Académie de Montpellier

Université de Montpellier II

— Sciences et Techniques du Languedoc —

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ MONTPELLIER II

Spécialité :	**Informatique**
Formation Doctorale :	**Informatique**
École Doctorale :	**Information, Structures et Systèmes**

Comparaison de séquences répétées en tandem et application à la génétique

par

Sèverine Bérard

Présentée et soutenue publiquement le 5 décembre 2003 devant le Jury composé de :

Marie-France Sagot, Directrice de Recherche, Inria Rhône-Alpes	Rapporteur
Alain Denise, Professeur, LRI, Université Paris-Sud	Rapporteur
Gilles Vergnaud, Ing. Chef Arm., Institut de Génétique et Microbiologie, Université Paris-Sud	Rapporteur
Michel Habib, Professeur, Université Montpellier II	Président
Olivier Gascuel, Directeur de Recherche, CNRS, LIRMM	Directeur de thèse
Éric Rivals, Chargé de Recherche, CNRS, LIRMM	Co-directeur
Maxime Crochemore, Professeur, Université de Marne-la-Vallée	Invité

Académie de Montpellier

Université de Montpellier II

— Sciences et Techniques du Languedoc —

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ MONTPELLIER II

Spécialité :	**Informatique**
Formation Doctorale :	**Informatique**
École Doctorale :	**Information, Structures et Systèmes**

Comparaison de séquences répétées en tandem et application à la génétique

par

Sèverine Bérard

Présentée et soutenue publiquement le 5 décembre 2003 devant le Jury composé de :

Marie-France Sagot, Directrice de Recherche, Inria Rhône-Alpes	Rapporteur
Alain Denise, Professeur, LRI, Université Paris-Sud	Rapporteur
Gilles Vergnaud, Ing. Chef Arm., Institut de Génétique et Microbiologie, Université Paris-Sud	Rapporteur
Michel Habib, Professeur, Université Montpellier II	Président
Olivier Gascuel, Directeur de Recherche, CNRS, LIRMM	Directeur de thèse
Éric Rivals, Chargé de Recherche, CNRS, LIRMM	Co-directeur
Maxime Crochemore, Professeur, Université de Marne-la-Vallée	Invité

Université de Montpellier II (Université des Sciences et Techniques du Languedoc)
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)

Référence
Bérard, Sèverine *« Comparaison de séquences répétées en tandem et application à la génétique ».* Thèse de doctorat, Université de Montpellier II, 5 décembre 2003.

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et courtes citations dans un but d'exemple et d'illustration, « toute représentation intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (article L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

À mes parents.

À Audrey, Clara, Claude et Sandra.

En souvenir de mes grand-mères.

Remerciements

Je voudrais tout d'abord remercier Marie-France Sagot, Alain Denise et Gilles Vergnaud d'avoir accepté d'être les rapporteurs de ce manuscrit. Je voudrais également remercier Maxime Crochemore d'avoir accepté de faire partie du jury de thèse et Michel Habib, non seulement pour sa participation au jury mais aussi pour ses cours d'algorithmique passionnants.

Pendant ma thèse j'ai eu deux directeurs : Olivier Gascuel et Éric Rivals. Je voudrais remercier Olivier pour tous les conseils judicieux qu'il m'a donnés et pour ses remarques pertinentes. Je voudrais remercier Éric pour m'avoir encadrée durant ces 3 années ainsi que lors de mon stage de DEA. Je l'ai entendu dire que s'il était thésard il n'aurait pas aimé avoir un directeur de thèse comme lui, heureusement pour moi c'est tout le contraire, je pense que je n'aurai pas pu avoir meilleur encadrant.

Pour les discussions scientifiques que j'ai pu avoir avec eux et qui m'ont fait avancer (ou reculer) dans mes recherches, je voudrais remercier Vincent Ranwez avec qui j'ai partagé mon bureau et qui a toujours trouvé mes problèmes d'arches très attrayants (heureusement il faisait déjà une thèse, sinon il m'aurait piqué la mienne), Stéphane Vialette avec qui j'ai pu discuter des modèles qu'il a proposés lors de sa thèse et qui auraient pu servir pour la mienne, merci pour sa gentillesse et sa disponibilité, François Nicolas, même s'il n'a jamais voulu montrer que mon problème est NP-complet alors qu'il fait par ailleurs de très belles réductions, enfin, merci à Fabien de Montgolfier et Jean Privat qui ont trouvé des contre-exemples à mes méthodes géniales. Du coté biologie, je voudrais remercier Stéphane Guindon avec qui j'ai partagé mon bureau, il qui n'a jamais rechigné à répondre à mes questions de biologie élémentaire, Jérôme Buard pour ses discussions sur les minisatellites, ainsi que Mark Jobling pour le jeu de données.

Les relecteurs de cette thèse, Éric bien sûr, Olivier, Audrey Richard, Laurent Bréhélin, ma maman, Jérôme Buard et surtout Sandra Bérasaluce qui l'a lue presque autant de fois que moi, merci pour toutes ces remarques.

Merci aussi à Josette Durante pour sa sympathie et son dynamisme, à Martine Péridier et Isabelle Gouat pour leur gentillesse et pour toujours trouver l'article dont on a besoin, à Marie Leroux pour son efficacité, à Séverine Heurdier pour son efficacité également et sa sympathie, enfin à Céline Berger pour tout ce qu'elle met en œuvre au LIRMM.

Merci à François Major de m'avoir permis de travailler depuis Montréal.

Lors d'une thèse il est important d'avoir une bonne ambiance au bureau et cela a été possible grâce à tous les thésards de la zone technique :

- ceux qui sont partis, David GENEST, dit Davy merci pour sa bonne humeur, sa gentillesse, ses conseils de programmation en C++ (entre autres) et sa diffusion pirate des sketches de François Pérusse, Vincent RANWEZ, le mangeur de kloug pour sa gentillesse aussi et son humour qui n'a d'égal que sa gourmandise, Gabriel PAVILLET, Le Gab™ c'est lui qui m'a montré comment couper la vanne de chauffage quand la température devenait insupportable, de plus j'ai presque hérité de sa couleur au blast et je n'en suis pas peu fière, Sandra BÉRASALUCE, nous partagions la caractéristique rare dans la zone technique d'être de sexe féminin, merci pour sa complicité et tous les bons moments qu'on a passés ensemble (même quand elle me traînait pour faire quelques tours de stade), Gilles ARDOUREL qui est à l'origine de l'expression « faire son Gilles », c'est-à-dire ne pas bouger, Stéphane GUINDON, dit Bouley un grand joueur de blast de renommé internationale puisqu'il paraîtrait qu'il a été récemment recruté par une équipe néo-zélandaise, Fabien RICO, Riiiiiico élu meilleure mascotte toutes catégories confondues, les tableaux de la zone technique parlent encore de lui 2 ans après son départ ... et enfin Christophe FAGOT, le Corse et très mystique « roi de la télécommande » ;
- ceux qui ne sont pas vraiment partis, ils ne sont plus thésards mais on les voit toujours au labo, Laurent BRÉHÉLIN, le roi du blast il a profité que Davy soit exilé dans une contrée lointaine pour lui prendre son trône, et Yannic TOGNETTI, PaBo™ le pénible de service dont les taquineries ont ensoleillé ma thèse et le goût particulier pour les assortiments de couleurs a illuminé le blast (je lui dédie d'ailleurs la couverture de cette thèse) ;
- ceux qui sont encore là, Patrice DUROUX, le cow-boy du blast merci pour tous ses cours pour mon diplôme d'apprentie-root (il en fallait du courage avec moi et mon ordinateur de fille ...) et pour tout l'aléatoire qu'il apporte dans les parties de blast, Fabien Jourdan, le PJQSLP™ merci pour tous les bonbons qu'il amène sur mon bureau mais ce n'est pas comme ça qu'il pourra me corrompre, François Nicolas, merci pour sa sympathie et sa bonne humeur (à lui tout seul son rire anime toute la zone technique, voire plus) et Denis Bertrand, le petit dernier du blast il

ne manque pas d'égaler PaBo^{TM} quant aux choix de ses couleurs, ni Gilles et Fabien pour ses attaques en traître : un grand espoir du blast.

Comme vous l'aurez compris, le blast est notre jeu favori, il permet d'évacuer les pressions parfois trop importantes ou plus véritablement de passer un bon moment et il justifie l'abondance de petits bonhommes dans la page ci-contre.

Enfin je voudrais remercier mes parents et mes amies de toujours, Audrey, Clara et Claude, pour leur soutien et leurs encouragements. Ainsi que toute ma famille et Marc, Céline et Jean-Pierre, Laurent et Laurette, Stéphane et Gaëlle, Sandrine et Dany, William, Sandrine et Cyril, Sanch et Livia, Ludo et Ludivine, Thierry et Carine, Lucie et Guillaume, Charles, Philippe, Louis et Céline, tous mes amis d'Argelliers et tous ceux que j'oublie certainement ... Et puis un plus que merci à mon chéri qui a été plus que parfait pendant la fin de ma thèse et bien plus encore.

Et pour finir un dernier merci aux « arches fantômes », initialement *portions fantômes*, pour toute la curiosité qu'elles ont suscité de la part de mes collègues de bureau à propos de mon travail de thèse.

Sèverine Bérard.

Table des matières

Remerciements **v**

Table des matières **ix**

Table des figures **xv**

Notes de lecture **xvii**

Notations **xix**

Introduction **1**

1 Notions de génétique **7**
1.1 Histoire de la génétique 8
1.2 Information génétique 9
1.2.1 Cellule 9
1.2.2 Chromosomes 10
1.2.3 ADN 11
1.2.3.1 Structure de l'ADN 11
1.2.3.2 Réplication et réparation de l'ADN 12
1.2.4 Gènes et protéines 14
1.2.5 Allèles et polymorphisme 14
1.3 Séquences répétées 16
1.4 Minisatellites 17
1.5 Évolution du patrimoine génétique 18
1.5.1 Divisions cellulaires 18
1.5.1.1 Division de la cellule eucaryote ou mitose 18
1.5.1.2 Division cellulaire sexuée ou méiose 19
1.5.2 Brassages chromosomiques 19
1.5.3 Mutations 20
1.5.3.1 Substitutions 21
1.5.3.2 Recombinaisons 21
1.5.3.3 Délétions et insertions 22
1.5.3.4 Inversions 25

1.5.4 Processus de mutation des minisatellites . . . 25
1.5.5 Un cas particulier : le chromosome Y . . . 27
1.6 Outils de détermination du patrimoine génétique . . . 28
1.6.1 Séquençage . . . 28
1.6.2 Marqueurs génétiques . . . 28
1.6.3 Carte génétique . . . 29
1.6.4 PCR . . . 29
1.6.5 Génie génétique . . . 30

2 Rappels d'informatique 33
2.1 Complexité algorithmique . . . 34
2.2 Alignements de séquences . . . 35
2.2.1 Notations . . . 36
2.2.2 Comparaison de séquences . . . 37
2.2.2.1 Pourquoi comparer des séquences ? . . . 37
2.2.2.2 Comment les séquences diffèrent-elles ? . . . 38
2.2.2.3 Analyse des différences : trace, alignement et listing . . . 38
2.2.3 Distances entre séquences . . . 41
2.2.4 Alignements . . . 43
2.2.4.1 Alignement global . . . 44
2.2.4.2 Les différents types d'alignements . . . 48
2.2.4.3 Plus longue sous-séquence commune . . . 49
2.3 Graphes . . . 51
2.3.1 Définitions et notations . . . 51
2.3.2 Graphes d'intervalles . . . 53
2.3.3 Graphes de chevauchement . . . 54
2.3.4 Graphes de cordes . . . 55

I Informatique 57

3 État de l'art 59
3.1 Problème du stable max . . . 59
3.1.1 Stable max dans les graphes de chevauchement . . . 60
3.1.2 Stable max dans les graphes de 2-intervalles . . . 67
3.1.3 Stable max dans les graphes de croisement . . . 69
3.2 Séquences répétées en tandem . . . 72
3.2.1 Reconstruction de l'histoire des répétitions en tandem . . . 72
3.2.2 Alignement de séquences contenant des répétitions en tandem . . . 74

4 Alignement avec amplification et contraction sous le modèle SSE 79
4.1 Modèle d'évolution SSE . . . 80
4.1.1 Événements évolutifs . . . 80

4.1.2 Distance induite par le modèle SSE . . . 81
4.2 Arches . . . 84
4.2.1 Définition . . . 84
4.2.2 Alignement et arches . . . 85
4.2.3 Ordre des événements . . . 87
4.3 Algorithme d'alignement . . . 87
4.3.1 Calcul de la matrice de programmation dynamique . . . 88
4.3.2 Coût de génération et compression d'arche . . . 92
4.3.3 Prétraitement . . . 95
4.3.3.1 Prétraitement à base de graphe . . . 96
4.3.3.2 Prétraitement sans décalage . . . 103
4.3.4 Complexité de l'algorithme . . . 106
4.4 Autres alignements optimaux . . . 106
4.4.1 Arches fantômes . . . 107
4.4.2 Récurrence . . . 108
4.4.3 Calcul des coûts de compression d'arche fantôme . . . 109
4.4.4 Prétraitement . . . 111
4.4.5 Complexité . . . 111

5 Le logiciel MS_ALIGN 113
5.1 Présentation . . . 113
5.1.1 MS_ALIGN2 . . . 114
5.1.2 MS_ALIGNN . . . 116
5.2 Implémentation . . . 120
5.2.1 Le programme . . . 120
5.2.2 L'interface web . . . 121

6 Extensions du modèle SSE 123
6.1 Qu'est ce qui change ? . . . 125
6.1.1 Les amplifications . . . 125
6.1.2 Les arches . . . 125
6.1.3 Le nombre d'arches . . . 127
6.1.3.1 Nombre d'arches exactes, ordre et arité quelconques . . . 127
6.1.3.2 Nombre d'arches exactes, ordre quelconque et arité 1 . . . 130
6.1.3.3 Nombre d'arches approchées, ordre quelconque et arité 1 . 130
6.2 Alignement avec amplification d'ordre 1 et d'arité $\rho - 1$ . . . 130
6.3 Nouvelles problématiques . . . 133
6.3.1 Coût des générations/compressions d'arche . . . 133
6.3.2 Arches approchées . . . 134
6.3.2.1 Limite de la modélisation en arches . . . 136
6.3.2.2 Relation avec le problème d'histoire des duplications . . . 137
6.3.3 Conclusion : extensions envisagées . . . 138
6.4 Génération optimale de séquence avec les arches EOS . . . 139

