuelles d'un étiquetage parcimonieux d'un arbre de partage d'histoire duplicative où on compte 4 feuilles et où les feuilles sont annotées avec des nucléotides. Les auteurs génèrent ensuite la totalité des étiquetages parcimonieux par énumération exhaustive et calculent la moyenne des mutations pré-spéciation sur cet ensemble d'étiquetages. Nous détaillons dans cette section les étapes de leur méthode.

Rappelons que, dans cette expérience, on analyse deux séquences codantes de TMP identifiées dans les espèces de phage A2_Kyoto et Ba4_657. Puisqu'il n'y que deux séquences, la phylogénie est nécessairement la suivante :



Et on a l'arbre de partage d'histoire duplicative suivant :



Les segments issus de la duplication la plus récente sont à la même position p sur chacune des séquences. L'exemple de la Figure 3.11 en est où la longueur des segments k est 3. Les deux segments adjacents sur A2_Kyoto sont `ATG` et `CTG` et `ATC` et `TGG` sur Ba4_657.

On annote les feuilles de l'arbre de partage d'histoire duplicative avec les listes de nucléotides issus de la duplication la plus récente. Les feuilles à gauche sont les nucléotides

A2_Kyoto	...	ATG	CTG	...
Ba4_657	...	ATC	TGG	...

Figure 3.11: Exemple de segments issus de la duplication la plus récente sur 2 séquences

orthologues du segment 1 et les feuilles à droite sont leurs paralogues, soit k positions plus loin. Les listes de nucléotides pour l'exemple de la Figure 3.11 sont `AACT`, `TTTG` et `GCGG`.

Une fois les feuilles annotées, on construit l'arbre-F correspondant. Cette procédure établit combien de mutations compte un étiquetage parcimonieux. Par exemple, avec la première liste de nucléotides de la Figure 3.11, on a l'arbre-F suivant, où on compte 2 unions :

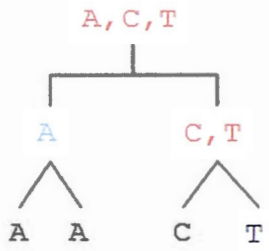


Figure 3.12: Arbre-F correspondant à la première liste de nucléotides des segments de la Figure 3.11.

Par contre, comme on a vu dans la section précédente, la procédure de Fitch ne génère pas tous les étiquetages parcimonieux. Heureusement, puisqu'on ne travaille que sur deux séquences, on peut calculer tous les étiquetages parcimonieux par inspection en s'assurant qu'ils comptent bel et bien le nombre d'unions calculé lors de la construction de l'arbre-F. Les étiquetages parcimonieux de la Figure 3.13 correspondent à la liste de nucléotides `AACT`.

On ne peut pas générer l'étiquetage en haut à droite avec la procédure de Fitch. En

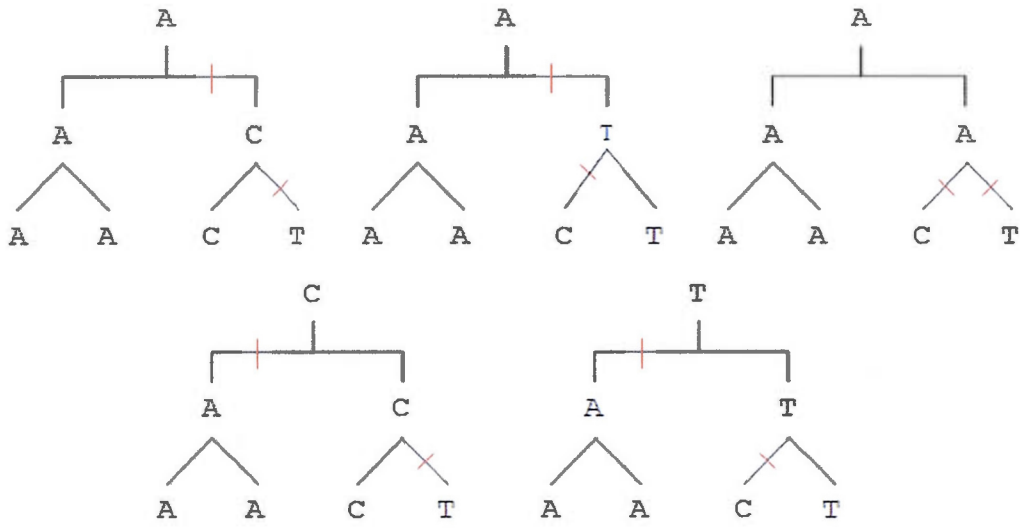


Figure 3.13: Les 5 étiquetages parcimonieux de la liste de nucléotides AACT.

effet, le nucléotide A ne se trouve pas dans l'ensemble de Fitch {C,T} de l'enfant droit de la racine de l'arbre-F de la Figure 3.12. Ces étiquetages ont été obtenus par énumération exhaustive.

Puisqu'il existe généralement plusieurs étiquetages parcimonieux pour une liste de feuilles et que chacun de ces étiquetages ne compte pas nécessaire le même nombre de mutations pré-spéciation, comme on peut le constater sur la Figure 3.13, il devient logique de calculer la *moyenne du nombre de mutations pré-spéciation.*

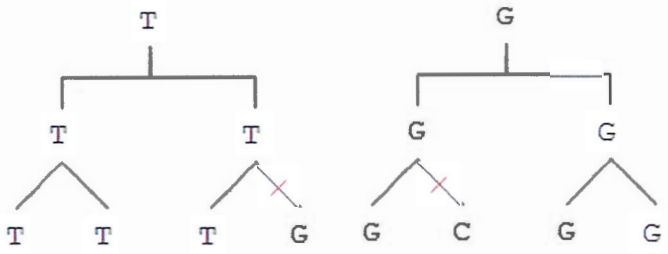
Définition 6. La *moyenne du nombre de mutations pré-spéciation d'une liste de nucléotides issus de la duplication la plus récente* est la somme des mutations pré-spéciation sur tous les étiquetages parcimonieux divisée par le nombre d'étiquetages parcimonieux de l'arbre de partage d'histoire duplicative où les feuilles sont annotées par les nucléotides de cette liste .

Dans les étiquetages de la Figure 3.13, on a au total 4 mutations pré-spéciation et 5 étiquetages parcimonieux. La moyenne est donc 4/5. Notons que ce qui nous intéresse particulièrement est la moyenne des mutations pré-spéciation des **segments** issus de la

duplication la plus récente.

Définition 7. La *moyenne du nombre de mutations pré-spéciation des segments issus de la duplication la plus récente* est la somme des moyennes du nombre de mutations pré-spéciation pour chacune des listes de nucléotides issus de cette duplication normalisée sur la longueur des segments.

Avec les segments de la Figure 3.11, nous avons les 3 listes de feuilles `AACT`, `TTTG` et `GCGG`. Nous avons calculé la moyenne la première liste de feuilles. Les deux dernières listes ont chacune un seul étiquetage parcimonieux. Il compte une mutation ponctuelle et elle se situe après la spéciation. Les étiquetages pour les listes de feuilles `TTTG` et `GCGG` sont présentés ci-dessous.



On a donc une somme de $4/5+0+0 = 4/5$ pour les segments de la Figure 3.11. Puisque $k = 3$, la moyenne du nombre de mutations pré-spéciation pour ces segments est $4/15$.

Il existe $4^4 = 256$ listes de nucléotides possibles lorsqu'on traite deux séquences en parallèle. Dans les travaux de Belcaid et al., les auteurs regroupent les listes de nucléotides en 7 *classes* où chaque classe correspond à un ensemble de listes où les étiquetages parcimonieux sont similaires. Cette technique permet de calculer rapidement la moyenne des mutations d'une liste de nucléotides en déterminant à quelle classe elle appartient.

Dans les travaux de Belcaid et al., on utilise la méthode présentée dans cette section pour identifier la duplication la plus récente.

3.5.1 Résultats des calculs de la moyenne des mutations pré-spéciation de Belcaid et al. (2011)

Dans l'expérience de Belcaid et al.(2011), on calcule la moyenne des mutations pré-spéciation pour toutes les positions p possibles où la longueur des segments k est 33 nucléotides. La courbe en vert du tableau de la figure 3.14 montre ces résultats. Elle unifie les données des deux autres courbes. Les valeurs minimales se regroupent toutes autour du centième nucléotide, ce qui se rapproche des attentes initiales.

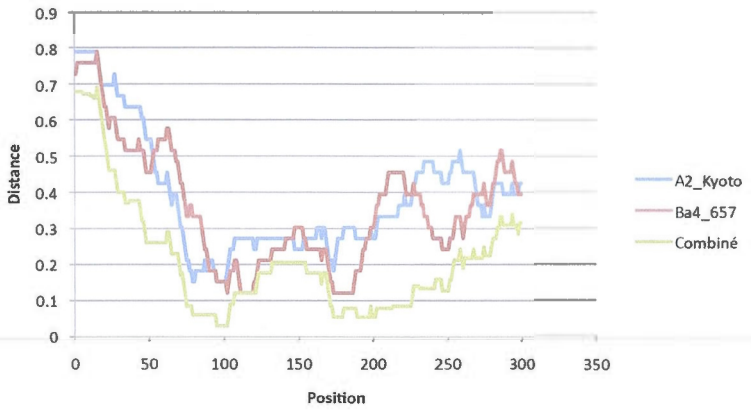


Figure 3.14: Courbes des distances de Hamming et des moyennes du nombre de mutations pré-spéciation normalisées obtenues par Belcaid et al. (2011).

Dans la prochaine section, on voit comment étendre le calcul de la moyenne du nombre de mutations pré-spéciation à plusieurs séquences en parallèle.

3.6 Algorithmes de Sankoff

Les algorithmes de Sankof (1975) nous permettent d'automatiser la génération de tous les étiquetages parcimonieux pour un arbre où les feuilles sont annotées avec une liste de nucléotides pour un nombre quelconque de feuilles.

Avertissement : Les pages suivantes sont plutôt techniques, tout en constituant l'essentiel de notre contribution sur le plan algorithmique. En première lecture, le lecteur qui

a confiance en notre capacité de calculer le nombre moyen de mutations pré-spéciation peut passer directement au Chapitre 4.

Ces algorithmes sont des généralisations de ceux de Fitch où on calcule des coûts de mutations sur des états plutôt que des nombres de mutations. L'ensemble des états est noté E. Dans notre contexte, cet ensemble correspond à l'ensemble des nucléotides, soit $E = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Le coût d'une mutation d'un état vers un autre est noté $m(e_1, e_2)$. Dans ce mémoire, nous allons utiliser la fonction de coût suivante :

$$m(e_1, e_2) = \begin{cases} 0 & \text{si } e_1 = e_2 \\ 1 & \text{sinon} \end{cases}$$

Ce qui revient à compter le nombre de mutations.

Supposons la topologie illustrée à la Figure 3.15.

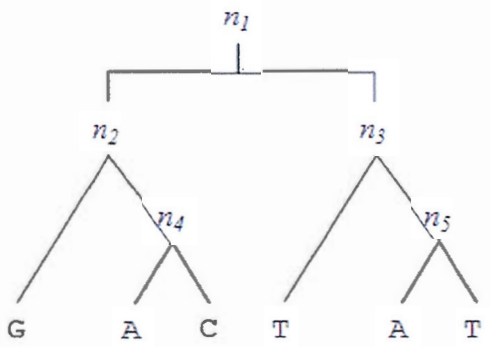


Figure 3.15: Exemple de topologie où les feuilles sont annotées.