6.4.1 Arches EOS, arches exactes d'ordre supérieur 139
6.4.2 Relation de compatibilité entre arches EOS 140
6.4.2.1 Exemples 140
6.4.2.2 Définitions 141
6.4.3 La relation chronologiquement compatible 144
6.4.4 Ensemble maximal d'arches EOS compatibles 146
6.5 Rapprochement avec des problèmes déjà étudiés 149
6.5.1 Problème de stable max 150
6.5.1.1 Problème de 2-intervalles 150
6.5.1.2 Problème de trapézoïdes circulaires 151
6.5.2 Problème de graphe orienté 151
6.5.2.1 Exemple de graphe de compatibilité 152
6.5.2.2 Les pistes 153

II Génétique 155

7 État de l'art 157
7.1 Instabilité des minisatellites 157
7.2 Calcul de distances génétiques à partir de cartes de minisatellite 159
7.3 Étude du minisatellite MSY1 161

8 Application au minisatellite MSY1 169
8.1 Introduction 169
8.2 Le jeu de données et la distance utilisée 171
8.3 Résultats 172
8.3.1 Prédiction d'haplogroupes avec les cartes MSY1 172
8.3.2 Arbre des relations entre haplogroupes du chromosome Y 173
8.3.3 Relations évolutives à l'intérieur des haplogroupes 174
8.3.4 Évolution interne dans deux grandes populations 180
8.4 Conclusion 183

Conclusion 187

A Détails du jeu de données biologiques 189

B Entrée de MSY1 dans GenBank 193

C Publications 195

Références : informatique et bioinformatique 197

Références : biologie 203

Index **209**

Glossaire **213**

Table des figures

1.1 Représentation schématique d'un chromosome 10
1.2 Structure de l'ADN 11
1.3 Principe de réplication de l'ADN 13
1.4 Représentation symbolique de la transmission des allèles 15
1.5 Exemple de séquence répétée en tandem 16
1.6 Double brassage génétique lors de la méiose 19
1.7 Structure de Holliday 21
1.8 Recombinaison 22
1.9 Crossing-over inégal 23
1.10 Délétion intrabrin 24
1.11 Glissement à la réplication 24
1.12 Mécanismes de l'inversion 25
1.13 Mutation spécifique des minisatellites 26

2.1 Trois modes d'analyse de différences entre séquences 39
2.2 Différentes analyses pour la même paire de séquences 41
2.3 Dépendances dans la matrice de programmation dynamique 45
2.4 Exemple de matrice de programmation dynamique 46
2.5 Parcours arrière dans la matrice de programmation dynamique 47
2.6 Équivalence entre graphe de chevauchement et graphe de cordes 56

3.1 Exemple d'application de l'algorithme de Gavril 61
3.2 Exemple d'application de l'algorithme de Apostolico et ses collaborateurs . 65
3.3 Compatibilité entre deux 2-intervalles 68
3.4 Graphe de trapézoïdes circulaires 69
3.5 Compatibilité entre doubles intervalles 70
3.6 Exemple d'alignement terminant par une duplication 76

4.1 Exemple d'évolution d'un minisatellite chez 2 individus 85
4.2 Alignement de cartes de minisatellites 86
4.3 Équation de récurrence 88
4.4 Dépendances dans la matrice de programmation dynamique 89
4.5 Deux manières de compresser une arche 93

4.6 Arches simples, complexes, compatibles ou incompatibles 94
4.7 Transformation du problème de l'ensemble maximal d'arches compatibles en problème de stable max 96
4.8 Transformation directe des arches en intervalles 97
4.9 Liste des positions 99
4.10 Exemple de construction d'un codage α 102
4.11 Exemples d'alignements optimaux alternatifs 107
4.12 Illustration des deux cas du lemme 4 110

5.1 Écran d'accueil de MS_Align2 114
5.2 Écran de résultat de MS_Align2 115
5.3 Exemple de résultat d'alignement avec MS_Align2 116
5.4 Écran d'accueil de MS_AlignN 117
5.5 Écran de résultat de MS_AlignN 118
5.6 Exemple de matrice de distances obtenue avec MS_AlignN 119

6.1 « Nouvelles arches » et alignement 126
6.2 Illustration du dénombrement de N_2^2 129
6.3 Alignement de cartes de minisatellite 132
6.4 Exemples d'arches approchées 135
6.5 Exemples d'arches approchées (2) 135
6.6 Un exemple embêtant 136
6.7 Les arches multipied, une solution ? 137
6.8 Exemple d'arches exactes d'ordre supérieur, arches eos 140
6.9 Compatibilité entre arches selon leur sens 143
6.10 Arche eos découpée en trois facteurs 144
6.11 Relation de voisinage entre arches 145
6.12 Différence de la relation de compatibilité entre deux 2-intervalles 150
6.13 Transformation d'arches en triplets de facteurs 152
6.14 Graphe de compatibilité associé à la séquence de la figure 6.13 153

7.1 Exemple de la distance de Bríon 160
7.2 Cartes du minisatellite MSY1 163
7.3 Cartes du minisatellite MSY1, haplogroupe 8 166

8.1 Arbre phylogénétique des haplogroupes du chromosome Y 173
8.2 Arbre des haplogroupes du chromosome Y de Jobling et Tyler-Smith . . . 175
8.3 Arbre phylogénétique de l'haplogroupe 4 176
8.4 Arbre phylogénétique de l'haplogroupe 12 177
8.5 Arbre phylogénétique de l'haplogroupe 16 178
8.6 Arbre phylogénétique de l'haplogroupe 18 179
8.7 Arbre phylogénétique des allèles finnois 181
8.8 Arbre phylogénétique des allèles mongols 182

Notes de lecture

Nous donnons ici les particularités de ce manuscrit :

Bibliographie Dans ce mémoire de thèse, nous avons choisi de séparer les références bibliographiques en deux. Les références que nous pensons être du domaine de l'informatique ou de la bioinformatique se situent à la page 197 et sont citées dans le corps du document entre crochets, comme [Apostolico et al., 1992] par exemple. Les références bibliographiques que nous pensons être du domaine de la biologie seulement se situent à la page 203 et sont citées dans le corps du document entre parenthèses comme par exemple (Alonso et Armour, 1998). Lorsque nous référençons des sites web, le texte de la citation apparaît dans la police `Type Writer` et ne contient en général pas de date : [`Phylip`].

Glossaire Un glossaire se trouve à la fin de ce manuscrit (p. 213). Il contient les mots employés dans ce mémoire dont nous avons jugé utile de rappeler la définition. Lorsqu'un mot est cité dans le glossaire, il apparaît en gras dans le corps du texte (une seule fois, en général lors de sa première occurrence).

Typographie Certains mots ou expressions employés dans cette thèse sous leur forme française sont parfois plus connus sous leur nom anglais. Lorsque nous pensons que c'est le cas, nous mettons en note de bas de page leur traduction.

Les commentaires dans les algorithmes sont notés de la manière suivante : */* Commentaires */*.

Ce document a été écrit avec LaTeX et en essayant de respecter les règles typographiques de l'imprimerie nationale suivant le livre [Imprimerie Nationale, 2002]. Les guides et manuels utilisés, et ô combien nécessaires, pour LaTeX et BibTeX sont [Bayart, 1995], [Markey, 2002a] et [Markey, 2002b].

Logiciels et version électronique Les logiciels MS_Align2 et MS_AlignN dont il est question dans ce mémoire sont accessibles depuis l'adresse :

http://degas.lirmm.fr/REPSEQ/ dans la rubrique Softwares.

Une version électronique de ce document, au format ps ou pdf, est téléchargeable à l'adresse :

http://www.lirmm.fr/~berard/ dans la rubrique Recherche/Thèse.

Notations

Nous donnons ici les principaux symboles que nous utilisons dans ce document.

Chaîne de caractères

- $\Sigma = \{a, b, c, \ldots\}$: un alphabet de symboles ;
- Σ^* : ensemble des mots constructibles à partir de Σ ;
- s et r : deux chaînes de caractères sur Σ^* de longueurs respectives n et m ;
- $|s|$: longueur de la séquence s ;
- $s[i]$: $i^{\text{ème}}$ caractère de la séquence s où les caractères sont indexés de 1 à n ;
- $s[i..j] = s[i]s[i+1]\ldots s[j]$: un facteur de s ;
- $s.r$: la concaténation des chaînes s et r.

Graphe et ensemble

- $\mathcal{G}, \mathcal{H}$: graphes simples non orientés ;
- $\mathcal{D}$: graphe simple orienté ;
- V : ensemble de sommets d'un graphe ;
- E : ensemble d'arêtes d'un graphe non orienté ;
- A : ensemble d'arcs d'un graphe orienté ;
- ω : fonction de poids associée aux sommets d'un graphe ;
- $\alpha(\mathcal{G})$: nombre de stabilité du graphe $\mathcal{G}$, c'est-à-dire le cardinal d'un stable max de $\mathcal{G}$;
- $\alpha_\omega(\mathcal{G})$: poids d'un stable de poids maximal de $\mathcal{G}$ pondéré par ω ;
- x, y, z : des sommets d'un graphe ;
- $A \subset B$: désigne l'inclusion stricte de A dans B, c'est-à-dire $A \neq B$;
- $A \subseteq B$: désigne l'inclusion de A dans B où l'égalité est possible.

Programmation dynamique

- $\mathcal{A}$: matrice de programmation dynamique de dimension $(n+1) \times (m+1)$;
- $\mathcal{A}(i, j)$: une entrée de la matrice.

Modèle évolutif

- A : coût d'une amplification ;
- C : coût d'une contraction ;
- M : coût d'une mutation ;
- D : coût d'une délétion ;
- I : coût d'une insertion ;
- A_M : coût d'une amplification+mutation ;
- M_C : coût d'une mutation+contraction ;
- $G(t)$: coût de la génération de l'arche t ;
- $K(t)$: coût de la compression de l'arche t.

Intervalle

- $\mathcal{F}$: famille d'intervalles sur la droite réelle ;
- d : densité d'une famille d'intervalles ;
- I, J : intervalles de la droite réelle ;
- g_I : extrémité gauche de l'intervalle I ;
- d_I : extrémité droite de l'intervalle I.

Arche et chronologie

- $A(s)$: nombre d'arches sur la séquence s ;
- a : arité d'une arche ;
- o : ordre d'une arche ;
- arches EOS : arche Exacte d'Ordre Supérieur ;
- $\mathcal{K}$: relation chronologiquement compatible ;
- K : une chronologie.

Introduction

Quoi de plus intéressant que d'essayer de comprendre les mécanismes de la vie ? L'Homme tente depuis longtemps de percer les mystères de la machinerie biologique et de s'en servir pour améliorer son quotidien. De nombreuses découvertes indiquent que des pratiques médicales existaient déjà en Mésopotamie et en Égypte notamment, dès le III^{e} millénaire avant J-C. Depuis plusieurs années maintenant, la voie de la génétique est explorée et nous savons que beaucoup de nos caractéristiques sont déterminées par l'information génétique que nous portons dans nos cellules. Cette information, qui est appelée *génome*, est codée dans des molécules d'ADN sous la forme de *gènes*. Les gènes sont composés d'un enchaînement de quatre nucléotides, A, C, G et T. Ce sont nos gènes qui déterminent, par exemple, certains de nos caractères physiques, comme la couleur de nos yeux, mais aussi nos susceptibilités à des maladies. Nous avons tous les mêmes gènes, placés aux mêmes endroits sur l'ADN, et pourtant nous sommes tous différents. D'un individu à l'autre, le même gène peut s'écrire de manière légèrement variable. Ces variations sont appelées *polymorphismes*. Il y en a une telle quantité que le nombre de combinaisons de ces polymorphismes est quasiment infini. Ainsi, chaque être humain est unique. Certains polymorphismes sont associés à des maladies. La connaissance de l'ensemble de notre génome, et la compréhension de ses mécanismes de fonctionnement et d'évolution, peut donc nous permettre de prévenir et de soigner certaines de nos maladies.

Récemment, un groupement international d'équipes de recherche a réussi à séquencer presque entièrement les trois milliards de lettres que comporte le génome humain. D'autres génomes ont été séquencés complètement, comme celui des bactéries *E. coli* et *B. subtilis*, ainsi que les génomes de la levure, d'*A. thaliana* (arabette), du nématode (ver microscopique), de la drosophile (mouche) et depuis peu du chien. Ces avancées dans le domaine de la génétique demandent de plus en plus de moyens pour stocker et analyser la masse d'informations obtenues. L'informatique peut résoudre certains de ces problèmes. L'application des techniques et outils de l'informatique – mais aussi des mathématiques et de la physique – à la génétique se nomme la *bioinformatique*.

* *

*

Un des grands champs de recherche en bioinformatique est l'*analyse de séquences*. Les buts sont la cartographie des séquences génétiques, la recherche de gènes et la prédiction

de leur fonction. Pour cela, le bioinformaticien recherche des signaux et des répétitions dans les séquences, essaye de déterminer leur repliement bi- ou tridimensionnel, ou bien recherche des ressemblances entre ces séquences.

Les ressemblances permettent de prédire la fonction d'une séquence en connaissant la fonction d'une séquence déjà étudiée qui lui est similaire. En effet, lorsque deux séquences présentent de fortes similitudes, il y a des chances pour qu'elles aient la même fonction, et ceci même si elles ne sont pas de la même espèce. Sachant que nous partageons plus de 98 % de nos gènes avec les chimpanzés, 80 % avec la souris et même 30 % avec une laitue, le nombre de séquences similaires à exploiter est donc très important.

Dans le cadre du projet inter-EPST de l'action Bioinformatique : « Méthodes pour l'étude de l'évolution intragénomique des séquences répétées », nous nous sommes intéressés à la comparaison de séquences d'ADN particulières. Les séquences répétées sont très nombreuses dans les génomes eucaryotes. Elles couvrent, par exemple, environ 40 % du génome humain (`Genoscope`). Elles se divisent en deux classes : les séquences répétées dispersées et les séquences répétées en tandem, qui nous intéressent plus particulièrement. Les séquences répétées en tandem sont composées de motifs adjacents similaires en taille et en composition. Le polymorphisme des séquences répétées en tandem est grand, tant au niveau de la séquence, que du nombre de copies, et ce entre espèces proches, voire entre individus de la même espèce. Ces séquences sont donc des marqueurs d'évolutions récentes, mais aussi des acteurs importants dans l'évolution du génome. L'objectif du projet est d'avancer dans la compréhension des mécanismes évolutifs qui dirigent et contraignent l'apparition des séquences répétées. C'est dans ce but que nous nous intéressons à la comparaison de répétitions en tandem, et plus particulièrement de *minisatellites*. Un minisatellite est une séquence répétée en tandem dont la longueur des motifs est de l'ordre de 10 à 100 nucléotides. Ces motifs sont appelés les *variants* du minisatellite.

Pour chercher des similitudes entre séquences, la méthode la plus largement employée est l'*alignement*. Aligner deux séquences est un moyen de les comparer en mettant en évidence les ressemblances qui existent entre elles. Un alignement est représenté sous la forme d'une matrice à deux lignes, chaque ligne contenant une des séquences à aligner. Par exemple, l'alignement de $SCIENTIFIQUE$ et de $SYNOPTIQUE$ peut se représenter de la manière suivante :

S	C	I	E	N	$-$	$-$	T	I	F	I	Q	U	E
S	$-$	Y	$-$	N	O	P	T	I	$-$	$-$	Q	U	E
1	2	3	4	5	6	7	8	9	10	11	12	13	14

L'alignement peut être vu comme une série de transformations permettant de passer d'une séquence à l'autre. En génétique, les transformations ou *événements mutationnels* considérés sont : l'insertion, la délétion et la substitution. Dans l'exemple précédent, pour passer de $SCIENTIFIQUE$ à $SYNOPTIQUE$, on peut déléter le C à la position 2, substituer le I à la position 3 par un Y, déléter le E à la position 4, insérer un O et un P aux positions 6 et 7, et déléter le F et le I aux positions 10 et 11.

Dans les méthodes d'alignement classiques, seuls ces trois événements – insertion, délétion et substitution – sont pris en compte. Cependant, d'autres événements mutationnels

entrent en jeu dans l'évolution des séquences génétiques, et en particulier des séquences répétées.

Les séquences répétées en tandem sont supposées évoluer grâce à deux événements supplémentaires et spécifiques : la *duplication en tandem* et la *délétion en tandem*. Ces deux événements caractéristiques, que nous définissons de façon plus formelle sous les noms d'*amplification* et de *contraction*, agissent sur un ensemble de nucléotides, et non pas sur un seul résidu comme c'est le cas pour les événements mutationnels classiques. Voici un exemple d'amplification :

$$CAG\ \underline{CGG}\ CGG \longrightarrow CAG\ CGG\ \mathbf{CGG\ CGG\ CGG}\ CGG$$

Le motif *CGG* souligné dans la séquence de gauche est amplifié, c'est-à-dire qu'il est copié et que les copies générées – en gras dans la séquence de droite – sont placées côte à côte avec l'original.

Les minisatellites sont des séquences polymorphes qui évoluent rapidement en comparaison au reste du génome. De ce fait, elles contiennent un signal évolutif récent qui permet l'étude des populations. Elles peuvent ainsi apporter des informations concernant l'origine des langues et les migrations des premiers humains par exemple. Leur polymorphisme en fait de bons marqueurs génétiques pouvant être utilisés pour l'identification ou les tests de paternité.

En outre, de nombreuses questions concernant les séquences répétées en tandem restent obscures. Leur expansion, quand elle est trop importante, est associée à des maladies comme l'épilepsie, le diabète et certains cancers. On ignore encore les fonctions exactes des minisatellites. On ne sait pas non plus précisément modéliser leur processus de mutation propre.

* *
*

Pour aligner des séquences répétées en tandem, il est nécessaire de prendre en compte les événements d'amplification et de contraction. Hors, à ce jour aucune méthode d'alignement ne permet cela. Le but de cette thèse est la prise en compte de ces deux événements spécifiques pour comparer des séquences répétées en tandem et plus exactement des *cartes de minisatellite*. Une carte de minisatellite est la succession de ses variants, où chaque variant est représenté par un symbole (deux variants identiques étant représentés par le même symbole).

La première étape de ce travail a consisté à modéliser l'évolution des minisatellites pour formaliser le problème de l'alignement. La difficulté de ce problème provient du fait que les opérations ne sont pas commutatives. L'ordre d'application de celles-ci est donc important, à la différence du problème d'alignement classique où les opérations peuvent s'effectuer dans un ordre quelconque. Par exemple l'amplification d'un variant a à partir d'un a adjacent peut s'effectuer avant que ce dernier ne soit muté en un variant b, mais pas après. Nous avons créé un modèle d'évolution simple que nous avons nommé SSE. Un

premier algorithme d'alignement a vu le jour à partir de ce modèle SSE. Cet algorithme de programmation dynamique utilise la recherche de stable max dans des graphes construits à partir de la structure de la séquence. Il est exact, en ce sens qu'il calcule l'alignement optimal entre deux cartes. Nous avons intégré notre algorithme dans un logiciel nommé MS_ALIGN (accessible *via* le web). Nous avons ensuite appliqué notre méthode sur un jeu de données biologiques dans le but de valider notre modèle. Ce jeu de données est un ensemble de cartes du minisatellite humain MSY1 provenant d'individus différents. Nous disposons d'un *arbre phylogénétique* de ces individus, c'est-à-dire d'un arbre retraçant leurs relations évolutives. Cet arbre a été construit par des biologistes d'après des données indépendantes de MSY1. Nous avons reconstruit un arbre similaire à celui des biologistes en exploitant, avec notre algorithme, les données issues du minisatellite MSY1. Cela nous a permis non seulement de valider notre approche, mais encore d'apporter de nouvelles informations concernant l'évolution récente entre les individus qui n'apparaissaient pas dans l'arbre des biologistes. Notre modèle SSE est bien adapté aux données mais il est relativement contraint. Dans un second temps, nous avons donc essayé de relâcher les contraintes du modèle, mais la combinatoire devient alors trop importante. Les recherches menées dans ce sens ne donnent pas d'algorithmes polynomiaux, mais sont de bonnes pistes pour des heuristiques.

L'algorithme d'alignement sous le modèle SSE a été publié dans les actes d'une conférence ACM nommée RECOMB (*International Conference on Computational Molecular Biology*) [Bérard et Rivals, 2002] et une description complète a paru dans le *Journal of Computational Biology* [Bérard et Rivals, 2003]. Un article écrit en collaboration avec Jérôme Buard et Éric Rivals sur les résultats de l'application au minisatellite MSY1 est en cours de correction. Une liste complète des publications en relation avec ce travail de thèse est donné à l'annexe C, page 195.

* *
*

Ce mémoire de thèse est divisé en deux parties : une **partie Informatique** et une **partie Génétique**. La partie Informatique concerne l'algorithme d'alignement, son implémentation et les extensions du modèle SSE ; la partie Génétique traite des applications de notre méthode au minisatellite MSY1.

Ces deux parties sont précédées de deux chapitres de rappels. Le **chapitre 1** contient les informations nécessaires à la compréhension des enjeux biologiques de cette thèse ; le **chapitre 2** contient des rappels sur les techniques informatiques que nous utilisons. Nous avons voulu que cette thèse soit accessible à la fois aux informaticiens et aux biologistes, aussi ces chapitres de rappels débutent à un niveau élémentaire.

Le reste du mémoire est organisé de la manière suivante, dans la partie Informatique, le **chapitre 3** donne un état de l'art sur le problème du stable max dans différentes classes de graphes et sur les problèmes liés aux séquences répétées. Le **chapitre 4**, décrit le modèle SSE et l'algorithme d'alignement de cartes de minisatellite sous ce modèle. Le

logiciel MS_Align, qui implémente cet algorithme, est présenté au **chapitre 5**. Enfin, le **chapitre 6** traite des différentes extensions du modèle SSE que nous avons envisagées. Dans la partie Génétique, le **chapitre 7** donne un état de l'art sur les connaissances des processus de mutation des minisatellites et sur les études portant sur le minisatellite MSY1. Pour finir, le **chapitre 8** présente les résultats que nous avons obtenus en appliquant notre méthode sur les données de MSY1.

Chapitre 1

Notions de génétique

Sommaire

1.1 Histoire de la génétique 8
1.2 Information génétique 9
1.2.1 Cellule 9
1.2.2 Chromosomes 10
1.2.3 ADN 11
1.2.3.1 Structure de l'ADN 11
1.2.3.2 Réplication et réparation de l'ADN 12
1.2.4 Gènes et protéines 14
1.2.5 Allèles et polymorphisme 14
1.3 Séquences répétées 16
1.4 Minisatellites 17
1.5 Évolution du patrimoine génétique 18
1.5.1 Divisions cellulaires 18
1.5.1.1 Division de la cellule eucaryote ou mitose 18
1.5.1.2 Division cellulaire sexuée ou méiose 19
1.5.2 Brassages chromosomiques 19
1.5.3 Mutations 20
1.5.3.1 Substitutions 21
1.5.3.2 Recombinaisons 21
1.5.3.3 Délétions et insertions 22
1.5.3.4 Inversions 25
1.5.4 Processus de mutation des minisatellites 25
1.5.5 Un cas particulier : le chromosome Y 27
1.6 Outils de détermination du patrimoine génétique 28
1.6.1 Séquençage 28
1.6.2 Marqueurs génétiques 28

1.6.3 Carte génétique 29
1.6.4 PCR 29
1.6.5 Génie génétique 30

La génétique, étude de l'hérédité, est une discipline récente de la biologie. En effet, si la notion du passage de façon héréditaire d'informations biologiques entre générations remonte au XVII^{e} siècle, la mise en évidence de l'ADN comme support de l'hérédité ne date que de 1944, et sa structure ne fut élucidée qu'en 1953. Les avancées du génie génétique vont permettre un développement considérable des connaissances sur les gènes, leur fonctionnement et leur régulation. Les techniques atteignent très vite un niveau élevé de performance permettant aujourd'hui l'étude du génome humain.

Dans ce travail de thèse, nous nous intéressons à des séquences d'ADN particulières, les séquences répétées en tandem, et plus spécialement aux minisatellites. Le but de ce chapitre est de comprendre ce que sont les séquences répétées en tandem et les minisatellites, de connaître leur processus d'évolution et les outils génétiques permettant leur étude, ainsi que d'appréhender les enjeux biologiques liés à leur connaissance.

Nous commençons par décrire succinctement l'histoire de la génétique. Ensuite, à la section 1.2, nous donnons les bases de génétique utiles à la compréhension du reste du chapitre. Dans la section 1.3, nous détaillons les séquences génétiques auxquelles nous nous intéressons : les séquences répétées et plus précisément les minisatellites à la section 1.4. La section 1.5 est consacrée aux phénomènes impliqués dans l'évolution des génomes, une section spécifique traite des processus de mutation propres aux minisatellites. Enfin, la section 1.6 présente les techniques génétiques actuelles d'exploration des génomes.

1.1 Histoire de la génétique

Cette section a été réalisée en assemblant des informations provenant de (Bernot et Alibert), (Laurewce) et (Gayon).

La génétique moderne remonte aux travaux de Mendel dans les années 1860, qui le premier établit les lois de l'hérédité (Mendel, 1866, Mendel, 1961). Mendel comprend qu'un caractère héréditaire peut exister sous différentes versions – allèles –, les unes dominantes, les autres récessives (voir section 1.2.5). Il énonce les lois de la transmission de certains traits héréditaires. L'élément cellulaire responsable est le gène (section 1.2.4). Les gènes se transmettent entre les générations et fonctionnent de manière indépendante.

En 1900, trois botanistes, De Vries, Correns et Tschermak, « redécouvrent » les lois de l'hybridation de Mendel. En moins d'une décennie, une science nouvelle est fondée sur cette base, Bateson la nomme « **génétique** ». Dans le milieu des années 1910, ce sont les travaux de Morgan, sur la drosophile, qui conduisent au développement de la théorie chromosomique de l'hérédité. Les gènes sont alors localisés sur les chromosomes (section 1.2.2), et les travaux de Sturtevant, permettent d'ordonner les gènes le long des chromosomes, constituant les premières cartes génétiques (section 1.6.3).

Cependant, jusqu'au milieu du XX^{e} siècle, les gènes, bien que localisés sur les chromosomes, demeurent des entités théoriques. Leurs caractères physiques, leur nature matérielle, tout comme leur mode d'action demeurent mystérieux. Le support matériel de l'hérédité, l'acide désoxyribonucléique, ou ADN (section 1.2.3), fut identifié comme tel en 1944. La structure en double hélice de l'ADN est élucidée par Watson et Crick en 1953. Les années d'or de la biologie moléculaire sont les années 1950. Elles culminent avec la découverte par Jacob et Monod au début des années 1960, (Jacob et Monod, 1961), du modèle de régulation de la production des protéines chez les bactéries et virus : ARN messager, **transcription** de l'ADN en ARN messager, **traduction** de l'ARN messager en séquences polypeptidiques, c'est-à-dire les protéines. La découverte du code génétique (section 1.2.4) par Nirenberg et Matthaei est contemporaine de ces travaux sur la régulation (Nirenberg et al., 1963).