On appelle un étiquetage sous un noeud n un étiquetage du sous-arbre dont la racine est le noeud n. Un étiquetage sous le noeud n_1 correspond à un étiquetage de l'arbre entier. Un étiquetage sous le noeud n_2 correspond à un étiquetage du sous-arbre où n_2 est la racine, soit un étiquetage où les noeuds n_2 et n_4 sont étiquetés. La Figure 3.16 présente un étiquetage sous le noeud n_1 de la Figure 3.15 à gauche et un étiquetage sous le noeud n_2 à droite.

Le *coût d'un étiquetage* est la somme des coûts de mutations pour chaque arête reliant

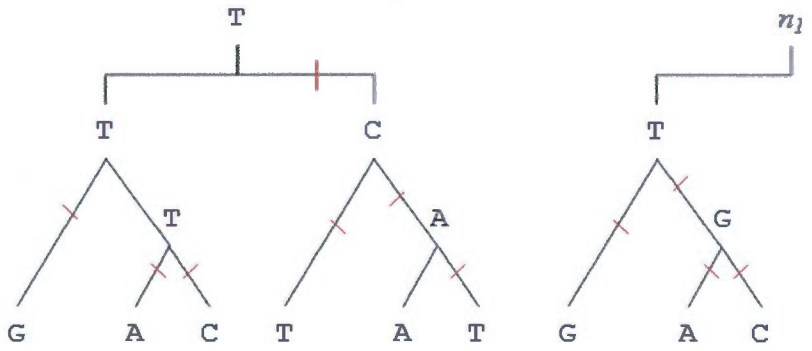


Figure 3.16: Exemple d'étiquetages sous le noeud n_1 de la Figure 3.15 à gauche et sous le noeud n_2 à droite.

deux états différents dans l'étiquetage. Le coût de l'étiquetage à gauche de la Figure 3.16 est 7 et celui de l'étiquetage à droite est 4.

On s'intéresse particulièrement à calculer des coûts d'étiquetages parcimonieux sous un noeud n où on force l'étiquetage du noeud avec un nucléotide e. On pose donc 2 contraintes sur l'étiquetage sous le noeud n :

1. L'étiquetage est parcimonieux.
2. Le noeud n est étiqueté avec le nucléotide e.

L'exemple de la Figure 3.17 est un étiquetage parcimonieux sous le noeud n_2 de l'arbre de la Figure 3.15 où on force l'étiquetage de n_2 avec le nucléotide `T`. Le coût de cet étiquetage parcimonieux est 3.

L'Algorithme 4 suivant calcule sur un arbre dont les feuilles sont annotées par une liste de nucléotides les coûts d'étiquetages parcimonieux sous chacun des noeuds intermédiaires n où on force l'étiquetage de n avec chaque nucléotide e à tour de rôle. Par exemple, pour le noeud intermédiaire n_2 de la Figure 3.15, on veut connaître les coûts des étiquetages parcimonieux sous le noeud n_2 où le noeud n_2 est étiqueté par le nucléotide `A`, ensuite par `C`, ensuite par `G` et finalement par `T`. La Figure 3.18 montre ces étiquetages parcimonieux. On a donc un coût de 2 pour les étiquetages parcimonieux où n_2 est étiqueté par les

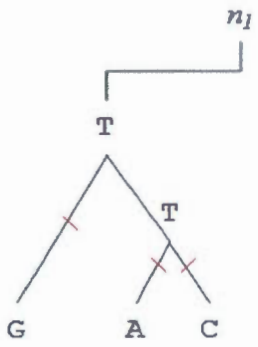


Figure 3.17: Étiquetage parcimonieux sous le noeud n_2 de l'arbre de la Figure 3.15 étiqueté par le nucléotide T.

nucléotides A, C et G et un coût de 3 lorsque le noeud n_2 est étiqueté par le nucléotide T.

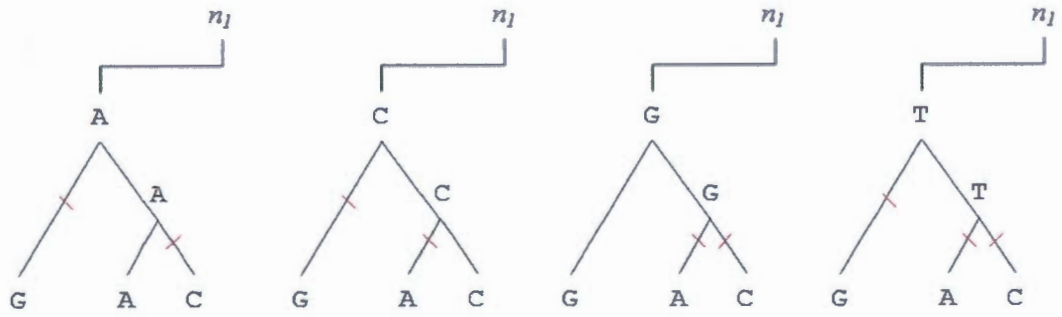


Figure 3.18: Étiquetages parcimonieux sous le noeud n_2 de l'arbre de la Figure 3.15 pour chaque nucléotide.

Algorithme 4 (Calcul des coûts d'étiquetages parcimonieux, Sankoff). Soit un ensemble d'états E, une fonction de coût des mutations d'états $m(e_1, e_2)$ et un arbre A dont les feuilles sont annotées par une liste de nucléotides. On assigne à chaque noeud n de l'arbre une valeur de coût pour chaque état e appartenant à l'ensemble E, noté $c(n, e)$. Cette valeur représente le coût d'un étiquetage parcimonieux sous le noeud n où on force l'étiquetage de n par le nucléotide e.

Le coût $c(n, e)$ pour les feuilles est 0 si e annote la feuille, ∞ sinon, car il n'existe tout simplement aucun d'étiquetage parcimonieux sous le noeud n s'il est étiqueté par un nucléotide différent de celui qui l'annote déjà.

La valeur de $c(n, e)$ pour un noeud intermédiaire est donnée par :

$$c(n, e) = \sum_{n' \text{ enfant de } n} c_{min}(e, n')$$

Où $c_{min}(e, n')$ représente le minimum du coût de mutation du nucléotide e vers chacun des nucléotides e' additionné du coût d'un étiquetage parcimonieux sous le noeud enfant n' où le le noeud n' est étiqueté par e' :

$$c_{min}(e, n') = \min_{e' \in E} \{m(e, e') + c(n', e')\}$$

On appelle un arbre où chaque noeud contient les coûts $c(n, e)$ pour chaque nucléotide un *arbre de Sankoff*.

Exemple. Nous allons reprendre la topologie de la Figure 3.15 pour mettre en pratique l'Algorithme 4. On commence par calculer les coûts pour chacune des feuilles. Si e est le nucléotide qui annote la feuille, alors son coût est 0, ∞ sinon. On obtient le résultat à la Figure 3.19.

On monte ensuite d'un niveau pour calculer les coûts $c(n, e)$ pour les noeuds dont les deux enfants sont des feuilles, soit les noeuds n_4 et n_5. On veut calculer les coûts $c(n_4, e)$ et $c(n_5, e)$ pour chaque nucléotide e. Or, cette valeur est donnée par la somme des coûts $c_{min}(e, n')$ pour chaque enfant n'. Puisqu'on a deux enfants, on peut immédiatement noter que les coûts $c(n, e)$ sont les sommes, pour chaque nucléotide e, $c_{min}(e, n_g) + c_{min}(e, n_d)$ où n_g est l'enfant de gauche et n_d est l'enfant de droite.

Commençons par calculer $c(n_4, \mathtt{A})$. On calcule les valeurs suivantes pour son enfant gauche, soit la feuille n_g annotée par **A** :

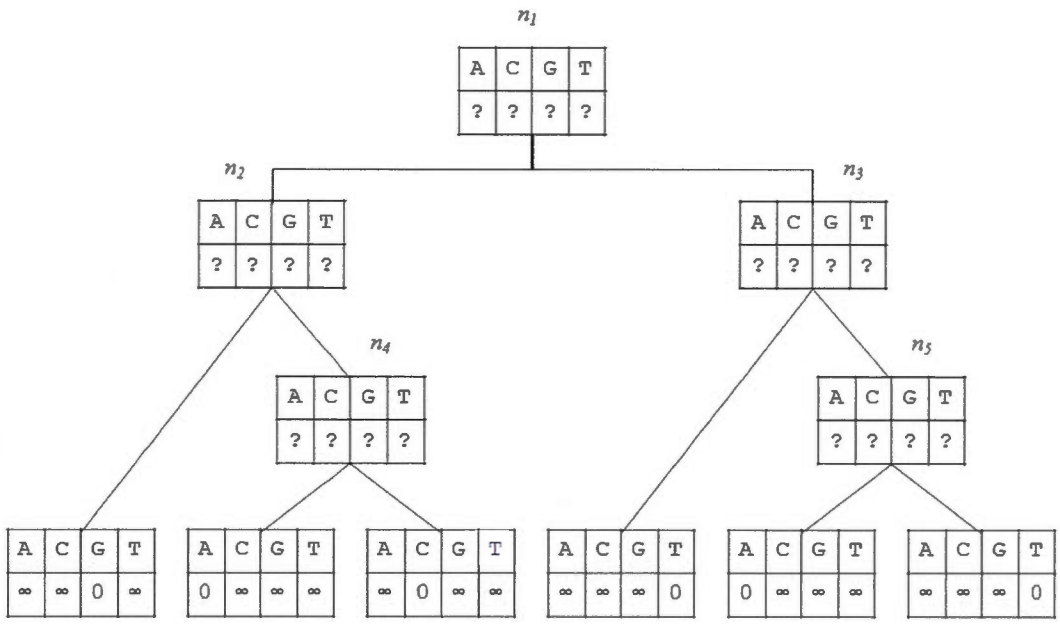


Figure 3.19: Première étape de l'exemple de l'Algorithme 4.

e'	$m(\mathtt{A}, e')$	$c(n_g, e')$	$m(\mathtt{A}, e') + c(n_g, e')$	$c_{min}(\mathtt{A}, n_g)$
A	0	0	0	0
C	1	∞	∞	
G	1	∞	∞	
T	1	∞	∞	

Notons que, puisque les nucléotides C, G et T ont des valeurs de coûts $c(n_g, e')$ infinies, la somme $m(e, e') + c(n_g, e')$ sera aussi infinie et ce, pour n'importe quel nucléotide e. Le coût $c_{min}(e, n_g)$ est donc toujours la valeur de la somme du coût d'une mutation vers le nucléotide A et de son coût dans la table de la feuille, soit 0. Cette somme est calculée avec l'expression suivante :

$$c_{min}(e, n_g) = m(e, \mathtt{A}) + c(n_g, \mathtt{A}) = m(e, \mathtt{A}) + 0 = m(e, \mathtt{A})$$

On peut donc rapidement calculer les coûts suivants pour les nucléotides e autres que A :

$$\begin{aligned} c_{min}(\mathtt{C}, n_g) &= m(\mathtt{C}, \mathtt{A}) &= 1 \\ c_{min}(\mathtt{G}, n_g) &= m(\mathtt{G}, \mathtt{A}) &= 1 \\ c_{min}(\mathtt{T}, n_g) &= m(\mathtt{T}, \mathtt{A}) &= 1 \end{aligned}$$

Le calcul pour l'enfant droit du noeud n, soit la feuille n_d annotée par C est similaire, à la différence que $c_{min}(e, n_d)$ équivaut au coût de la mutation du nucléotide e vers le nucléotide C $m(e, \mathtt{C})$ au lieu du coût de la mutation vers le nucléotide A. On obtient les coûts suivants :

$$\begin{aligned} c_{min}(\mathtt{A}, n_d) &= m(\mathtt{A}, \mathtt{C}) &= 1 \\ c_{min}(\mathtt{C}, n_d) &= m(\mathtt{C}, \mathtt{C}) &= 0 \\ c_{min}(\mathtt{G}, n_d) &= m(\mathtt{G}, \mathtt{C}) &= 1 \\ c_{min}(\mathtt{T}, n_d) &= m(\mathtt{T}, \mathtt{C}) &= 1 \end{aligned}$$

On peut maintenant calculer les coûts $c(n_4, e)$ pour chaque nucléotide e en faisant la somme $c_{min}(e, n_g) + c_{min}(e, n_d)$. Les résultats sont ci-dessous :

$$\begin{aligned} c(n_4, \mathtt{A}) &= c_{min}(\mathtt{A}, n_g) + c_{min}(\mathtt{A}, n_d) &= 0+1 &= 1 \\ c(n_4, \mathtt{C}) &= c_{min}(\mathtt{C}, n_g) + c_{min}(\mathtt{C}, n_d) &= 1+0 &= 1 \\ c(n_4, \mathtt{G}) &= c_{min}(\mathtt{G}, n_g) + c_{min}(\mathtt{G}, n_d) &= 1+1 &= 2 \\ c(n_4, \mathtt{T}) &= c_{min}(\mathtt{T}, n_g) + c_{min}(\mathtt{T}, n_d) &= 1+1 &= 2 \end{aligned}$$

Le calcul pour le noeud n_5 parent des deux feuilles A et T est analogue. L'arbre résultant est à la Figure 3.20.

Nous pouvons maintenant passer au niveau supérieur, soit celui des noeuds n_2 et n_3. Chacun de ces noeuds est parent d'une feuille et d'un noeud intermédiaire. Les calculs pour leurs enfants feuilles se font rapidement, comme précédemment.

Prenons le noeud n_2 et son enfant de gauche, soit la feuille n_g annotée par le nucléotide G. On calcule les valeurs suivantes :

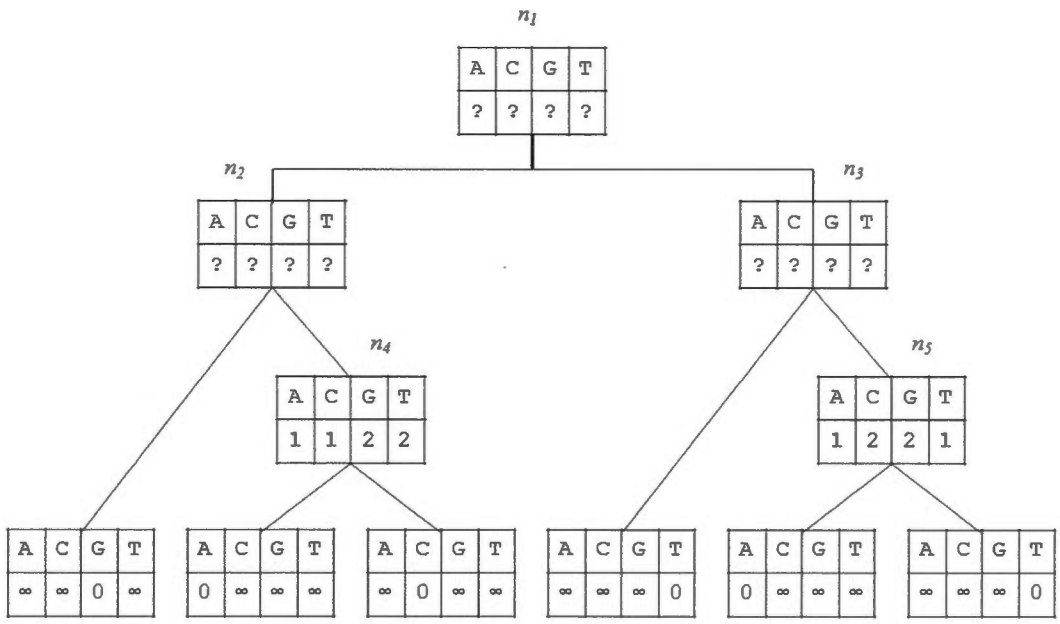


Figure 3.20: Deuxième étape de l'exemple de l'Algorithme 4.

$$
\begin{aligned}
c_{min}(\mathtt{A}, n_g) &= m(\mathtt{A}, \mathtt{G}) &= 1 \\
c_{min}(\mathtt{C}, n_g) &= m(\mathtt{C}, \mathtt{G}) &= 1 \\
c_{min}(\mathtt{G}, n_g) &= m(\mathtt{G}, \mathtt{G}) &= 0 \\
c_{min}(\mathtt{T}, n_g) &= m(\mathtt{T}, \mathtt{G}) &= 1
\end{aligned}
$$

Les calculs se compliquent légèrement pour l'enfant de droite n_d. Nous devons tester, pour chaque nucléotide e, la valeur de la somme $m(e, e') + c(n_d, e')$ et ce, pour chaque nucléotide e' pour déterminer le minimum. Posons e est le nucléotide A. Alors, on calcule les coûts suivants :

e'	$m(\mathtt{A}, e')$	$c(n_d, e')$	$m(\mathtt{A}, e') + c(n_d, e')$	$c_{min}(\mathtt{A}, n_d)$
A	0	1	1	1
C	1	1	2	
G	1	2	3	
T	1	2	3	

Si e est le nucléotide C, on calcule les coûts suivants :

e'	$m(\mathtt{C}, e')$	$c(n_d, e')$	$m(\mathtt{C}, e') + c(n_d, e')$	$c_{min}(\mathtt{C}, n_d)$
A	1	1	2	1
C	0	1	1	
G	1	2	3	
T	1	2	3	

Pour le nucléotide G on a :

e'	$m(\mathtt{G}, e')$	$c(n_d, e')$	$m(\mathtt{G}, e') + c(n_d, e')$	$c_{min}(\mathtt{G}, n_d)$
A	1	1	2	2
C	1	1	2	
G	0	2	2	
T	1	2	3	

Finalement, pour le nucléotide T on a :

e'	$m(\mathtt{T}, e')$	$c(n_d, e')$	$m(\mathtt{T}, e') + c(n_d, e')$	$c_{min}(\mathtt{T}, n_d)$
A	1	1	2	2
C	1	1	2	
G	1	2	3	
T	0	2	2	

On est maintenant en mesure de calculer tous les coûts $c(n_2, e)$. Ils sont montrés ci-dessous :

$$
\begin{aligned}
c(n_4, \mathtt{A}) &= c_{min}(\mathtt{A}, n_g) + c_{min}(\mathtt{A}, n_d) &&= 1+1 &&= 2 \\
c(n_4, \mathtt{C}) &= c_{min}(\mathtt{C}, n_g) + c_{min}(\mathtt{C}, n_d) &&= 1+1 &&= 2 \\
c(n_4, \mathtt{G}) &= c_{min}(\mathtt{G}, n_g) + c_{min}(\mathtt{G}, n_d) &&= 0+2 &&= 2 \\
c(n_4, \mathtt{T}) &= c_{min}(\mathtt{T}, n_g) + c_{min}(\mathtt{T}, n_d) &&= 1+2 &&= 3
\end{aligned}
$$

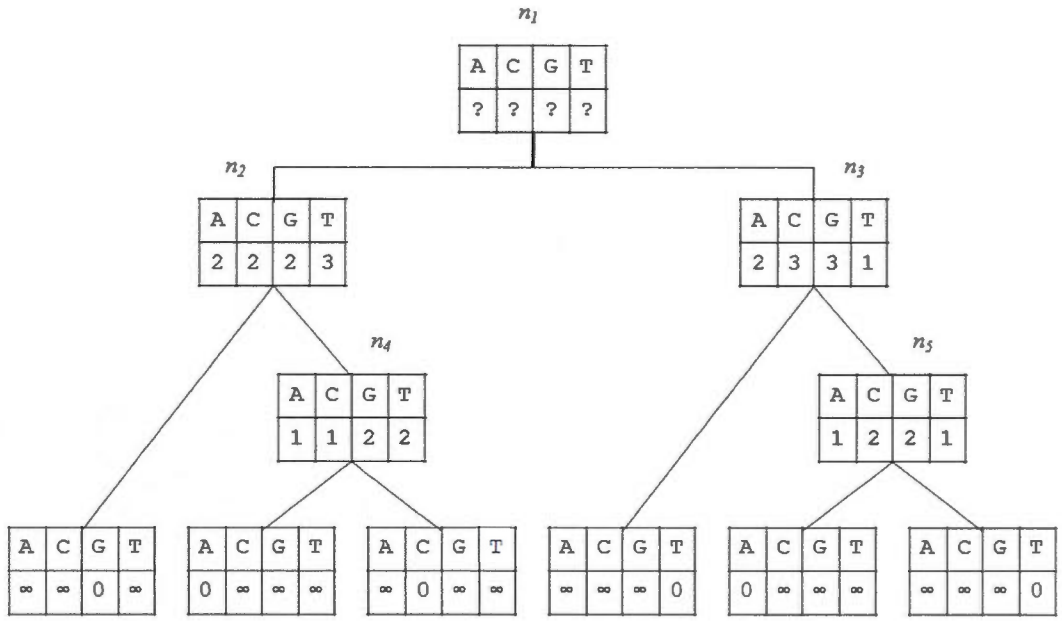


Figure 3.21: Troisième étape de l'exemple de l'Algorithme 4.

Le calcul pour le noeud n_3 est analogue. L'arbre résultant est présenté à la Figure 3.21.

Nous n'allons pas donner les calculs pour la racine ici à des fins de concision. Les coûts pour chacun de ces enfants sont calculés comme on a fait pour l'enfant droit de n_2. Le résultat final est présenté à la Figure 3.22.

Proposition 4. Les coûts $c(n, e)$ calculés par l'Algorithme 4 pour n'importe quel noeud n sont les coûts des étiquetages parcimonieux sous ce noeud étiqueté par chacun des nucléotides e.

Preuve. La proposition est vraie pour les feuilles. Pour tous nucléotides e, si e est le nucléotide qui annote la feuille, un seul étiquetage parcimonieux sous cette feuille est possible et il ne compte pas de mutation. Par contre, si e n'annote pas la feuille, alors il n'y a pas d'étiquetages possible, parcimonieux ou non.