Dans les années 1970, la génétique et la biologie moléculaire entrent dans une nouvelle phase. Le génie génétique (section 1.6.5) est fondé sur des découvertes et des procédés techniques permettant de découper et d'insérer à volonté des séquences d'ADN. Il a rendu possible une nouvelle génétique moléculaire. Une accélération considérable dans ce processus a été observée dans les années 80-90. Les méthodes de maîtrise de l'expression des gènes, de cartographie (section 1.6.3) et de séquençage (section 1.6.1) des génomes ont été développées et c'est surtout à cette période qu'a émané une volonté internationale d'élucider complètement la séquence de génomes particuliers.

Le décryptage du génome humain est une prouesse technique à la mesure des avancées théoriques de la génétique. D'ores et déjà plusieurs centaines de génomes ont été ou sont en cours de décryptage chez les micro-organismes, les plantes et les animaux.

1.2 Information génétique

Nous présentons ici les différents éléments jouant un rôle dans le support de l'information génétique : les cellules, les chromosomes, l'ADN, les gènes et les protéines. Nous concluons cette section en définissant les allèles – différentes formes sous lesquelles peuvent apparaître des régions d'ADN – et en évoquant le polymorphisme génétique.

Cette section se base sur des informations provenant de (Boden et al., 1989), (Graur et Li, 2000), (Maftah et Julien, 1999), (`Gourde`), (`Biotech-canada`), et (`Diop, 2002`).

1.2.1 Cellule

Tous les êtres vivants, animaux et végétaux, sont composés de **cellules**. Par exemple, on estime à quelques milliers de milliards le nombre de cellules composant le corps humain. On distingue deux types de cellules : les procaryotes, dépourvues de noyau, et les eucaryotes, comprenant un noyau. Tous les organismes pluricellulaires sont constitués de cellules eucaryotes. Les humains sont donc des êtres eucaryotes.

La cellule est la plus petite portion de matière vivante qui puisse vivre à l'état isolé et notamment se reproduire. La cellule eucaryote présente toujours la même structure : une membrane, un cytoplasme et un noyau. La membrane individualise et isole du milieu extérieur. Le **cytoplasme** contient les organites, ces organites sont les instruments des différentes fonctions cellulaires. Le noyau de la cellule renferme les chromosomes qui sont le support de l'information génétique. L'ensemble des informations génétiques d'un organisme est appelé **génome**.

1.2.2 Chromosomes

Les **chromosomes** sont formés d'une molécule d'ADN, Acide DésoxyriboNucléique. Ils contiennent le matériel héréditaire des êtres vivants.

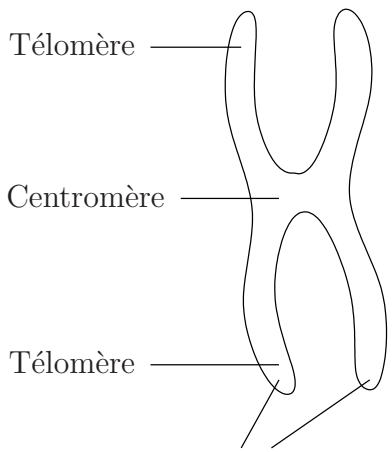


Fig. 1.1 – *Représentation schématique d'un chromosome « mitotique », c'est-à-dire sous la forme qu'il prend lors de la mitose.*

Dans une cellule au repos, c'est-à-dire entre deux divisions cellulaires, les chromosomes ne sont pas visibles individuellement. Ils font partie de la chromatine, présente dans le noyau sous forme de fines granules. C'est au cours de la division cellulaire que la chromatine acquiert une structure filamenteuse. Les chromosomes apparaissent alors sous leur forme caractéristique de deux bâtonnets parallèles, les **chromatides** ; celles-ci sont étroitement accolées au niveau du **centromère**, zone rétrécie qui sépare chaque chromatide en deux bras. Les **télomères** constituent les extrémités des chromosomes eucaryotes. La figure 1.1 montre un chromosome à l'état mitotique. Les deux chromatides d'un même chromosome sont strictement identiques et sont appelées chromatides sœurs.

Chaque espèce a un nombre de chromosomes déterminé. Les chromosomes existent en un seul exemplaire dans les cellules sexuelles (**cellules germinales** ou **gamètes** [1]), cellules dites **haploïdes**, alors qu'ils sont présents en paires dans toutes les autres cellules (les **cellules somatiques** ou cellules non sexuelles), dites **diploïdes**. Les deux éléments d'une paire de chromosomes sont dits **homologues**.

Chez les humains, il y a 23 chromosomes dans les cellules germinales et 46, ou 23 paires, dans toutes les autres. Un **autosome** est un chromosome qui ne joue pas de rôle dans la détermination du sexe, par opposition à chromosome sexuel. Il y a 22 paires d'autosomes et 1 paire de chromosomes sexuels. C'est la 23^{e} paire de chromosomes qui détermine le sexe d'un individu. Les femmes ont deux chromosomes X, et les hommes un chromosome X et un Y. Dans chaque paire de chromosomes, un exemplaire provient du père, *via* le spermatozoïde, et l'autre de la mère, *via* l'ovule.

[1] Ovules ou spermatozoïdes chez l'être humain.

1.2.3 ADN

La molécule d'**ADN** est le support du code de l'information génétique. Nous détaillons dans un premier temps la structure de cette molécule, nous décrivons ensuite ses mécanismes de réplication et de réparation.

1.2.3.1 Structure de l'ADN

La molécule d'ADN est formée par deux chaînes parallèles de nucléotides. Le génome humain contient au total quelques trois milliards de nucléotides. Une représentation planaire de la structure de l'ADN est donnée à la figure 1.2.

Chaque **nucléotide** est constitué par :
- une molécule d'acide phosphorique ;
- un sucre, le désoxyribose ;
- une base organique azotée (adénine (A), cytosine (C), guanine (G) ou thymine (T)).

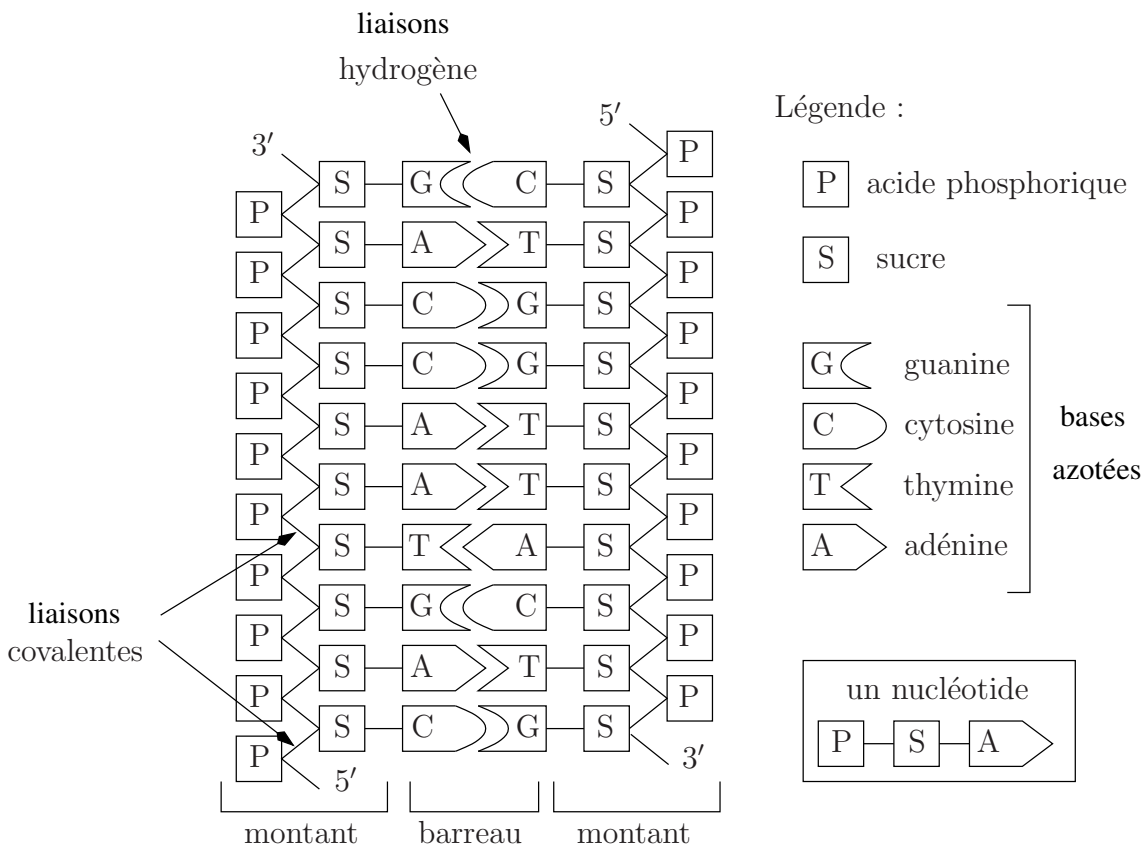


FIG. 1.2 – *Structure de l'ADN (Boden et al., 1989).*

La nature de la base forme l'unique différence entre les nucléotides. Il existe donc 4 nucléotides différents. Par abus de langage, un nucléotide est souvent confondu avec le nom de sa base azotée. Dans la chaîne de nucléotides, les liaisons chimiques solides, appelées liaisons covalentes, réunissent l'acide phosphorique d'un nucléotide et le désoxyribose du nucléotide suivant. Les deux chaînes de la molécule d'ADN sont reliées entre elles par des liaisons transversales plus fragiles, appelées liaisons hydrogène, établies entre deux bases de nucléotides face à face. Ces liaisons ne s'effectuent pas de manière quelconque car les bases sont complémentaires deux à deux :

– l'adénine ne peut se lier qu'avec la thymine ;
– la guanine ne peut se lier qu'avec la cytosine.

On dit que chaque chaîne ou brin est *complémentaire* de l'autre car chaque nucléotide d'une chaîne se lie à son partenaire complémentaire de l'autre côté. Ainsi, « l'échelle » que constitue la molécule d'ADN comporte seulement quatre types de « barreaux » : A-T, T-A, C-G et G-C. La longueur d'une séquence d'ADN se mesure en *paires de bases*, abrégé pb. Les séquences pouvant être très longues, on utilise également les notations kb, pour kilobase, et Mb, pour mégabase. Le rapport entre ces unités est le suivant : 1 Mb = 1 000 kb = 1 000 000 pb.

La molécule d'ADN est orientée, l'attachement des nucléotides ne se faisant que dans un sens : de 5′ vers 3′ (5′ et 3′ sont les numéros des atomes de carbone du sucre). Un nouveau nucléotide se fixe sur le carbone 3′ libre de la chaîne par son carbone 5′. Les deux brins complémentaires ont une orientation opposée.

Dans l'espace, en raison de contraintes de natures diverses existant entre les différentes parties de la molécule, l'échelle s'enroule en hélice. Comme les liaisons hydrogène sont plus faibles que les liaisons covalentes, les deux chaînes complémentaires formant l'échelle entortillée peuvent facilement être déroulées et séparées lors de la réplication par exemple.

1.2.3.2 Réplication et réparation de l'ADN

La **réplication**, ou duplication, est un dédoublement de l'ADN qui s'effectue lors de la division cellulaire. La réplication de l'ADN donne naissance à deux molécules filles identiques. Les deux chaînes de la molécule d'ADN s'écartent. Face à chacun des deux brins, un brin nouveau est synthétisé par incorporation des nucléotides présents dans la cellule à l'état dispersé. Par jeu de complémentarité des bases, la chaîne de nucléotide formée est identique à la moitié perdue. Chaque molécule fille est donc une réplique parfaite de la molécule mère. L'ensemble des réactions biochimiques comme l'ouverture de la molécule d'ADN, l'incorporation de nucléotides nouveaux, etc., est sous la dépendance d'enzymes spécifiques : les ADN polymérases. Une illustration du mécanisme de réplication est présentée à la figure 1.3.

L'ADN est porteur de l'information génétique de la cellule. Pour que cette information soit conservée à l'identique, il faut que sa réplication s'opère avec une fidélité totale. L'ADN est cependant sujet à de nombreuses agressions physiques et chimiques. En outre, durant la réplication, des accidents dans la polymérisation du brin naissant peuvent également se produire. Par conséquent, pour l'intégrité de l'information génétique, l'ADN ne doit pas

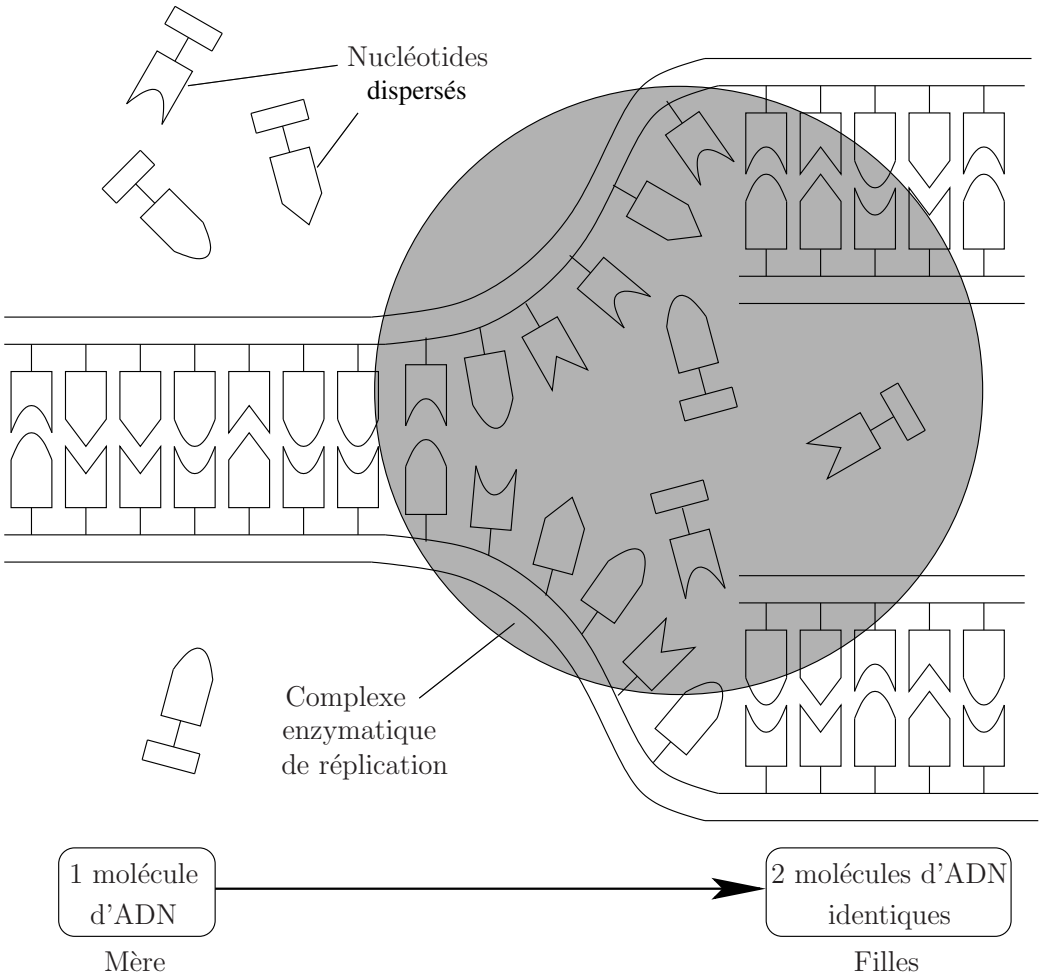


FIG. 1.3 – *Principe de réplication de l'ADN (Boden et al., 1989).*

être seulement protégé contre des lésions, mais il doit également bénéficier de systèmes de **réparation** qui garantissent à la cellule fille la transmission d'une information génétique identique à celle de la cellule mère. Une défaillance du système de réparation peut conduire à des changements permanents de la séquence nucléotidique de l'ADN, changements appelés aussi mutations (les différents types de mutations sont détaillés à la section 1.5.3).