Posons l'hypothèse inductive que la proposition est vraie pour chacun des noeuds enfants

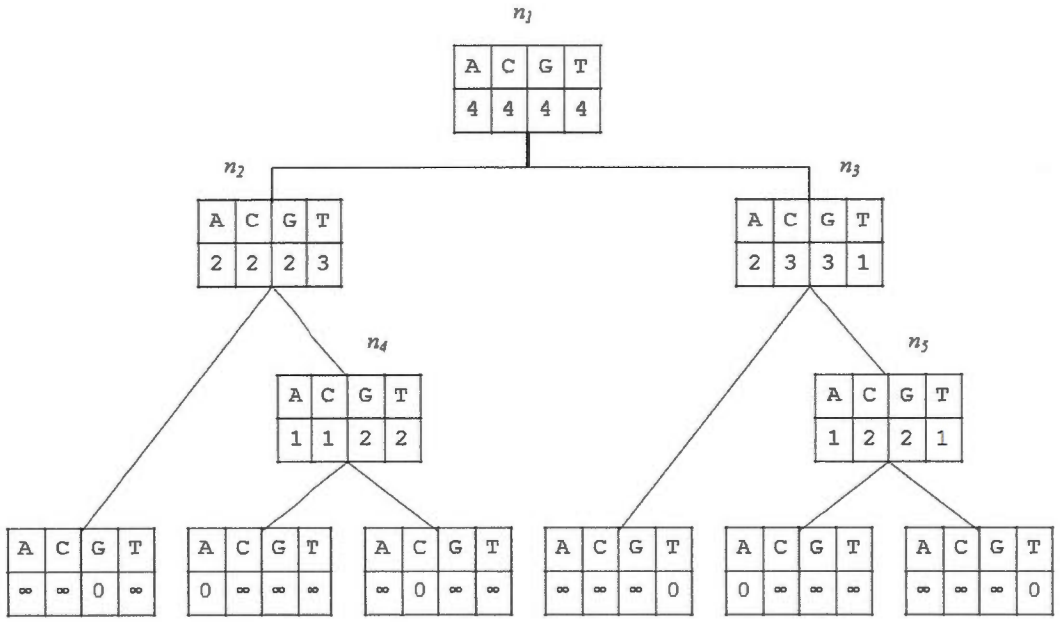


Figure 3.22: Arbre de Sankoff correspondant à l'arbre de la Figure 3.15 où tous les coûts ont été calculés.

n' d'un noeud n. En conséquence de l'hypothèse inductive, le coût d'un étiquetage parcimonieux sous un enfant étiqueté par un nucléotide e' est $c(n', e')$. Puisque les nucléotides avec lesquels ont peut étiqueter les enfants du noeud n peuvent varier de façon indépendante entre eux dans les étiquetages, on peut traiter chaque noeud enfant n' séparément.

Si n est étiqueté par e et n' est étiqueté par e', alors le coût d'un étiquetage parcimonieux sous le noeud n dans ces contraintes est la somme du coût de mutation $m(e, e')$ et du coût de l'étiquetage parcimonieux $c(n', e')$. On cherche les nucléotides e' qui minimisent cette somme. Or, on note cette valeur $c_{min}(e, n') = \min_{e' \in E}\{m(e, e') + c(n', e')\}$.

Puisqu'on teste tous les nucléotides e', il n'existe pas d'étiquetage qui puisse avoir un coût inférieur.

Les enfants peuvent être étiquetés de façon indépendante l'un de l'autre dans un étiquetage

parcimonieux sous le noeud n. Alors, le coût de cet étiquetage parcimonieux est la somme des coûts $c_{min}(e, n')$ pour chacun de ses enfants, soit :

$$c(n, e) = \sum_{n' \text{ enfant de } n} c_{min}(e, n')$$

□

Les valeurs de coûts $c(n, e)$ pour la racine correspondent aux coûts des étiquetages parcimonieux de l'arbre entier lorsqu'on force l'étiquetage de la racine par le nucléotide e. Le coût des étiquetages parcimonieux de l'arbre, c'est-à-dire lorsqu'on est libre de choisir le nucléotide avec lequel on étiquette la racine, est la valeur de coût $c(n, e)$ minimale parmi toutes les valeurs de la racine.

Avec l'exemple à la Figure 3.22, ce minimum est 4. Tous les nucléotides sont éligibles pour étiqueter la racine dans un étiquetage parcimonieux de l'arbre, car ils ont tous un coût $c(n, e)$ de 4. Cependant, tel n'est pas toujours le cas. Par exemple, la Figure 3.23 présente deux arbres de Sankoff correspondant à deux arbres dont les feuilles sont annotées par les listes de nucléotides `AACT` et `ACCT`. Seuls les nucléotides `A`, `C` et `T` peuvent étiqueter la racine de l'arbre à gauche dans un étiquetage parcimonieux, car le minimum des valeurs de coûts de la racine est 2. Seul le nucléotide `C` peut étiqueter la racine de l'arbre à droite dans un étiquetage parcimonieux, car il est le seul à posséder le coût minimal 2.

On note E_r l'ensemble des nucléotides candidats pour étiqueter la racine r dans un étiquetage parcimonieux de l'arbre tel que :

$$E_r = \left\{ e_r : c(r, e_r) = \min_{e \in E} \{c(r, e)\} \right\}$$

On peut générer des étiquetages parcimonieux en suivant la procédure suivante :

Algorithme 5 (Génération d'étiquetages parcimonieux, Sankoff). Annoter la racine avec un nucléotide e tel que $e \in E_r$. Pour chaque noeud intermédiaire n' enfant d'un

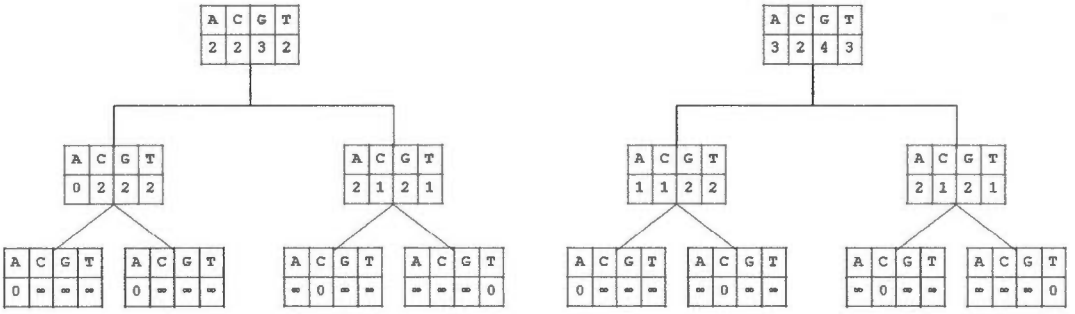


Figure 3.23: Exemple où les racines ne peuvent pas être étiquetées par n'importe quel nucléotide dans un étiquetage parcimonieux.

noeud n déjà étiqueté, étiqueter n' avec un nucléotide e' tel que $m(e, e') + c(n', e') = c_{min}(e, n')$.

Exemple. On reprend l'arbre de Sankoff de la Figure 3.22 pour générer un étiquetage parcimonieux. Le minimum des coûts de la racine est 4, soit la même valeur que le nombre d'unions qu'on a calculé sur l'arbre-F de la Figure 3.9. N'importe quel nucléotide est éligible pour annoter la racine. On choisit le nucléotide T. Pour son enfant gauche n_2, on choisit un nucléotide e' tel que $m(\mathtt{T}, e') + c(n_2, e') = c_{min}(\mathtt{T}, n_2)$. Ce coût $c_{min}(\mathtt{T}, n_2)$ est 3. N'importe quel nucléotide satisfait ce coût. On choisit T. Pour l'enfant droit n_3 de la racine, un seul nucléotide satisfait le coût $c_{min}(\mathtt{T}, n_3)$, qui est égal à 1, soit le nucléotide T.

On a 3 choix pour le noeud n_4, soit on garde T, soit on opte pour une mutation vers les nucléotides A ou C. On choisit C. Un seul nucléotide peut étiqueter le noeud n_5, soit T. L'étiquetage correspondant est illustré à la Figure 3.24.

Cet étiquetage est celui qui a été manqué par la procédure de Fitch présenté à la Figure 3.10.

Proposition 5. L'Algorithme 5 génère tous les étiquetages parcimonieux.

Preuve. La Proposition 5 est une conséquence de la Proposition 4. Le coût d'un étiquetage

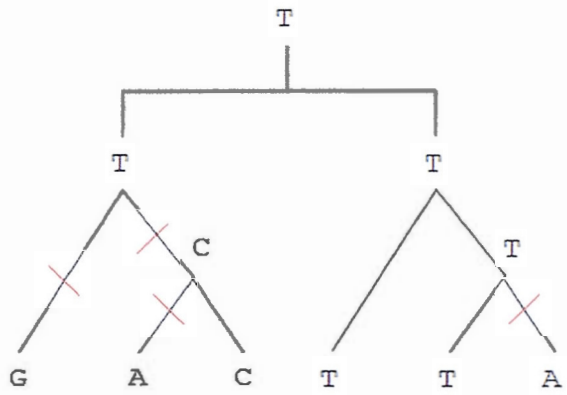


Figure 3.24: Exemple d'étiquetage parcimonieux généré avec l'Algorithme 5.

parcimonieux est le minimum parmi les coûts de la racine. N'importe quel nucléotide qui a ce coût et est donc candidat pour annoter la racine.

On a vu que chacun des enfants doit être étiqueté avec un nucléotide e' qui minimise le coût de mutation $m(e, e')$ du nucléotide e du noeud parent additionné du coût d'un étiquetage parcimonieux sous l'enfant étiqueté par e', soit $c(n', e')$.

Ce coût est $c_{min}(e, n')$ et il a été calculé en testant tous les nucléotides possibles. N'importe quel nucléotide e' qui satisfait l'expression $m(e, e') + c(n', e') = c_{min}(e, n')$ est donc éligible pour annoter le noeud enfant n' dans un étiquetage parcimonieux. □

Le calcul du coût moyen de mutations pré-spéciation pour une liste de nucléotides se fait comme auparavant : on annote les feuilles de l'arbre de partage d'histoire duplicative correspondant avec les nucléotides de cette liste, on calcule l'arbre de Sankoff, on génère les étiquetages parcimonieux avec l'Algorithme 5, on fait la somme des coûts de mutations pré-spéciation sur chacun des étiquetages parcimonieux et on divise par le nombre d'étiquetages parcimonieux.

Proposition 6. Le complexité temporelle du calcul du coût moyen de mutations pré-spéciation avec les algorithmes de Sankoff est $O(F)$ où F est le nombre de feuilles.

Preuve. Nous allons compter le nombre total d'opérations effectuées par l'algorithme. Pour ce faire, nous allons diviser le travail en deux : compter le nombre d'opérations pour calculer les coûts des nucléotides dans les noeuds et compter les opérations pour générer l'ensemble des étiquetages parcimonieux.

L'initialisation du coût des feuilles est donnée par le nombre de feuilles multiplié par 4, soit le nombre de nucléotides. Pour F feuilles, l'initialisation se fait en $4F$ opérations.

Les comparaisons pour calculer les coûts des nucléotides se font sur les noeuds intermédiaires. Puisqu'on a 2 enfants par noeud, on a $F-1$ noeuds intermédiaires. À chaque noeud, on fait un calcul pour les 4 nucléotides. Pour chacun des nucléotides, on traite chacun des deux enfants. Pour chaque enfant n', on compare les 4 valeurs de la somme $m(e,e')+c(n',e')$, soit pour chaque nucléotide, pour déterminer la valeur minimale $c_{min}(e,n')$. En supposant que cette comparaison compte pour 4 opérations, on a cette expression qui représente la somme du travail pour les noeuds intermédiaires :

$$(F-1)\cdot 4\cdot 2\cdot 4 = 32F-32$$

Au total, le calcul de tous les coûts $c(n,e)$, incluant les feuilles, compte $36F-32$ opérations.

Pour évaluer le nombre d'opérations pour générer l'ensemble des étiquetages parcimonieux, imaginons le cas où chacun des 4 nucléotides est éligible pour annoter chaque noeud dans les étiquetages parcimonieux. Ce scénario est possible si le coût des mutations $m(e,e')=0$ pour n'importe quel nucléotide e et e'. On suppose que $c_{min}(e,n')$ est déjà connu pour chaque noeud n. Pour chaque nucléotide candidat à étiqueter un noeud n et pour chacun des deux enfants on fait 4 comparaisons pour déterminer quels sont les candidats pour étiqueter ces noeuds enfants. Dans notre scénario, les 4 nucléotides sont candidats pour chaque noeud. On fait ce calcul pour chacun des $F-1$ noeuds intermédiaires. On a donc ce nombre d'opérations :

$$(F-1)\cdot 4\cdot 2\cdot 4 = 32F-32$$

Au total, en additionnant le nombre d'opérations pour calculer les coûts $c(n,e)$ et le

nombre d'opérations pour la génération des étiquetages, on a $36F - 32 + 32F - 32 = 68F - 64$ opérations. Le calcul du coût moyen de mutations pré-spéciation a donc une complexité temporelle de $\mathrm{O}(F)$. □

Dans notre contexte, le nombre de feuilles F de l'arbre sera toujours 2 fois le nombre N de séquences, soit $2N$. On peut alors dire que la complexité temporelle du calcul du coût moyen des mutations pré-spéciation est $\mathrm{O}(N)$.

CHAPITRE IV

IMPLÉMENTATIONS ET RÉSULTATS

Ce chapitre discute de l'implémentation de l'algorithme du calcul du coût moyen de mutations pré-spéciation, des optimisations utilisées pour accélérer ces calculs, de leurs performances et des résultats obtenus avec de vraies séquences biologiques.

Trois optimisations sont présentées, en plus de l'algorithme standard. Nous discutons du fonctionnement et de la complexité temporelle de chacun de ces algorithmes.

Suit un comparatif des performances de nos implémentations des 4 algorithmes avec plusieurs ensembles de séquences aléatoires. Nous terminons ce chapitre avec une expérience impliquant 5 séquences biologiques partageant leur histoire duplicative identifiées dans le travail de Nicolas Massoulier (2013). On calcule des courbes de distances de Hamming normalisées pour chacune des séquences et on les compare à celle obtenue en calculant les coûts moyens de mutations pré-spéciation pour les 5 séquences en parallèle.

4.1 Algorithme standard

On présente ici l'heuristique de Benson et Dong (1999) adaptée avec la nouvelle fonction de coût moyen de mutations pré-spéciation.

La première étape est de construire la topologie de l'arbre de partage d'histoire duplicative des séquences S à partir de la phylogénie *phylo* fournie à l'algorithme.

Comme dans l'Algorithme 1, la boucle externe génère toutes les longueurs k possibles,

en allant de la borne minimale à la borne maximale avec un pas de 3.

Dans la boucle intermédiaire, on itère sur toutes les positions p possibles où des paires de segments adjacents de longueur k peuvent débuter sur chacune des séquences. Comme dans l'Algorithme 1, la position maximale est $l - 2k$ en posant que la première position est 0.

Pour chaque position p_n à l'intérieur du premier segment, soit pour chaque valeur de p_n allant de p à $p + k - 1$, on extrait la liste de nucléotides formée des ceux localisés sur cette position sur chacune des séquences et des nucléotides k positions plus loin. On annote ensuite les feuilles de l'arbre de partage de l'histoire duplicative avec cette liste.

On calcule la moyenne de mutations pré-spéciation pour chacune des listes de nucléotides des segments. On fait la somme de ces valeurs et on normalise finalement sur k. L'Algorithme 6 détaille la procédure.

Nous avons vu que le problème est d'identifier les segments issus de la duplication la plus récente. En réalité, il est intéressant d'obtenir un certain nombre R des meilleurs résultats pour pouvoir les interpréter. Un résultat se définit comme un tuple avec une position p, une longueur de segments k et le coût moyen d_n de mutations pré-spéciation calculé pour ces deux valeurs.

Ces résultats sont ordonnés avec une structure dont la taille est bornée à R et qui permet d'accéder rapidement, à tout moment, au pire des résultats emmagasinés. Lorsqu'on calcule un nouveau résultat, on ne le garde en mémoire que s'il est meilleur que le pire des résultats connus.

La fonction d'ordonnancement compare d'abord les coûts d_n des deux résultats : la plus petite valeur l'emporte. S'il advient que ces deux distances sont identiques, alors le résultat qui a la longueur k la plus grande l'emporte. Dans le cas où il y égalité de ces deux valeurs, alors un classement arbitraire est effectué avec la position p : la plus proche de 0 l'emporte.

Algorithme 6 *algoStandard*$(S, phylo, minK, maxK, R, m)$

1: $l \leftarrow$ longueur des séquences S
2: $N \leftarrow$ nombre de séquences S
3: $resultats \leftarrow Resultats(R)$ ▷ Structure gardant les R meilleurs résultats.
4: $arbrePartage \leftarrow$ l'arbre de partage d'histoire duplicative correspondant à *phylo*
5: **pour** $k \leftarrow minK$ à $maxK$ avec un pas de 3 **faire**
6: **pour** $p \leftarrow 0$ à $l - (2k)$ **faire**
7: $d \leftarrow 0$
8: **pour** $p_n \leftarrow p$ à $p + k - 1$ **faire**
9: $liste \leftarrow$ la liste des nucléotides aux positions p_n et $p_n + k$
10: annoter les feuilles de *arbrePartage* avec *liste*
11: $d_p \leftarrow$ coût moyen de mutations pré-spéciation de *arbrePartage* avec m
12: $d \leftarrow d + d_p$
13: **fin pour**
14: $d_n \leftarrow d/k$
15: $resultat \leftarrow (p, k, d_n)$
16: ajouter *resultat* dans *resultats*
17: **fin pour**
18: **fin pour**
retourner *resultats*

Il peut être intéressant de borner les longueurs minimales et maximales de k. En effet, si on soupçonne la longueur des segments dupliqués d'être de 30 nucléotides, par exemple, calculer les distances pour toutes les valeurs de k peut être un gaspillage de ressources. Ainsi, les bornes $minK$ et $maxK$ sont fournies à nos algorithmes.

Rappelons qu'avec l'Hypothèse 2, seules les longueurs k multiples de 3 nous intéressent. On suppose que les bornes fournies aux programmes sont des multiples de 3 et qu'elles sont valides, c'est-à-dire que $minK \leq maxK$ et $maxK \leq \lfloor l/2 \rfloor$

La fonction du coût des mutations m pour calculer les coûts moyens de mutations

pré-spéciation est aussi fournie à nos algorithmes.

Proposition 7. La complexité temporelle de l'Algorithme 6 est $O(l^3 \cdot N)$, où l est la longueur des séquences et N est le nombre de séquences.

Preuve. La complexité temporelle de cet algorithme est exactement la même que celle de l'Algorithme 1, à l'exception que les k comparaisons effectuées par la distance de Hamming sont remplacées par k fois le calcul du coût moyen de mutations pré-spéciation, qui est de complexité temporelle $O(N)$, comme vu au chapitre précédent. On a donc un algorithme dont la complexité temporelle est $O(l^3 \cdot N)$.

□

4.2 Algorithme avec fenêtre coulissante

Nous montrons dans cette section une optimisation qui réutilise, pour une longueur k, les coûts moyens de mutations pré-spéciation calculés par l'itération précédente de la position de départ des segments p.

Pour une longueur k donnée et une position $p > 0$, la paire de segments adjacents correspondante à ces valeurs chevauche celle de l'itération précédente sur toute sa longueur sauf une position, soit $2k - 1$ positions. L'exemple suivant montre ce chevauchement pour une séquence et une longueur $k = 4$:

p	$k = 4$
0	AACACATATGTGATAACCCCGTATATATGTA
1	AACACATATGTGATAACCCCGTATATATGTA
2	AACACATATGTGATAACCCCGTATATATGTA
3	AACACATATGTGATAACCCCGTATATATGTA
	...
$l - 2k - 1$	AACACATATGTGATAACCCCGTATATATGTA
$l - 2k$	AACACATATGTGATAACCCCGTATATATGTA

Le coût d'une seule liste de nucléotides qui n'appartient pas aux segments courants a été calculé par l'itération précédente, soit la première liste de cette itération. Cette liste correspond aux nucléotides aux positions $p-1$ et $p-1+k$, où p est la valeur courante, où les nucléotides aux positions p_{prec} et $p_{prec}+k$ où p_{prec} est la position de départ de l'itération précédente. Le coût de la dernière liste de nucléotides appartenant aux segments courants n'a pas été calculé par l'itération précédente. Cette liste correspond aux nucléotides aux positions $p+k-1$ et $p+2k-1$.

Calculer le coût moyen de mutations pré-spéciation d_n des segments débutant à la position $p=0$ est comme dans l'Algorithme 6. Calculer ce coût d_n pour une position $p>0$ se fait en prenant la somme des moyennes de mutations pré-spéciation calculées pour chacune des listes de nucléotides de l'itération précédente, en y soustrayant le coût de la première liste de nucléotides de l'itération précédente et en y ajoutant le coût de la dernière liste de nucléotides de l'itération courante. On normalise ensuite cette nouvelle somme sur k. L'Algorithme 7 détaille l'optimisation de la fenêtre coulissante.

Algorithme 7 *algoFenetreCoulissante*$(S, phylo, minK, maxK, R, m)$

$l \leftarrow$ longueur des séquences S
$N \leftarrow$ nombre de séquences S
$resultats \leftarrow Resultats(R)$
$arbrePartage \leftarrow$ l'arbre de partage d'histoire duplicative correspondant à *phylo*
pour $k \leftarrow minK$ à $maxK$ avec un pas de 3 **faire**
    $liste \leftarrow$ la liste des nucléotides aux positions 0 et k
    annoter les feuilles de $arbrePartage$ avec $liste$
    $d_{premier} \leftarrow$ coût moyen de mutations pré-spéciation de $arbrePartage$ avec m
    $d \leftarrow d_{premier}$
    pour $p_n \leftarrow 1$ à $k-1$ **faire**
        $liste \leftarrow$ la liste des nucléotides aux positions p_n et $p_n + k$
        annoter les feuilles de $arbrePartage$ avec $liste$
        $d_p \leftarrow$ coût moyen de mutations pré-spéciation de $arbrePartage$ avec m
        $d \leftarrow d + d_p$
    fin pour
    $d_n \leftarrow d/k$
    $resultat \leftarrow (p, k, d_n)$
    ajouter $resultat$ dans $resultats$

Algorithme 7 *algoFenetreCoulissante*$(S, phylo, minK, maxK, R, m)$ (suite)

pour $p \leftarrow 1$ à $l - (2k)$ **faire**
  $d \leftarrow d - d_{premier}$ ▷ Ajustement de la somme.
  liste ← la liste des nucléotides aux positions p et $p + k$
  annoter les feuilles de *arbrePartage* avec *liste*
  $d_{premier} \leftarrow$ coût moyen de mutations pré-spéciation de *arbrePartage* avec m
  liste ← la liste des nucléotides aux positions $p + k - 1$ et $p + 2k - 1$
  annoter les feuilles de *arbrePartage* avec *liste*
  $d_{dernier} \leftarrow$ coût moyen de mutations pré-spéciation de *arbrePartage* avec m
  $d \leftarrow d + d_{dernier}$
  $d_n \leftarrow d/k$
  resultat $\leftarrow (p, k, d_n)$
  ajouter *resultat* dans *resultats*
fin pour
fin pour
retourner *resultats*

Proposition 8. La complexité temporelle de l'Algorithme 7 est en $O(l^2 \cdot N)$, où l est la longueur des séquences et N est le nombre de séquences.