Les principaux mécanismes de réparation de l'ADN sont basés sur l'existence de deux exemplaires de l'information génétique, sur les deux brins. Un dommage sur l'un des brins d'ADN peut être réparé grâce à l'information portée par le brin complémentaire intact. La réparation s'opère souvent par excision du brin porteur de l'erreur. L'ADN polymérase assure alors le remplacement du fragment endommagé par un fragment correct. Encore faut-il pouvoir différencier les deux brins, mais les cellules possèdent cette capacité.

1.2.4 Gènes et protéines

Chaque individu présente un **phénotype** particulier, c'est-à-dire un certain nombre de caractères directement observables dont la plupart sont héréditaires, comme la couleur des cheveux, la taille, etc. Quel que soit le type de caractère, on peut démontrer qu'il dépend étroitement de la présence ou de l'absence d'au moins une protéine donnée (ou le plus souvent d'une enzyme). En effet, dans l'organisme chaque protéine a au moins une fonction, celle-ci étant en étroite relation avec sa conformation tridimensionnelle. Les **protéines** sont constituées par l'enchaînement de molécules d'**acides aminés**. Sachant qu'il existe 20 acides aminés dans la nature, on peut entrevoir l'extrême diversité des enchaînements possibles.

Nous choisissons d'appeler **gène** la plus petite fraction d'ADN capable de diriger la synthèse d'une protéine[2]. Le passage de la séquence du gène, c'est-à-dire de l'enchaînement de 4 nucléotides différents, à la séquence de la protéine, un enchaînement de 20 acides aminés différents, fait intervenir un système de correspondance que l'on appelle code génétique.

Chaque unité du **code génétique** est une séquence de 3 bases, appelée codon (AAC, TGC, etc.). Un **codon** code pour un certain acide aminé dans une protéine, par exemple CCA code pour la glycine. Il y a 64 (4^3) codons possibles. Plusieurs codons pouvant coder pour le même acide, le code est dit *dégénéré*. Enfin, il existe 1 codon initiateur et 3 codons stop qui permettent de délimiter le début et la fin de la zone à décoder pour former la protéine. Chaque gène contient les instructions pour coder une ou plusieurs protéines.

Chez les eucaryotes, le gène est composé de deux types de séquences d'ADN, nommés exons et introns. Les **exons** se retrouvent, sous forme traduite, dans la protéine. Les **introns**, intercalés entre les exons, ne se retrouvent pas dans la protéine. Le gène a donc une structure morcelée.

1.2.5 Allèles et polymorphisme

Les **allèles** sont les deux séquences qui occupent le même site sur deux chromosomes homologues. L'un d'eux représente l'information venue du père, et l'autre, celle venue de la mère. On dit d'un sujet qu'il est **hétérozygote** si dans un couple d'allèles, celui venant du père est différent de celui venant de la mère. Le sujet est **homozygote** si, au contraire, les deux allèles sont strictement identiques.

Les allèles peuvent être des parties codantes, comme des gènes, ou bien des parties non codantes, comme les allèles d'un minisatellite par exemple. Chez l'homme, la combinaison d'allèles de certains gènes détermine la couleur des yeux, des cheveux ou encore le groupe sanguin.

Considérons par exemple le caractère couleur des yeux. On trouve à un endroit d'une chromatide le gène codant pour la couleur des yeux. En face, sur le chromosome homologue, et au même endroit, on trouvera ce même gène. L'ensemble de ces deux emplacements s'appelle un **locus**. Un allèle codant pour un caractère, comme la couleur des yeux, est **dominant** s'il arrive toujours à s'exprimer, c'est-à-dire que le caractère qu'il code est visible

[2]Il existe en effet plusieurs définitions du mot gène.

dans le phénotype de l'individu même quand cet allèle est seul à l'état hétérozygote. L'autre allèle, qui ne s'exprime qu'à l'état homozygote, est dit **récessif**.

Des exemples de transmission des allèles sont donnés à la figure 1.4. Dans cette illustration, 'M' dénote l'allèle codant pour le caractère « yeux marron » et 'b' l'allèle codant pour le caractère « yeux bleus ». M est dominant, b est récessif. Dans le premier exemple, les deux parents sont homozygotes. Le père a les yeux marron et la mère a les yeux bleus : tous les enfants seront hétérozygotes aux yeux marron. Dans le second exemple, les deux parents sont hétérozygotes et ont les yeux marron. Il y a 75 % de chances pour qu'un enfant issu de ce couple ait les yeux marron, et 25 % de chances qu'il ait les yeux bleus.

	1er exemple		2e exemple	
Parents	père $\left(\frac{M}{M}\right)$	mère $\left(\frac{b}{b}\right)$	père $\left(\frac{M}{b}\right)$	mère $\left(\frac{M}{b}\right)$
Gamètes	(M)	(b)	(M) ou (b)	(M) ou (b)
Enfants	100 % $\left(\frac{M}{b}\right)$		$\left(\frac{M}{M}\right)$ $\left(\frac{M}{b}\right)$ $\left(\frac{b}{b}\right)$ 25 % 50 % 25 %	

FIG. 1.4 – *Représentation symbolique de la transmission des allèles (Boden et al., 1989).*

Chaque individu possède une combinaison d'allèles qui lui est propre. Les loci où la molécule d'ADN peut être différente d'un individu à l'autre sont qualifiés de polymorphes. Le **polymorphisme** indique donc l'existence de plusieurs allèles à un locus donné. Le polymorphisme existe aussi bien sur les parties codantes que sur des parties non codantes. Sur le plan moléculaire, on peut classer le polymorphisme de l'ADN en trois catégories (de Vienne, 1998) :

- le polymorphisme ponctuel de séquence ou **SNP** (*Single Nucleotide Polymorphism*) ;
- le polymorphisme d'insertion-délétion ;
- le polymorphisme de nombre d'unités de répétition dans les régions répétées, plus spécialement dans les micro et minisatellites polymorphes.

Le polymorphisme est à l'origine de la diversité génétique à l'intérieur d'une même espèce.

1.3 Séquences répétées

Certaines séquences d'ADN peuvent présenter des particularités. C'est le cas des **séquences répétées**. Elles sont présentes dans différentes espèces. Chez l'homme, l'ensemble des séquences répétées couvrent environ 40 % du génome (`Genoscope`). Deux types de séquences répétées sont distinguées : les séquences répétées en tandem et les séquences répétées dispersées, suivant que les copies de l'élément répété sont adjacentes ou dispersées.

Les séquences répétées dispersées sont groupées dans 2 classes principales, les SINEs (*Short interspersed repeated sequences*) et les LINEs (*Long interspersed repeated sequences*). On les trouve partout, dans les régions géniques, dans les régions intergéniques, et aussi dans les introns.

Les séquences répétées en tandem sont constituées de motifs adjacents similaires en taille et en composition. Nous nous intéressons plus particulièrement à ces séquences. Un exemple de séquence répétée en tandem, entourée de ses séquences flanquantes, est donné à la figure 1.5. Dans cette répétition, la taille de l'unité répétée est 4, le nombre de répétition est 8 et les lettres en gras dénotent les différences entre les copies et le *motif consensus* : *ACGT*.

$$\ldots ACGCTCTGT \quad ACGT\ ACG\mathbf{G}\ ACGT\ ACG\mathbf{G}T\ ACGT\ \mathbf{C}CGT\ ACT\ ACGT \quad CATTGGT \ldots$$

séquence flanquante ⟵ répétition en tandem ⟶ séquence flanquante

Fig. 1.5 – *Exemple de séquence répétée en tandem entourée de ses séquences flanquantes.*

Il est commun de distinguer trois sous-classes parmi les séquences répétées en tandem, nous les donnons ici par ordre décroissant de la taille de l'unité répétée : les satellites, les minisatellites et les microsatellites. Le terme satellite provient du fait que la plupart des premières répétitions en tandem pouvaient être séparées du reste du génome comme des fractions *satellites* d'ADN par centrifugation. Le tableau ci-dessous présente les principales caractéristiques de chacune des classes de séquences répétées en tandem. La longueur des répétitions et la taille du motif répété, colonnes 1 et 2, diffèrent légèrement selon les auteurs.

	Longueur	Motif	Localisation	Synonymes
Satellites	100 kb-1 Mb	$>$ 100 pb	Centromères	
Minisatellites	1-30 kb	10-100 pb	Régions non-codantes Télomères	VNTR[1]
Microsatellites	$<$ 150 pb	1-10 pb	Distribués le long du génome	STR[2]

[1] *Variable Number of Tandem Repeats.*
[2] *Short Tandem Repeats.*

L'abondance des répétitions en tandem soulève des questions théoriques non encore résolues sur leur rôle dans la structure et l'évolution du génome ((Holmes et Page, 1998) et (Li, 1997)). Comment et pourquoi apparaissent-elles et évoluent-elles ?

Les séquences répétées sont impliquées dans plusieurs maladies. Par exemple, des minisatellites variables sont associés au développement des diabètes de type I, à l'épilepsie et à certains cancers (Buard et Jeffreys, 1997). Certains microsatellites sont connus pour jouer un rôle dans la régulation de gènes (Goldstein et Schlotterer, 1999). D'autres sont impliqués dans le développement d'une douzaine de maladies neurodégénératives sévères entraînées par un grand nombre d'amplifications de microsatellites CAG/CCG à l'intérieur ou à proximité d'un gène : le syndrome de l'X fragile, la dystrophie myotonique de Steinert ou la maladie de Huntington, (voir (Li, 1997) et [Benson et Dong, 1999]) [Rivals, 2004].

1.4 Minisatellites

Les **minisatellites** sont une classe particulière de séquences répétées en tandem. Elles constituent le sujet principal de cette thèse.

Ce sont des structures composées de la répétition en tandem d'un motif élémentaire de longueur allant d'une dizaine à une centaine de nucléotides : entre 7 et 100 pb pour (Vergnaud et Denoeud, 2000), entre 10 et 50 pb pour (Jobling et al., 1998). Un minisatellite a une taille de l'ordre du kilobase. Chaque unité différente du motif répété est appelée un *variant*. Pour illustrer cette notion, dans l'exemple de la figure 1.5, il y a 5 variants : $\{ACGT, ACGG, ACGGT, CCGT, ACT\}$. La taille du motif n'est pas celle d'un minisatellite, une séquence réelle de minisatellite est donnée dans l'annexe B, page 194.

Le polymorphisme d'un minisatellite est dû à la différence, d'un individu à l'autre, du nombre et de la succession des variants. Certains minisatellites ont un taux de mutation très élevé, ce qui implique un niveau de polymorphisme important. Par conséquent, ces minisatellites sont particulièrement utiles entre autres pour l'identification génétique et les tests de paternité (`Genoscope`).

Dans les génomes eucaryotes, les minisatellites sont principalement associés aux régions télomériques. On trouve par exemple 90 % des minisatellites de l'homme dans les régions subtélomériques. Pour le cochon et le rat, le biais de localisation dans les régions subtélomériques est respectivement de 66 % et 30 % (Amarger et al., 1998).

Les minisatellites ne sont habituellement pas annotés dans les séquences. Ceci est dû à un manque de précision dans la définition de ces structures, mais surtout au fait que leur détection systématique n'était pas possible avant TRF, *Tandem Repeat Finder* et TRDB *Tandem Repeat DataBase* [Benson, 1999]. TRF est un logiciel de recherche de répétitions en tandem et TRDB est une base de données pour ce type de répétitions. Malgré cela, un intérêt de plus en plus grand pour les minisatellites apparaît à cause de leur polymorphisme.

Comme pour les microsatellites, les variations de longueur des minisatellites sont impliquées dans plusieurs maladies. Des signes indiquent qu'ils contribueraient à d'autres fonctions génomiques (voir (Buard et Jeffreys, 1997) et (Vergnaud et Denoeud, 2000)).

1.5 Évolution du patrimoine génétique

Presque tous les individus d'une espèce donnée possèdent la même formule chromosomique (le nombre et la forme structurale des chromosomes), le même caryotype (photographie ordonnée des chromosomes) et les mêmes gènes, et pourtant aucun n'est identique. Cela s'explique par la diversité génétique due aux polymorphismes. Les principaux mécanismes conduisant au polymorphisme sont les modifications accidentelles de l'ADN – les mutations – et le brassage génétique assuré par la reproduction sexuée – brassage inter et intrachromosomique.