Preuve. Pour chaque valeur de k générée, on calcule la somme d pour la position $p = 0$ de la même façon que dans l'Algorithme 6, soit en faisant k calculs du coût moyen de mutations pré-spéciation. Ce calcul se fait en CN opérations, où C est une constante. Pour chacune des positions $p = 1$ jusqu'à $l - 2k$, on calcule le coût moyen de mutations pré-spéciation de la première et de la dernière liste de nucléotides. On fait donc $2CN$ opérations pour ces valeurs de positions $p > 0$. L'expression suivante représente la somme du travail effectué par l'Algorithme 7 si l est un multiple de 6 :

$$3 \cdot \sum_{k=1}^{l/6} kCN + 3 \cdot \sum_{k=1}^{l/6} \sum_{p=1}^{l-2k} 2CN$$

On a :

$$
\begin{aligned}
&3 \cdot \sum_{k=1}^{l/6} kCN + 3 \cdot \sum_{k=1}^{l/6} \sum_{p=1}^{l-2k} 2CN \\
=&3CN \cdot \left(l^2/72 + l/12\right) + 6CN \cdot \sum_{k=1}^{l/6} l - 2k \\
=&3CN \cdot \left(l^2/72 + l/12\right) + 6CN \cdot \left(l^2/6 - l^2/36 - l/6\right)
\end{aligned}
$$

La complexité temporelle de l'Algorithme 7 est donc $\mathrm{O}\left(l^2 \cdot N\right)$.

□

4.3 Algorithme avec antémémoire pour classes de nucléotides

Nous montrons dans cette section une optimisation qui réutilise les coûts moyens de mutations pré-spéciation pour des listes de nucléotides appartenant à une même classe. On a vu au Chapitre 3 que les auteurs Belcaid et al. (2011) regroupent les listes de nucléotides par classes pour accélérer les calculs. On présente ici une optimisation similaire. Il est à noter que cette optimisation n'est valide que si la fonction de coût des mutations est :

$$
m(e_1, e_2) = \begin{cases} 0 & \text{si } e_1 = e_2 \\ 1 & \text{sinon} \end{cases}
$$

Les listes appartenant à la même classe ont le même coût moyen de mutations pré-spéciation. Chaque classe est identifiée par un *représentant*. On construit un représentant à partir d'une liste de nucléotides de la façon suivante :

1. On remplace chacune des instances du premier nucléotide par des 1.
2. On remplace chacune des instances du deuxième nucléotide, soit le premier nucléotide qui est différent de celui traité à l'étape précédente, par des 2.

3. On remplace chacune des instances du troisième nucléotide par des 3.
4. On remplace chacune des instances du quatrième nucléotide par des 4.

Par exemple, la liste `AAACGGTTTTCCGAT` donne le représentant `111233444422314`. On a remplacé les `A` par des 1, les `C` par des 2, les `G` par des 3 et les `T` par des 4. Les séquences suivantes produisent aussi le représentant `111233444422314` et appartiennent donc à la même classe :

```
1  CCCATTGGGGAATCG
2  TTTCGGAAAACCGTA
```

Dans le premier exemple, on remplace les `C` par des 1, les `A` par des 2, les `T` par des 3 et les `G` par des 4. Dans le deuxième exemple, on remplace les `T` par des 1, les `C` par des 2, les `G` par des 3 et les `A` par des 4.

On suppose que c'est plus rapide de calculer le représentant d'une liste de nucléotides et d'accéder au coût moyen pré-calculé correspondant que de calculer ce coût. Néanmoins, l'accélération véritable est dépendante de l'implémentation, puisque la complexité temporelle de l'algorithme ne change pas par rapport à l'algorithme standard, soit l'Algorithme 6. Nous allons voir dans la Section 4.5 qu'on obtient une accélération appréciable.

Puisque le nombre de classes possibles grandit rapidement en fonction du nombre de nucléotides d'une liste, on utilise une approche qui construit l'antémémoire sur demande, au lieu de la construire de façon hâtive. De plus, on a tendance à retrouver souvent les mêmes représentants sur des séquences d'ADN apparentées, ce qui justifie davantage le choix de cette approche. L'Algorithme 8 détaille cette optimisation.

4.4 Combinaison des deux optimisations

Les deux optimisations vues aux sections précédentes sont complémentaires. Dans l'algorithme avec fenêtre coulissante, on peut utiliser l'antémémoire là où on doit calculer obligatoirement les coûts moyens, c'est-à-dire, k fois lorsque $p = 0$ et 2 fois pour chaque position p supérieure à 0 et ce, pour chaque longueur k.

Algorithme 8 *algoAntememoire*$(S, phylo, minK, maxK, R, m)$

```
1: l ← longueur des séquences S
2: N ← nombre de séquences S
3: resultats ← Resultats(R)
4: arbrePartage ← l'arbre de partage d'histoire duplicative correspondant à la phylo
5: coutsRepresentants                          ▷ Coûts pré-calculés pour les classes.
6: pour k ← minK à maxK avec un pas de 3 faire
7:     pour p ← 0 à l − (2k) faire
8:         d ← 0
9:         pour p_n ← p à p + k − 1 faire
10:            liste ← la liste des nucléotides aux positions p et p + k
11:            representant ← représentant correspondant à liste
12:            si representant est dans coutsRepresentants alors
13:                d_p ← coût pré-calculé pour representant
14:            sinon
15:                annoter les feuilles de arbrePartage avec liste
16:                d_p ← coût moyen de mutations pré-spéciation
17:                associer d_p à representant dans coutsRepresentants
18:            fin si
19:            d ← d + d_p
20:        fin pour
21:        d_n ← d/k
22:        resultat ← (p, k, d_n)
23:        ajouter resultat dans resultats
24:    fin pour
25: fin pour
       retourner resultats
```

On ne détaille pas ici l'algorithme des deux optimisations car il est trivial de le déterminer à partir de l'Algorithme 7 et l'Algorithme 8. La complexité temporelle de l'algorithme

résultant est la même que celle de l'algorithme avec fenêtre coulissante, soit l'Algorithme 7. L'accélération produite par l'antémémoire par rapport à l'Algorithme 7 est dépendante de l'implémentation. Nous voyons dans la prochaine section que l'accélération est surtout remarquable pour un petit nombre de séquences.

4.5 Comparaison des performances

Dans cette section, nous montrons les performances des implémentations des algorithmes présentés précédemment. Nous les avons testées avec divers ensembles de séquences aléatoires. Nous faisons varier le nombre de séquences N de 3 à 12 par sauts de 3 et la longueur des séquences l de 300 à 1200 par sauts de 300 nucléotides.

Au total, chaque implémentation a été testée avec 16 ensembles de séquences différents. Nous présentons pour chacune d'entre elles des graphiques permettant de voir les temps d'exécution en fonction de N et l.

Les graphiques de la Figures 4.1 montrent les temps d'exécution pour l'implémentation de l'algorithme standard. Le meilleur temps enregistré est 7 secondes pour les valeurs minimales de N et l, soit 3 et 300 respectivement. Le pire temps enregistré est 1500 secondes, soit 25 minutes, pour les valeurs maximales de ces paramètres.

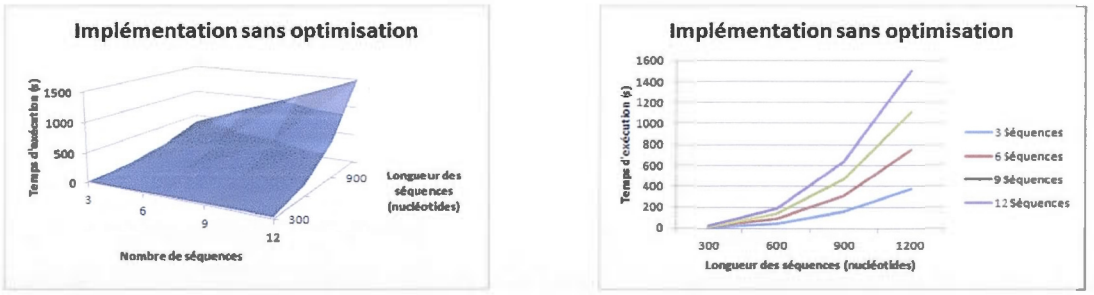


Figure 4.1: Temps d'exécution pour l'algorithme standard. Graphique en surface et courbes pour chaque nombre de séquences en fonction de leur taille.

Les graphiques de la Figure 4.2 montrent les temps d'exécution pour l'algorithme qui utilise l'optimisation de la fenêtre coulissante. Il est intéressant de noter l'impact du

changement de la complexité en fonction de la longueur des séquences l de cet algorithme sur les temps d'exécution. Le meilleur temps enregistré est $1,29$ secondes et le pire temps est $12,67$ secondes. Les temps d'exécution de cette implémentation sont de plusieurs ordres de grandeur inférieurs au temps de l'algorithme standard dans les pires cas testés.

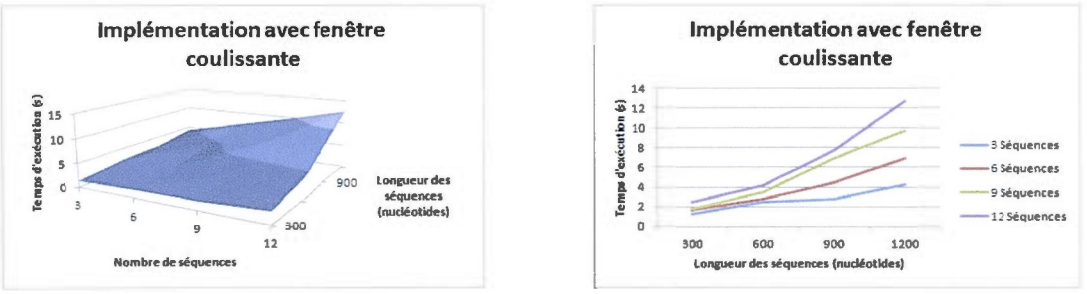


Figure 4.2: Temps d'exécution pour l'algorithme avec fenêtre coulissante. Graphique en surface et courbes pour chaque nombre de séquences en fonction de leur taille.

Les graphiques de la Figure 4.3 montrent les temps d'exécution pour l'algorithme qui utilise l'antémémoire. On remarque que la complexité de l'algorithme ne change pas, mais que les calculs sont toutefois accélérés de façon non négligeable. Le meilleur temps enregistré est $2,29$ secondes et le pire temps enregistré est $263,20$ secondes, soit un peu plus de 4 minutes.

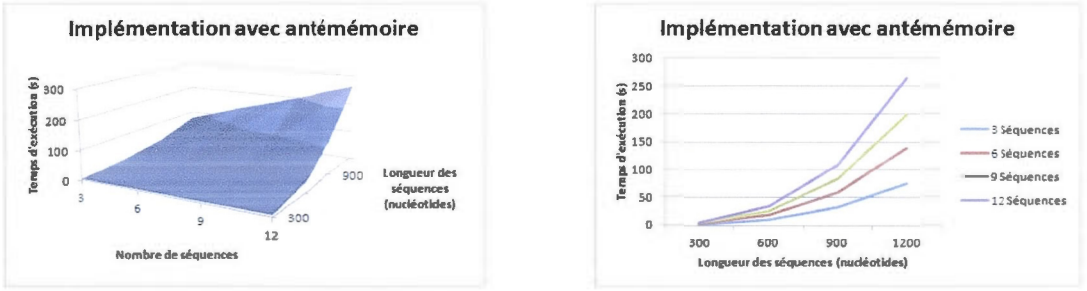


Figure 4.3: Temps d'exécution pour l'algorithme avec antémémoire. Graphique en surface et courbes pour chaque nombre de séquences en fonction de leur taille.

Les graphiques de la Figure 4.4 montrent les temps d'exécution pour l'implémentation qui combine les optimisations. On remarque que les temps d'exécution pour $N = 3$ sont

presque constants. Les temps d'exécution pour 6, 9 et 12 séquences sont comparables à l'algorithme sans l'antémémoire. Le meilleur temps enregistré est 1,147 secondes et le pire temps enregistré est 14,60 secondes, soit un peu plus lent que ce que fait l'algorithme à fenêtre coulissante sans l'antémémoire.

Nous avons refait l'expérience pour 4 et 5 séquences pour déterminer à partir de quel nombre de séquences l'optimisation avec antémémoire cesse d'être utile avec notre implémentation. Les temps d'exécution pour 3 et 4 séquences sont presque identiques. Les temps d'exécution pour 5 séquences commencent légèrement à être supérieur. On voit que l'optimisation avec antémémoire cesse d'être utile lorsqu'on traite 6 séquences ou plus en parallèle. Rappelons que ces séquences sont aléatoires. L'optimisation avec antémémoire performe mieux lorsqu'on retrouve souvent des listes de nucléotides appartenant à la même classe. On peut donc estimer qu'elle performe mieux sur de vraies séquences biologiques.

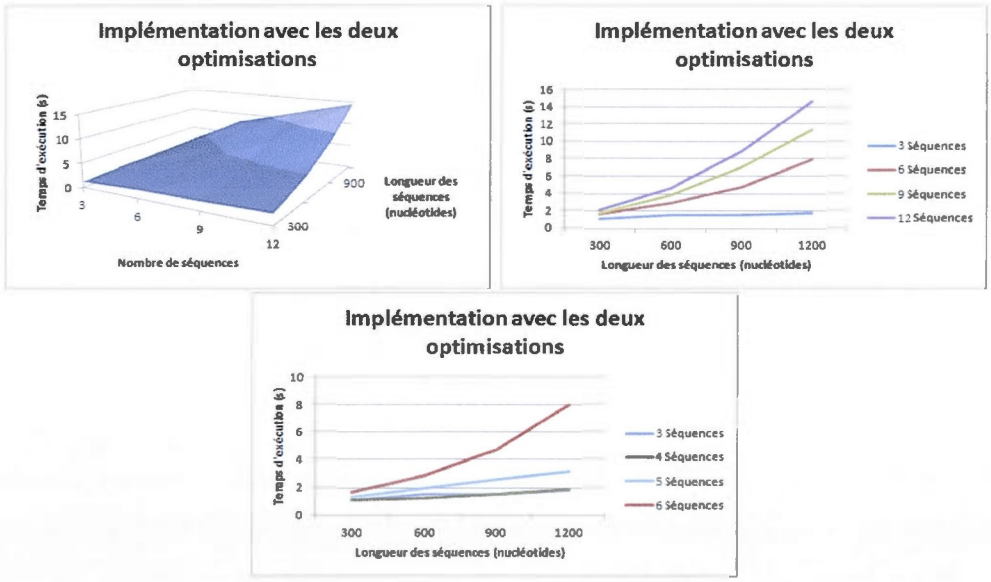


Figure 4.4: Temps d'exécution pour l'algorithme avec les deux optimisations. Graphique en surface et courbes pour chaque nombre de séquences en fonction de leur taille.

Finalement, on montre à la Figure 4.5 les accélérations des algorithmes par rapport à

l'implémentation sans optimisation. L'accélération est le ratio du temps d'exécution de l'implémentation standard sur le temps d'exécution de l'optimisation.

Plus la longueur des segments l est grande, plus l'implémentation avec la fenêtre coulissante accélère par rapport à l'implémentation standard. Elle gagne aussi en accélération au fur et à mesure que N est grand, mais de façon moins importante. On voit clairement le changement de complexité temporelle. Cette implémentation obtient une accélération de 118 dans le meilleur cas, où $N = 12$ et $l = 1200$.

La meilleure accélération est obtenue par l'implémentation qui combine les deux optimisations pour $N = 4$ et $l = 1200$, soit 275 fois plus rapide que l'implémentation standard.

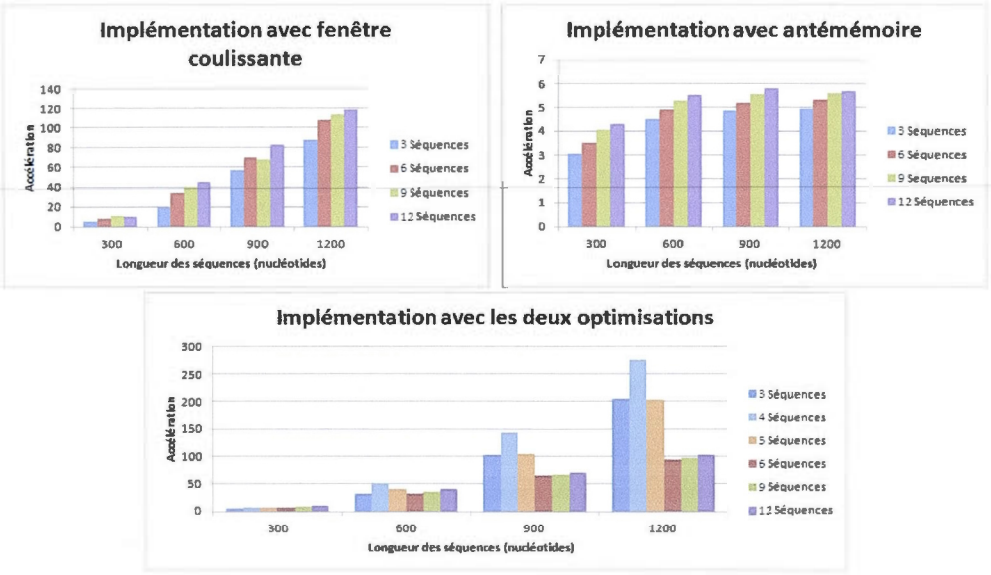


Figure 4.5: Accélérations des optimisations par rapport à l'implémentation standard.

4.6 Résultats avec des séquences codantes de TMP

On présente dans cette section une expérience où on utilise un ensemble de 5 séquences partageant leur histoire duplicative identifié dans le travail de Nicolas Massoulier (2013). Ces séquences appartiennent à des phages qui infectent la bactérie *Lactococcus lactis*.

On les présente ci-dessous :

Nom du phage	Numéro d'accession : bornes
P2	GQ979703.1 :10303-11382
CB20	FJ848885.1 :12530-13609
CB14	FJ848883.1 :12977-14056
SL4	FJ848881.1 :11003-12082
712	NC_008370.1 :11033-12112

L'Algorithme 1, qui calcule les distances de Hamming normalisées, identifie la duplication la plus récente de chaque séquence comme ayant une longueur de segments $k = 120$. Leurs positions diffèrent, par contre. Le graphique de la Figure 4.6 montre les valeurs de distances normalisées pour toutes les positions et sur chacune des séquences où $k = 120$. À des fins de comparaison, nous présentons aussi les valeurs de distances normalisées pour $k = 60$ à la Figure 4.7.

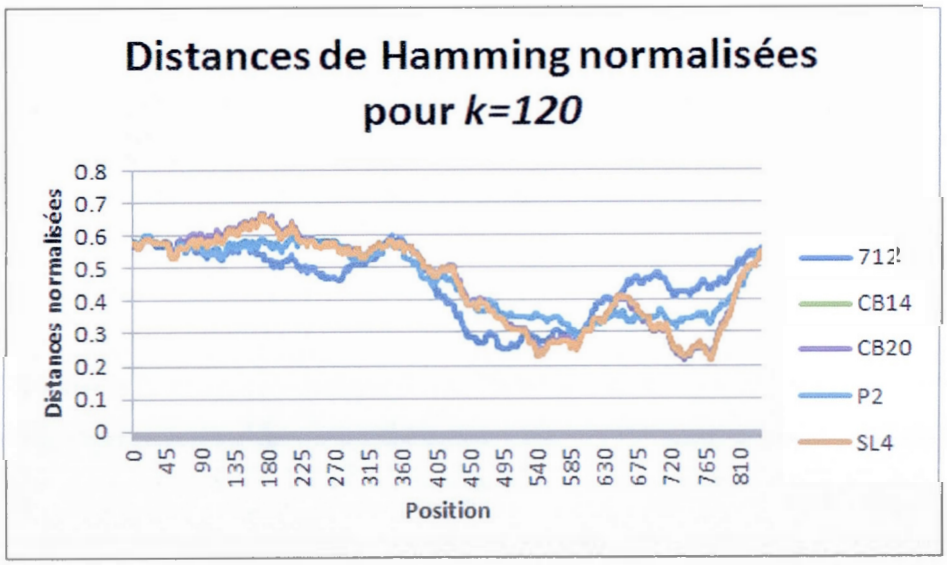


Figure 4.6: Courbes de distances de Hamming normalisées pour une longueur de segments de 120 nucléotides.

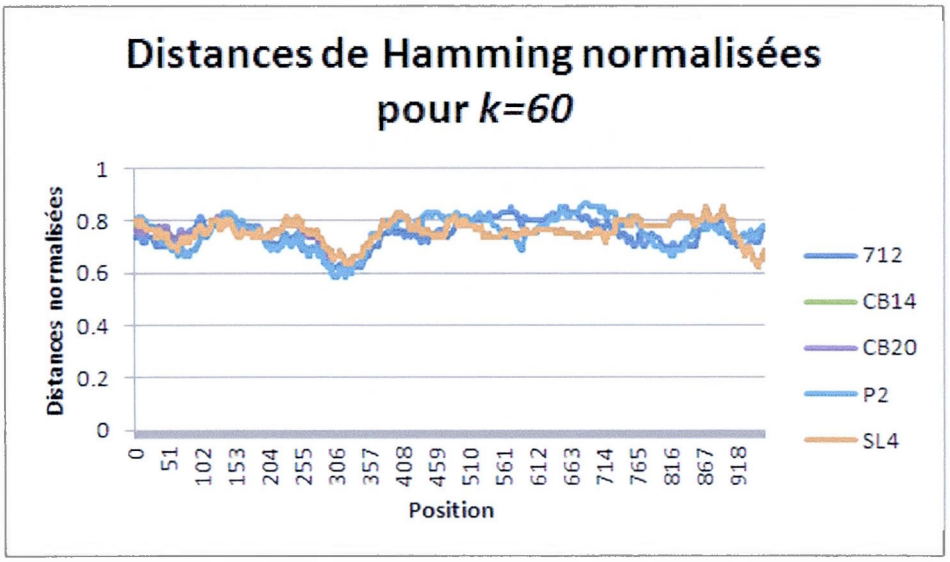


Figure 4.7: Courbes de distances de Hamming normalisées pour une longueur de segments de 60 nucléotides.