La plupart des mutations et les brassages chromosomiques se produisent au cours des divisions cellulaires. Nous décrivons donc dans un premier temps les principes de ces divisions. À la section 1.5.2, nous présentons les brassages chromosomiques. Nous donnons ensuite un aperçu des différents types de mutations qui se produisent sur les séquences d'ADN à la section 1.5.3. Puis, à la section 1.4, nous nous intéressons plus particulièrement aux mutations subies par les minisatellites. Enfin, nous concluons par un cas particulier du point de vue du brassage de l'information génétique : le chromosome Y.

1.5.1 Divisions cellulaires

Il existe deux divisions cellulaires, la mitose et la méiose. La mitose permet la formation de deux cellules filles de **génotypes** identiques à partir d'une cellule mère. Elle permet donc la reproduction cellulaire. La méiose, quant à elle, est une division cellulaire propre aux organismes qui se reproduisent sexuellement. Elle joue un rôle majeur dans les processus génétiques et héréditaires, car elle assure le brassage de l'information génétique.

1.5.1.1 Division de la cellule eucaryote ou mitose

La **mitose** aboutit à la transmission à chacune des cellules filles d'une copie complète de l'information génétique de la cellule mère. La mitose se déroule en quatre phases successives : prophase, métaphase, anaphase et télophase.

Durant l'interphase qui précède la mitose, chaque molécule d'ADN est répliquée mais les deux copies ne se séparent pas totalement ; elles restent unies en un point : le centromère.

Au début de la mitose, en prophase, on assiste à une extraordinaire condensation des molécules d'ADN. Chaque molécule forme alors un batônnet appelé chromatide. Les deux chromatides correspondant aux deux copies d'une même molécule d'ADN demeurent unies au niveau du centromère et forment le chromosome mitotique (illustré figure 1.1, page 10).

Ensuite, en métaphase, tous les chromosomes se rangent au milieu de la cellule puis, en anaphase, les deux chromatides de chaque chromosome se séparent et s'éloignent l'une de l'autre pour se retrouver aux deux pôles opposés de la cellule.

La dernière phase, la télophase, se caractérise par la formation d'un noyau au niveau de chacun des deux lots de chromatides. L'obtention de deux cellules filles nécessite une division du cytoplasme entre les deux noyaux. Chaque fille a reçu une chromatide de chaque

chromosome, c'est-à-dire une copie de chacune des molécules d'ADN de la cellule mère, c'est-à-dire encore la totalité de l'information génétique de la cellule mère.

1.5.1.2 Division cellulaire sexuée ou méiose

La **méiose** est seulement destinée à produire les cellules germinales (ovules et spermatozoïdes). Deux différences majeures distinguent la mitose et la méiose. Dans la méiose, et contrairement à la mitose :

- la répartition du matériel génétique dans les cellules filles se fait tout en réduisant de moitié le matériel génétique de la cellule mère (on passe de $2n$ à n chromosomes) ;
- une cellule en donne quatre et non deux, il y a mélange du matériel génétique provenant des parents et les cellules filles issues de la méiose ne sont pas génétiquement identiques entre elles.

La méiose est organisée en deux divisions successives. La première est dite réductionnelle (division méiotique I) et la seconde est dite équationnelle (division méiotique II). Lors de la première division, les chromosomes homologues s'apparient puis se séparent. Les deux cellules filles ne contiennent plus qu'un chromosome de chaque paire, provenant indifféremment du père ou de la mère, c'est la *ségrégation indépendante des chromosomes*. Les deux cellules sont haploïdes. Lors de la deuxième division, semblable à une mitose, les chromatides de chaque chromosome se séparent. On obtient donc 4 cellules haploïdes contenant n chromosomes.

1.5.2 Brassages chromosomiques

Méiose et fécondation assurent au cours de la reproduction sexuée un important brassage des chromosomes. La méiose assure le brassage génétique par un double mécanisme : un brassage intrachromosomique réalisé par les recombinaisons et un brassage interchromosomique lors de la ségrégation indépendante des chromosomes. Au cours de ce brassage, les allèles venant des parents sont redistribués conduisant à de nouvelles combinaisons alléliques dans les gamètes. Ceci est illustré à la figure 1.6 avec deux paires de chromosomes.

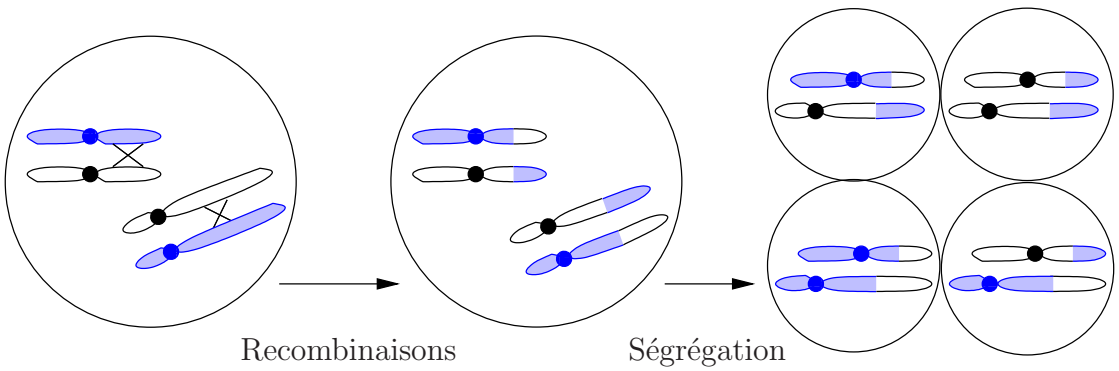


FIG. 1.6 – *Double brassage génétique lors de la méiose.*

Brassage intrachromosomique : Il s'effectue lors de la première division de la méiose. En effet, les chromosomes étant appariés et par là même très rapprochés l'un de l'autre, des échanges de segments de chromatides se produisent entre chromosomes homologues, ce sont les recombinaisons. Ainsi, tous les gènes portés par un chromosome ne sont pas forcément transmis en bloc à la descendance.

Brassage interchromosomique : La transmission des chromosomes d'une génération à l'autre fait intervenir deux fois le hasard :

- à l'issue de la méiose chaque gamète ne reçoit que l'un ou l'autre des chromosomes de chacune des paires d'homologues. Cela aboutit dans les cellules haploïdes à un nombre de combinaisons chromosomiques possibles d'autant plus grand que le nombre de chromosomes de l'espèce est lui-même élevé. Chez l'homme par exemple, il y a 2^{23} combinaisons possibles ;
- au moment de la fécondation, c'est encore le hasard qui fait que tel spermatozoïde plutôt que tel autre pénètre l'ovule.

1.5.3 Mutations

Les séquences d'ADN sont normalement copiées à l'identique lors du processus de réplication. Cependant, des erreurs dans la réplication ou dans la réparation de l'ADN peuvent survenir, produisant de nouvelles séquences. Ces erreurs sont nommées mutations. Les **mutations** sont à l'origine de l'apparition de nouveaux allèles. Elles contribuent à la diversité génétique. Elles peuvent se produire dans les deux types de cellules d'un organisme, les cellules germinales et les cellules somatiques. Notons que les mutations des cellules somatiques ne sont pas héritées.

Les mutations se partagent en deux catégories : celles que notre patrimoine génétique porte à notre naissance et celles qui se produisent au cours de notre vie. On estime que le style de vie, la diète ou l'environnement jouent un rôle dans l'apparition des mutations du patrimoine génétique. Des mutations peuvent aussi survenir à la suite d'agressions chimiques ou physiques, comme les rayons ultraviolets ou les rayons X, le benzopyrène, un composant de la cigarette, et de nombreux solvants, mais aussi à la suite d'expositions à des radiations.

Les mutations peuvent être classées selon la longueur de la séquence d'ADN qu'elles affectent. Par exemple, les mutations peuvent affecter un seul nucléotide (mutations ponctuelles) ou plusieurs nucléotides adjacents (mutations segmentaires). Les mutations peuvent également être classées selon le type de changement qu'elles causent :

- *substitution* : le remplacement d'un nucléotide par un autre ;
- *recombinaison* : l'échange d'une séquence avec une autre (la recombinaison n'est pas mutagène si elle entraîne un échange réciproque) ;
- *délétion* : la suppression d'un ou plusieurs nucléotides de l'ADN ;
- *insertion* : l'ajout d'un ou plusieurs nucléotides à la séquence ;
- *inversion* : la rotation de 180° d'un segment de double brin d'ADN comprenant au moins deux paires de bases.

Nous détaillons dans la suite, chacun de ces types de mutations.

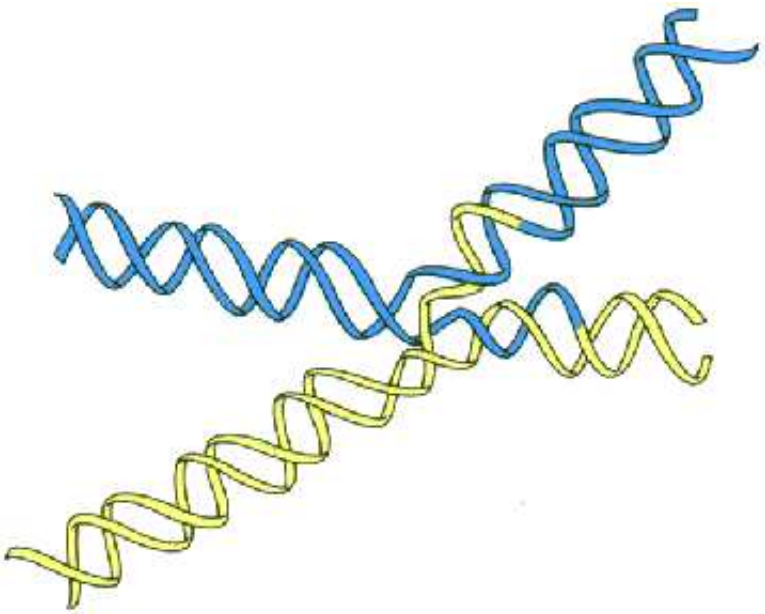


FIG. 1.7 – *Structure de Holliday. L'ADN hétéroduplex est composé de brins de chromatides mal appareillés (brins clairs et foncés).*

1.5.3.1 Substitutions

Les **substitutions** sont des mutations ponctuelles. Elles sont de deux types, transitions et transversions. Les transitions sont les changements entre A et G (purines) ou entre C et T (pyrimidines). Les transversions sont les changements entre une purine et une pyrimidine. Les substitutions qui se produisent sur des régions codant pour une protéine peuvent en affecter le résultat en modifiant le codon correspondant à un acide aminé. La structure et/ou la fonction de la protéine peuvent en être altérées, cela peut causer des maladies génétiques par exemple.

1.5.3.2 Recombinaisons

Il y a deux types de **recombinaison** homologue : le **crossing-over** qui est une recombinaison réciproque et la **conversion génique** qui est une recombinaison non réciproque. La recombinaison réciproque implique un échange de longueur égale de séquences homologues entre chromosomes homologues, les deux séquences impliquées dans la recombinaison sont conservées mais elles ont changé de place. La recombinaison non réciproque implique un remplacement inégal d'une séquence par une autre, et donc la perte d'une des séquences impliquées dans la recombinaison.

La recombinaison a lieu lors de la réparation des cassures de l'ADN, c'est-à-dire soit en mitose, soit en méiose. Les deux types de recombinaisons homologues sont supposés impliquer un intermédiaire moléculaire appelé structure de Holliday (fig. 1.7). La résolution

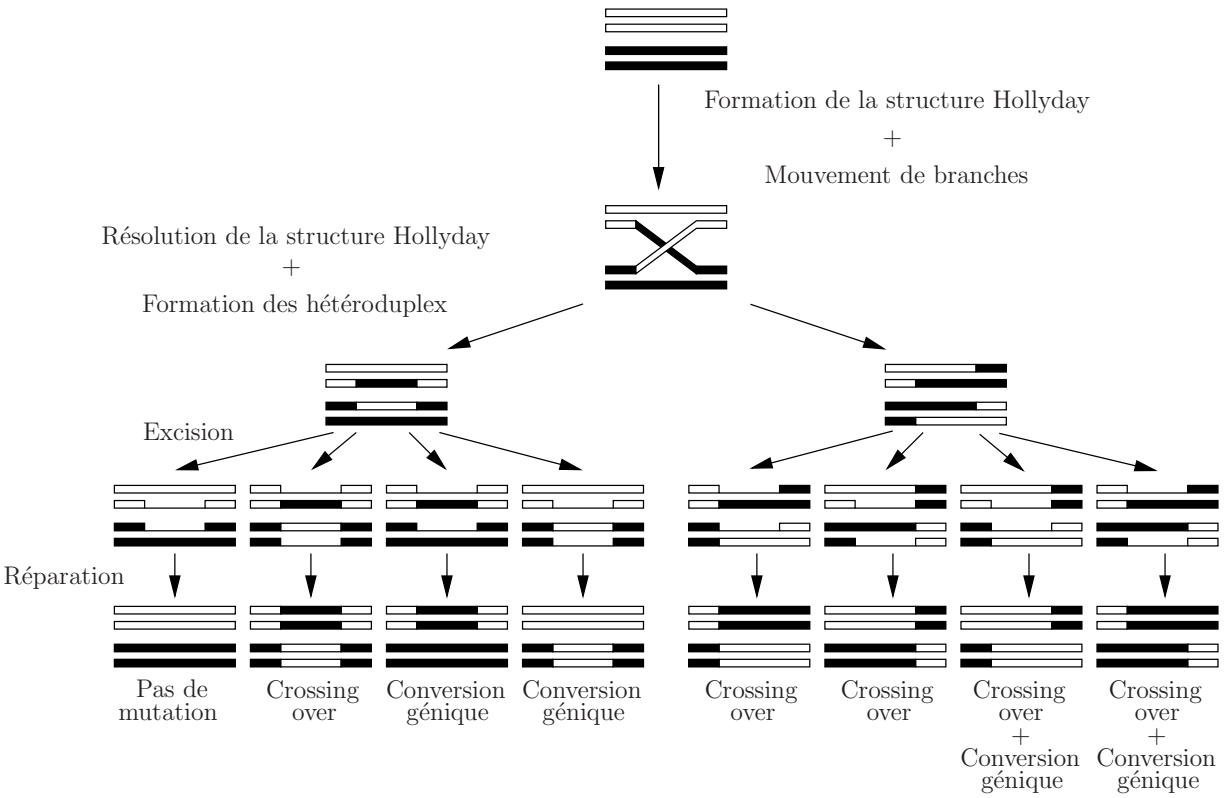


Fig. 1.8 – *Résultats possibles de la résolution de la structure de Holliday, de l'excision et de la réparation de l'ADN hétéroduplex consécutive (Graur et Li, 2000).*

de la structure de Holliday aboutit à la formation de bandes d'ADN double brin mal appariées et appelées hétéroduplex. Les mauvais appariements dans l'hétéroduplex sont reconnus et excisés par des enzymes cellulaires. Alors, en utilisant le brin complémentaire comme modèle, l'ADN polymérase remplit les trous résultant de l'excision.

La manière dont la structure de Holliday est résolue (excision) et le mode de réparation de l'hétéroduplex déterminent le résultat de la recombinaison. Les différentes possibilités sont montrées à la figure 1.8. Dans ce schéma, chaque rectangle représente un brin d'ADN. Les lignes simples représentent les endroits de l'excision. Suivant le type de résolution et le choix des brins à exciser et réparer, on obtient soit aucune mutation, soit un crossing-over, soit une conversion génique, soit crossing-over et conversion génique.

1.5.3.3 Délétions et insertions

Les **délétions** et **insertions** peuvent se produire par le biais de différents mécanismes. Un des mécanismes est le **crossing-over inégal**[3]. La figure 1.9 (a) présente un modèle simple, dans lequel le crossing-over inégal aboutit à la délétion d'un segment d'ADN dans

[3] *Unequal crossing over.*

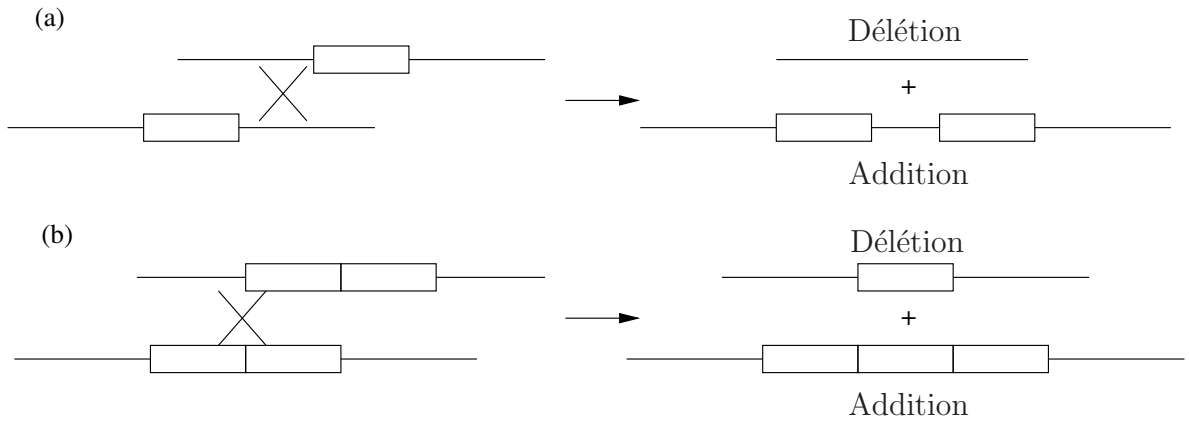


FIG. 1.9 – *(a) Crossing-over inégal aboutissant à la délétion d'une séquence d'ADN dans un des brins et à la duplication de la même séquence dans l'autre brin. (b) Quand un segment d'ADN est dupliqué en tandem, les chances de mauvais alignement augmentent, et donc également les chances d'un crossing-over inégal (Graur et Li, 2000).*

un chromosome et à l'addition réciproque dans un autre. Les chances d'un crossing-over inégal sont grandement augmentées si le segment d'ADN est dupliqué en tandem, c'est-à-dire si plusieurs copies de ce même segment sont adjacentes (cf. figure 1.9 (b)). En effet, l'ADN dupliqué en tandem induit une probabilité plus grande de mauvais alignement.

Un autre mécanisme est la délétion « intrabrin »[4]. C'est une recombinaison survenant lorsqu'une séquence répétée s'apparie avec une autre séquence répétée dans la même orientation, sur la même chromatide. Ceci est illustré à la figure 1.10, les séquences répétées sont symbolisées par des flèches et leur orientation est indiquée par le sens de la flèche. La conséquence d'une délétion intrabrin est un crossing-over intrachromosomique, symbolisé par une croix, entraînant la perte de la partie qui se situait entre les deux séquences répétées, ceci est représenté à gauche de la figure, et la génération d'un élément extrachromosomal, représenté à droite de la figure.

Un troisième mécanisme est le glissement à la réplication[5], ou mauvais appariement de brins décalés[6]. Ce type d'événement se produit dans des régions d'ADN qui contiennent des courtes répétitions en tandem. Pendant la réplication de l'ADN, un glissement de l'ADN polymérase peut se produire à cause d'un mauvais appariement entre des répétitions voisines, et ce glissement peut induire soit une délétion, soit une duplication du segment d'ADN. Un exemple de duplication d'un segment d'ADN par glissement est présenté à la figure 1.11. L'étape 1 montre la synthèse d'un nouveau brin lors de la réplication, à l'étape 2, un glissement se produit et forme une boucle, ce qui conduit au final à l'expansion du brin synthétisé, montré à l'étape 3.

Enfin, un quatrième mécanisme responsable de délétions ou d'insertions de séquences

[4] *Intrastrand deletion.*
[5] *Replication slippage.*
[6] *Slipped-strand mispairing.*

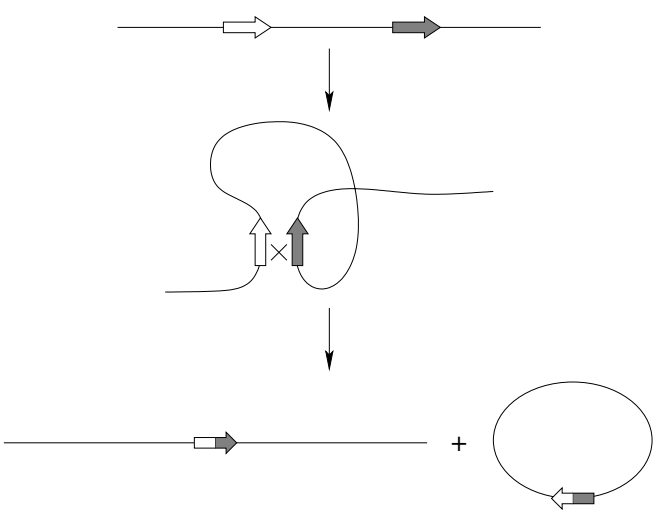


Fig. 1.10 – *Génération d'une délétion par le processus de délétion intrabrin (Graur et Li, 2000).*

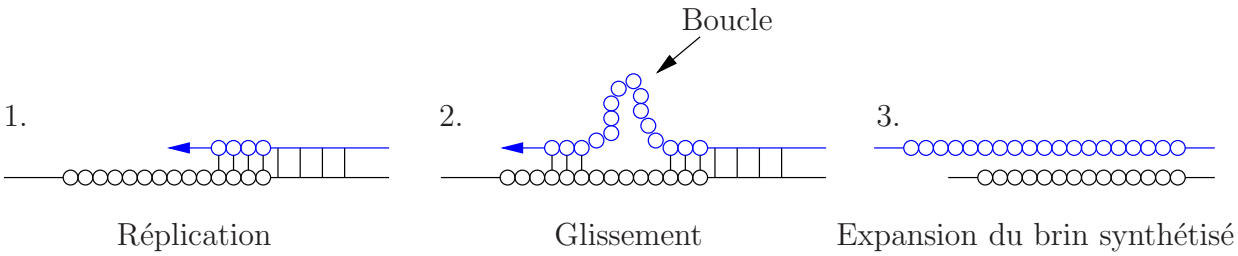


Fig. 1.11 – *Glissement à la réplication. Les cercles représentent les motifs d'une répétition en tandem.*

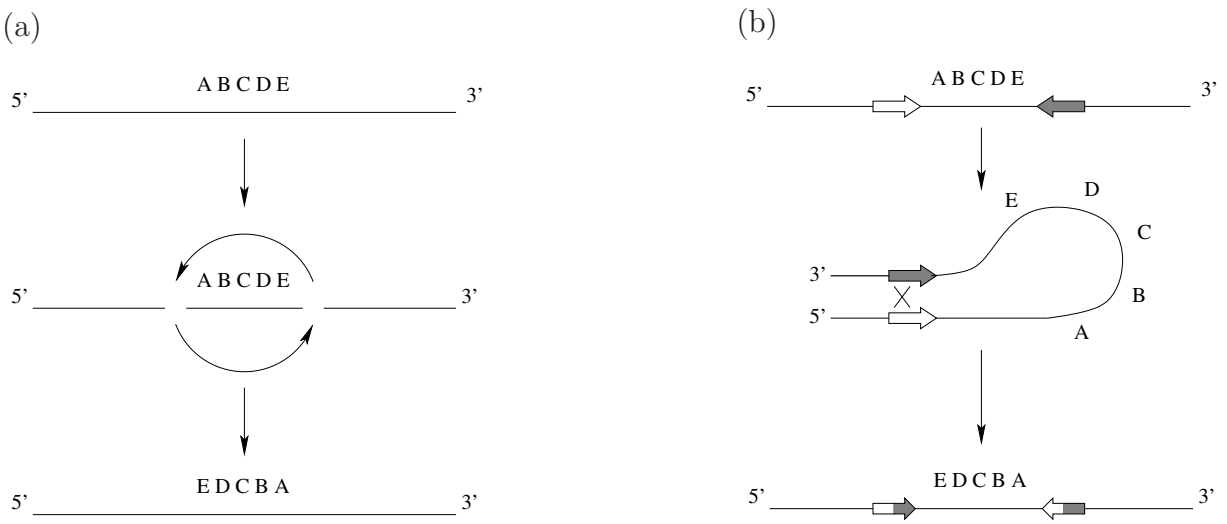


FIG. 1.12 – *Mécanismes de l'inversion (Graur et Li, 2000).*

d'ADN est la transposition d'ADN, c'est-à-dire le mouvement dans le génome de certains éléments répétés dispersés (L1 et ALU chez l'homme).

1.5.3.4 Inversions

L'**inversion** est un réarrangement de l'ADN qui peut se produire à la suite d'un de ces deux processus :

- cassure d'un chromosome suivie d'un mauvais ré-assemblage, comme montré à la figure 1.12 (a) ;
- crossing-over intrachromosomique entre deux segments homologues qui sont orientés de manière opposée, montré à la figure 1.12 (b). La croix représente le crossing-over, les segments homologues sont représentés par les flèches, leur orientation est donnée par le sens de la flèche.

Certaines inversions peuvent impliquer une bande d'ADN de centaines ou milliers de kilobases de longueur.

1.5.4 Processus de mutation des minisatellites

Les minisatellites, comme les autres séquences d'ADN, subissent les différents types de mutations que nous venons de présenter. Des études montrent que les minisatellites subissent des mutations principalement lors de la méiose. On sait également qu'ils subissent peu de crossing-over égal ou inégal et peu de glissement à la réplication. Par contre des événements de type conversion génique jouent un rôle dans le maintien de leur variabilité. De plus, une cassure double brin décalée comme événement initial d'un processus de mutation nommé SSA, *Single Strand Annealing*, a été proposée pour expliquer la variabilité des minisatellites (Buard et Jeffreys, 1997, Buard et al., 2000). Ce phénomème est une sorte de

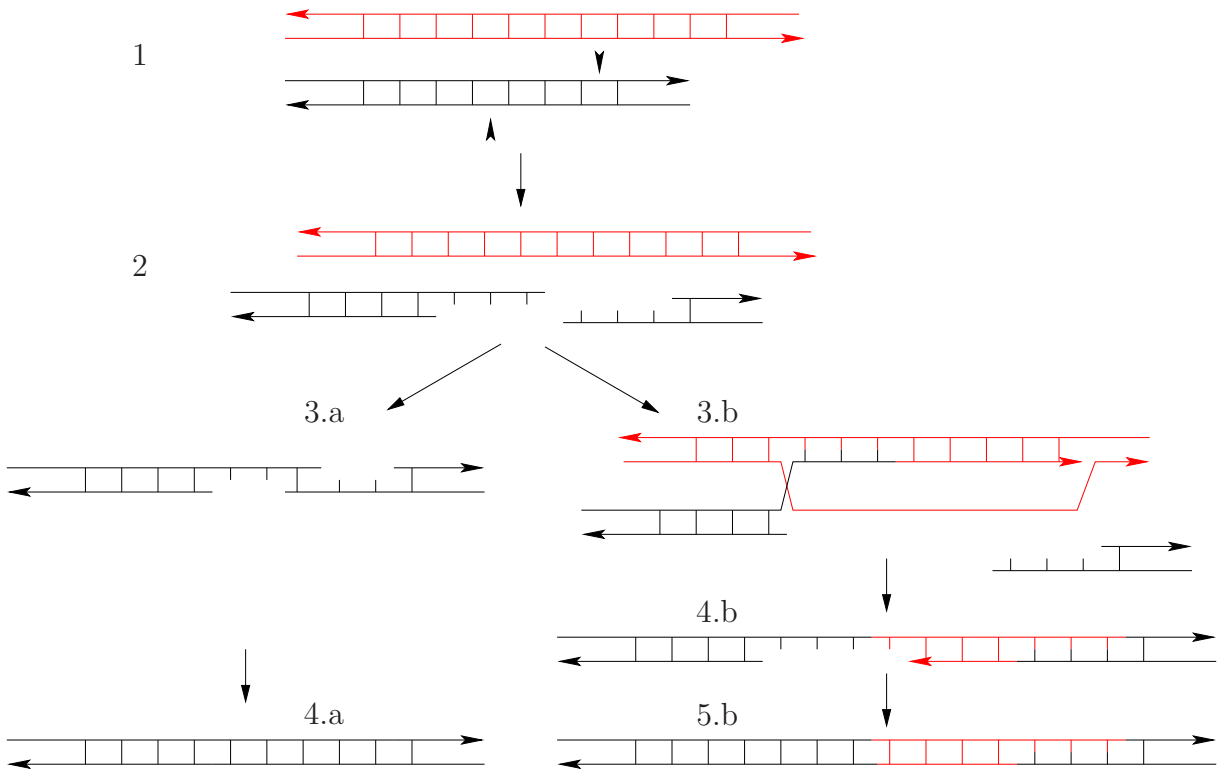


Fig. 1.13 – *Mutation spécifique des minisatellites : cassure double brin décalée suivie de Single Strand Annealing (SSA). Une double cassure décalée dans un double brin d'ADN donne des brins saillants, ceux-ci peuvent se ré-apparier ensemble (cas* 3.*a) mais de manière décalée, produisant une duplication intra-allélique (*4.*a) ; ou bien envahir le partenaire allélique (*3.*b). La plupart des événements d'invasion sont avortés, soit avant ou, dans une minorité de cas, après l'extension (cas montré ici), le brin allongé s'apparie alors avec l'autre brin saillant (*4.*b) (Buard et al., 2000).*

recombinaison impliquant des cassures décalées du double brin d'ADN suivi du processus de réparation. Une illustration de ce type de mutation est donnée à la figure 1.13.