En utilisant les algorithmes de Sankoff pour calculer le coût moyen de mutations pré-spéciation des segments issus de la duplication la plus récente avec la fonction de coût présentée au Chapitre 3, on obtient aussi des valeurs de coûts minimums pour $k = 120$. Le graphique à la Figure 4.8 montre la courbe pour tous les coûts moyens où $k = 120$.

La courbe obtenue montre des minimums locaux pour trois positions.

Position	Coût moyen de mutations pré-spéciation
542	0.198
737	0.194
773	0.194

Puisque la distance qui sépare les deux dernières positions $773 - 737 = 36$ est inférieure à la longueur de 120 nucléotides d'un segment, les deux dernières positions identifient des segments dupliqués qui se chevauchent. On va donc dire qu'elles identifient la même duplication. Intégrer davantage de séquences partageant leur histoire duplicative avec celles de l'expérience permettrait peut-être d'identifier plus clairement la position cor-

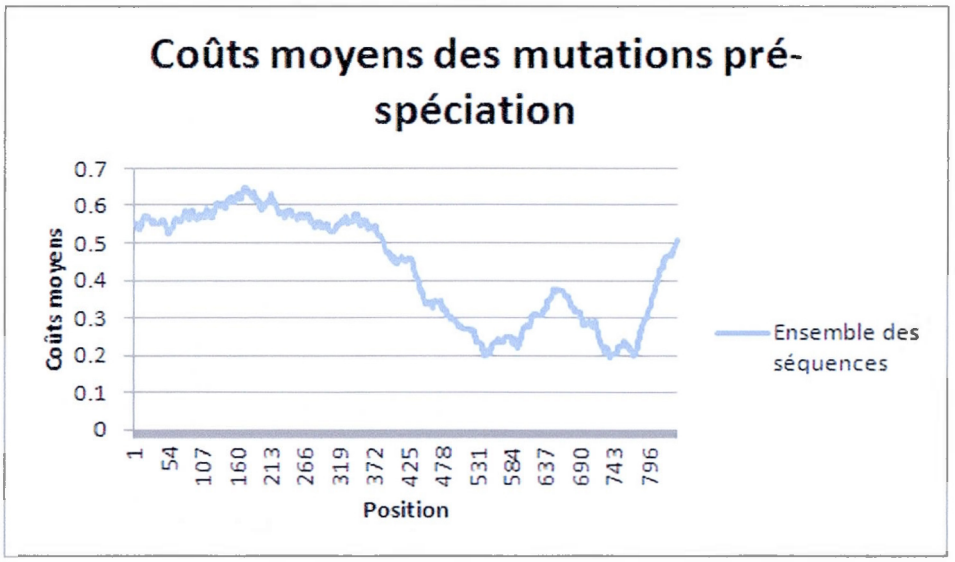


Figure 4.8: Courbes des coûts moyens de mutations pré-spéciation pour une longueur de segments de 120 nucléotides.

respondante à cette duplication.

Puisque la distance entre la première position et la deuxième position $737 - 542 = 195$ est supérieure à 120, alors les segments dupliqués ne se chevauchent pas. Par contre, si on suppose que la duplication à la première position a eu lieu avant celle de la deuxième position, alors la deuxième duplication s'est produite sur une des copies résultantes de la première duplication. À l'inverse, si la duplication qui a eu lieu sur la deuxième position est survenue avant, alors la duplication suivante s'est produite sur une position un peu avant celle de la première duplication, la repoussant ainsi vers la droite.

Ces deux duplications sont vraisemblablement arrivées à des moments très proches l'un de l'autre dans l'histoire duplicative de ces séquences.

CONCLUSION

On a développé dans ce mémoire des algorithmes calculant le coût moyen de mutations pré-spéciation basés principalement sur les travaux de Benson et Dong (1999), Sankoff (1975) et Belcaid et al. (2011). Nos algorithmes ont l'avantage de permettre le traitement de plusieurs séquences en parallèle.

Il serait intéressant de produire des segments fusionnés à partir des calculs, comme font Benson et Dong (1999). On pourrait réduire les segments issus de la duplication la plus récente de façon récursive jusqu'à ce qu'on termine sur un segment de longueur donné. Il n'est pas trivial d'élaborer une fonction de réduction car on traite plusieurs séquences en parallèle.

Il serait aussi intéressant d'utiliser une fonction de coût des mutations différente. Par exemple, on pourrait déterminer que, dans le contexte des séquences codantes de TMP pour un certain type de phage, les `A` ont plus de chance d'être substitués pour des `C` que pour n'importe quel autre nucléotide. Dans ce cas, la fonction de coût des mutations devrait refléter cette observation.

Aussi, nous avons fait abstraction des insertions et délétions dans nos algorithmes. Des travaux futurs pourraient étendre nos algorithmes pour supporter ces mutations. Une approche possible est de recevoir des alignements multiples avec écarts en entrées plutôt que des séquences. Il faudra bien sûr étendre l'ensemble des états à $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}, \texttt{-}\}$ et s'assurer que la fonction de coût des mutations produit un résultat pour chaque mutation possible.

Le domaine d'application de ces algorithmes ne se limite peut-être pas qu'aux séquences d'ADN. En effet, l'ensemble des états peut être n'importe quel alphabet et la topologie n'est pas forcément un arbre de partage d'histoire duplicative. Il serait intéressant de

voir s'il existe d'autres problèmes où nos algorithmes sont applicables.

BIBLIOGRAPHIE

Alberts, B., A. Johnson, J. Lewis, M. Raff, K. Roberts et P. Walter. 2007. *Molecular Biology of the Cell.* Garland Science, 5th édition.

Belcaid, M., A. Bergeron, A. Ouangraoua, P. Lavoie-Mongrain, N. Massoulier et G. Poisson. 2012. Duplications in tape measure proteins. Poster de la conférence RECOMB.

Belcaid, M., A. Bergeron et G. Poisson. 2011. « The evolution of the tape measure protein : units, duplications and losses ». *BMC Bioinformatics*, vol. 12 (Suppl 9), no. S10.

Benson, G. 1999. « Tandem repeats finder : a program to analyze dna sequences. ». *Nucleic acids research*, vol. 27, no. 2, p. 573–580.

Benson, G., et L. Dong. 1999. « Reconstructing the duplication history of a tandem repeat ». In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, p. 44–53. AAAI Press.

Brüssow, H., et R. W. Hendrix. 2002. « Phage genomics : small is beautiful ». *Cell*, vol. 108, no. 1, p. 13–16.

Delgrange, O., M. Dauchet et E. Rivals. 1999. « Location of repetitive regions in sequences by optimizing a compression method. ». In *Pacific Symposium on Biocomputing.* T. 4, p. 254–265. Citeseer.

Elemento, O., O. Gascuel et M.-P. Lefranc. 2002. « Reconstructing the duplication history of tandemly repeated genes ». *Molecular Biology and Evolution*, vol. 19, no. 3, p. 278–288.

Felsenstein, J. 2004. *Inferring phylogenies.* T. 2. Sinauer Associates Sunderland.

Fitch, W. M. 1970. « Distinguishing homologous from analogous proteins ». *Systematic Biology*, vol. 19, no. 2, p. 99–113.

———. 1971. « Toward defining the course of evolution : minimum change for a specific tree topology ». *Systematic Biology*, vol. 20, no. 4, p. 406–416.

———. 1977. « Phylogenies constrained by the crossover process as illustrated by human hemoglobins and a thirteen-cycle, eleven-amino-acid repeat in human apolipoprotein ai ». *Genetics*, vol. 86, no. 3, p. 623–644.

Hamming, R. W. 1950. « Error detecting and error correcting codes ». *The Bell System Technical Journal*, vol. 29, no. 2, p. 147–160.

Jaitly, D., P. Kearney, G. Lin et B. Ma. 2002. « Methods for reconstructing the history of tandem repeats and their application to the human genome ». *Journal of Computer and System Sciences*, vol. 65, no. 3, p. 494–507.

Katsura, I., et R. W. Hendrix. 1984. « Length determination in bacteriophage lambda tails ». *Cell*, vol. 39, no. 3, Part 2, p. 691 – 698.

Kolpakov, R., et G. Kucherov. 1999. « On maximal repetitions in words ». In *12th International Symposium on Fundamentals of Computation Theory-FCT'99*. T. 1684, p. 374–385. Springer.

———. 2001. *Finding approximate repetitions under Hamming distance*. Coll. « Algorithms—ESA 2001 », p. 170–181. Springer.

Massoulier, N. 2013. « Recherche d'unités de mesure dans les protéines ruban à mesurer ». Mémoire de maîtrise, UQAM.

Rivals, E. 2004. « A survey on algorithmic aspects of tandem repeats evolution ». *International Journal of Foundations of Computer Science*, vol. 15, no. 02, p. 225–257.

Rivals, E., M. Dauchet, J.-P. Delahaye et O. Delgrange. 1996. « Compression and genetic sequence analysis ». *Biochimie*, vol. 78, no. 5, p. 315–322.

Rivals, E., O. Delgrange, J.-P. Delahaye, M. Dauchet, M.-O. Delorme, A. Hénaut et E. Ollivier. 1997. « Detection of significant patterns by compression algorithms : the case of approximate tandem repeats in dna sequences ». *Computer applications in the biosciences : CABIOS*, vol. 13, no. 2, p. 131–136.

Sagot, M.-F., et E. W. Myers. 1998. « Identifying satellites and periodic repetitions in biological sequences ». *Journal of Computational Biology*, vol. 5, no. 3, p. 539–553.

Sanger, F., et A. R. Coulson. 1975. « A rapid method for determining sequences in dna by primed synthesis with dna polymerase ». *Journal of molecular biology*, vol. 94, no. 3, p. 441–448.

Sankoff, D. 1975. « Minimal mutation trees of sequences ». *SIAM Journal on Applied Mathematics*, vol. 28, no. 1, p. pp. 35–42.

Siponen, M., G. Sciara, M. Villion, S. Spinelli, J. Lichière, C. Cambillau, S. Moineau et V. Campanacci. February 1, 2009. « Crystal structure of orf12 from lactococcus lactis phage p2 identifies a tape measure protein chaperone ». *Journal of Bacteriology*, vol. 191, no. 3, p. 728–734.

Staden, R. 1979. « A strategy of dna sequencing employing computer programs ».

Nucleic acids research, vol. 6, no. 7, p. 2601–2610.

Stoye, J., et D. Gusfield. 2002. « Simple and flexible detection of contiguous repeats using a suffix tree ». *Theoretical Computer Science*, vol. 270, no. 1, p. 843–856.

Tang, M., M. Waterman et S. Yooseph. 2002. « Zinc finger gene clusters and tandem gene duplication ». *Journal of Computational Biology*, vol. 9, no. 2, p. 429–446.

Thorburn, W. M. 1915. « Occam's razor ». *Mind*, vol. 24, no. 2, p. 287–288.