L'étape 1 montre les cassures décalées des deux brins de l'ADN représenté en foncé. À l'étape 2, l'ADN ayant subi la double cassure se retrouve avec des brins « saillants ». Un de ces brins peut alors envahir l'ADN du chromosome homologue ou de la chromatide sœur, représenté en clair sur la figure. Cet événement est présenté au cas 3.b. La synthèse permet au brin « envahisseur » de s'allonger. Cependant, la plupart des événements d'invasion sont avortés après une extension limitée. Le brin étendu s'apparie alors avec l'autre brin saillant issu de la double cassure (étape 4.b) et le résultat est la présence d'un morceau de séquence du chromosome homologue ou de la chromatide sœur dans l'ADN qui a subi la cassure (montré en 5.b).

Parfois, le taux d'apparition de nouveaux allèles est extrêmement élevé. Ces minisatel-

lites, dits hypermutables, s'avèrent très utiles à la compréhension de certains mécanismes de la recombinaison et de la réplication du génome humain. Les minisatellites hypermutables ont été proposés comme pouvant être des marqueurs utiles pour étudier les effets de l'exposition à de faibles doses de radiations ionisantes (Jeffreys et al., 1997).

Ce que l'on constate sur les séquences de minisatellites, c'est que les mutations qu'ils subissent conduisent à un accroissement ou à une diminution des unités répétées en tandem. Pour désigner ces phénomènes, on appelle *duplication en tandem* l'événement qui ajoute une copie dupliquée à côté de la copie originale et *délétion en tandem* l'événement inverse. Nous employons dans la suite de ce manuscrit, les noms d'*amplification* et de *contraction* qui sont des noms plus courts pour duplication en tandem et délétion en tandem. Ces deux opérations sont prises en compte dans les modèles évolutifs SSE et ESSE que nous avons conçus pour aligner des cartes de minisatellites (cf. chapitres 4 et 6).

1.5.5 Un cas particulier : le chromosome Y

La génétique et la biologie moléculaire peuvent permettre de préciser l'origine des différentes populations humaines en étudiant les variations génétiques entre les individus. Parmi tous les marqueurs génétiques utilisés, les plus intéressants sont les uniparentaux, comme le chromosome Y (hérité du père) ou l'ADN mitochondrial (hérité de la mère), car ils échappent à la recombinaison méiotique. Ce type de marqueurs a donc été utilisé pour préciser géographiquement et historiquement l'origine de nos ancêtres communs les plus récents et les interactions entre les différentes populations humaines.

On infère généralement l'évolution des populations sous la forme d'un arbre, appelé *arbre phylogénétique* ou phylogénie. Ces arbres sont obtenus en utilisant des caractéristiques génétiques des populations.

Le chromosome Y est présent uniquement chez les mâles. Il possède au niveau des télomères deux régions homologues au chromosome X (régions pseudo-autosomales) et une grande partie centrale non recombinante qui lui est spécifique. Le chromosome Y est donc divisé en deux parties, la partie pseudo-autosomale qui peut échanger des allèles avec le chromosome X, à la manière d'une paire de chromosomes homologues, et la partie non recombinante qui ne peut pas. L'utilité de ce marqueur génétique tient au fait que sa région spécifique ne recombine pas lors de la méiose et, par conséquent, est transmise sans modification à la génération suivante. Les seules mutations qui apparaissent proviennent de nouvelles mutations touchant les cellules germinales.

L'examen des séquences de sa partie spécifique a montré que le chromosome Y varie peu dans les populations humaines. Une étude de deux de ses marqueurs polymorphes a permis de retracer les migrations récentes des populations du Moyent-Orient en Europe (Semino et al., 1996). La même étude a permis de trancher entre 2 modèles proposés pour comprendre l'expansion de l'agriculture au néolithique, le modèle culturel (expansion sans mouvement de populations) et le modèle migratoire (mouvement de populations). Les résultats de cette étude ont montré que le deuxième modèle était plus probable que le premier (Quintana-Murci et al., 1999).

Le chromosome Y a donc un mode d'évolution particulier mais aussi plus « simple » que le reste du génome. C'est pourquoi nous avons choisi d'étudier un minisatellite se trouvant sur ce chromosome, le minisatellite MSY1.

* *

*

Nous venons de voir que le polymorphisme génétique trouve son origine dans les mutations. La diversité des organismes diploïdes résulte à la fois du brassage génétique de ce polymorphisme lors de la formation des gamètes (méiose) et de la fécondation qui unit aléatoirement deux gamètes entre eux. Ainsi, tout nouvel individu est un être unique et original. Dans la section suivante, nous présentons les techniques utilisées en génétique ayant pour but la connaissance du génome.

1.6 Outils de détermination du patrimoine génétique

Cette section est consacrée aux différents outils de génétique moderne. Ces outils sont de natures diverses. Le séquençage, section 1.6.1, permet de connaître la suite de nucléotides correspondant aux séquences que l'on souhaite étudier. Les marqueurs génétiques, section 1.6.2, permettent de « baliser » les génomes et sont utilisés pour l'élaboration de cartes génétiques, section 1.6.3. La section 1.6.4 décrit une méthode qui facilite l'étude des acides nucléiques. Enfin, la section 1.6.5 est consacrée au génie génétique.

1.6.1 Séquençage

Le **séquençage** consiste à retrouver la suite de nucléotides correspondant au brin d'ADN étudié. La méthode utilisée aujourd'hui est celle proposée par Frederick Sanger (Sanger et al., 1977). Depuis 1977, la méthode a considérablement évolué grâce à la mise au point de séquenceurs automatiques et du marquage des nucléotides à l'aide de **fluorochromes**. Elle est devenue aujourd'hui une technique rapide et fiable utilisée fréquemment dans le diagnostic des maladies héréditaires.

En pratique, les molécules à analyser sont très longues, ce qui rend le séquençage direct impossible. La stratégie utilisée est alors de découper le brin d'ADN en fragments dont la taille varie entre 200 et 700 acides nucléiques, de les séquencer, puis de reconstruire la molécule originale par assemblage des fragments chevauchant.

1.6.2 Marqueurs génétiques

Les **marqueurs génétiques** mettent à jour les variations entre individus et peuvent ainsi permettre de les identifier. Les plus courants de ces marqueurs génétiques sont des marqueurs morphologiques, comme la couleur des yeux, la forme du visage, etc., des marqueurs biochimiques, tels que les groupes sanguins, et des marqueurs moléculaires, au

niveau de la molécule d'ADN elle-même. Les deux premiers types de marqueurs présentent le désavantage d'être moins discriminants que le dernier groupe. Par abus de langage, le terme marqueur génétique est souvent employé à la place de marqueur moléculaire. Un marqueur moléculaire est un segment d'ADN qui peut présenter différents allèles dans la population (Sancristobal-Gaudy et al., 2000).

Les marqueurs moléculaires ont plusieurs utilités :

Carte génétique Les marqueurs moléculaires sont utilisés depuis le début des années 90 pour construire des cartes génétiques. Un exemple bien connu est la première carte du génome humain établie avec plus de 5000 marqueurs microsatellites (Cohen et al., 1993) ;

Empreinte génétique Les marqueurs de type microsatellite présentent l'intérêt d'avoir souvent un polymorphisme élevé. Un individu possède deux exemplaires de chaque marqueur, l'un transmis par son père et l'autre par sa mère. Si l'on analyse plusieurs régions microsatellites, la combinaison des paires de copies détectées permet de déterminer le génotype propre à chaque individu pour ces marqueurs, c'est ce que l'on appelle une **empreinte génétique** (Jeffreys et al., 1985a) et (Jeffreys et al., 1985b). De tels marqueurs servent d'identifiants génétiques, pour différencier deux individus de la même population ou identifier des cadavres en médecine légale, ou encore pour effectuer des tests de paternité, (Gill et al., 1985), (Hill et Jeffreys, 1985) ;

Génétique des populations Les marqueurs polymorphes permettent de tracer la propagation d'une caractéristique génétique dans les populations. Par exemple, des minisatellites hautement polymorphes permettent de confirmer l'hypothèse *Out of Africa*, qui suppose que l'espèce humaine est originaire de l'Afrique et a ensuite peuplé le reste du monde (Armour et al., 1996).

1.6.3 Carte génétique

Une **carte génétique** est une représentation de la disposition relative des marqueurs le long de la molécule d'ADN.

Pour une carte génétique, les marqueurs sont ordonnés grâce à l'analyse statistique de leur ségrégation au cours des générations. Trois types de marqueurs sont préférentiellement utilisés pour la construction de carte : les **RFLP** (*Restriction Fragment Length Polymorphism*), les microsatellites et plus récemment, les SNPs. Ces trois types de marqueurs définissent des loci uniques dans le génome et sont polymorphes. La plupart de ces marqueurs sont anonymes, c'est-à-dire qu'ils correspondent à des séquences non traduites.

1.6.4 PCR

Mise au point dans les années 80, la technique de réaction de polymérisation en chaîne ou **PCR** (*Polymerase Chain Reaction*) a ouvert de nouvelles voies dans l'étude et l'analyse des acides nucléiques et des gènes. Cette technique permet d'amplifier en un nombre très élevé de copies, une séquence particulière d'ADN ou d'ARN. La PCR, combinée au séquençage,

fournit un outil particulièrement puissant pour l'analyse des séquences, notamment quand la quantité d'ADN disponible est faible.

Les minisatellites sont plus difficiles à séquencer que les autres séquences d'ADN. Ceci est dû à leur structure, qui est une répétition de motifs similaires. En 1991, Jeffreys et ses collaborateurs ont mis au point une technique PCR pour typer les unités répétées de minisatellites. Cette méthode se nomme **MVR-PCR** où MVR signifie *Minisatellite Variant Repeat* (Jeffreys et al., 1991). Elle fournit la succession des variants le long du minisatellite, où chaque variant est codé par un symbole. De telles séquences sont appelées *cartes de minisatellite*. Cette technologie a amené une amélioration majeure dans la connaissance du processus d'instabilité des minisatellites, parmi lesquelles le modèle d'initiation des mutations par cassures du double brin (SSA), le rôle des séquences flanquantes, les différences entre mutations méiotiques et somatiques et le phénomène de variabilité polarisée (Jeffreys et al., 1997, Vergnaud et Denoeud, 2000).

1.6.5 Génie génétique

La découverte d'enzymes très particulières, dont certaines sont capables de couper la molécule d'ADN en des points précis et d'autres qui peuvent « recoller » les morceaux ainsi obtenus dans un ordre différent, est à l'origine du génie génétique.

Le **génie génétique** est un ensemble de techniques permettant le transfert d'un gène étranger dans une cellule en culture (ou dans les tissus d'un animal ou d'une plante) de manière à donner aux cellules receveuses une propriété nouvelle liée au gène transféré. La finalité de ce transfert peut être la production de grandes quantités d'un produit rare, comme des hormones ou des enzymes, ou l'obtention de variétés originales de plantes ou d'animaux présentant une potentialité nouvelle liée au gène transféré, comme la résistance à un herbicide, une croissance améliorée, etc.

Les enjeux pour l'homme La *thérapie génique*, encore à l'état expérimental, consiste à transférer un ou plusieurs gènes dans les cellules pour leurs propriétés thérapeutiques. L'idée est de prélever certaines cellules du malade, d'y implanter le gène déficient et de les replacer dans l'organisme afin qu'elles produisent la bonne enzyme. La thérapie génique peut permettre de lutter efficacement contre des affections pour lesquelles la médecine n'apporte que des solutions partielles ou palliatives : maladies monogéniques rares, comme les maladies neuro-musculaires ou la mucoviscidose, ou affections plus fréquentes, comme les maladies cardiovasculaires, le diabète de type I, les cancers, les maladies du système nerveux central, etc.

Quelques enjeux scientifiques Le séquençage complet du génome humain et les connaissances acquises en génétique moléculaire permettent une approche plus rationnelle des maladies humaines. Le début du XXI^{e} siècle sera vraisemblablement marqué par la mise en évidence de la fonction de nombreux gènes avec, en corollaire, la découverte de nouvelles cibles thérapeutiques (`Genopole`).

La génétique connaît actuellement de nombreux développements à visée thérapeutique ou économique. Mais ils posent d'importants problèmes éthiques ou de société : le diagnostic

génétique ou pré-implantatoire, la pharmacogénomique, la thérapie génique, le clonage, les empreintes génétiques, ou enfin la production de médicaments ou le développement de produits d'intérêt agronomique par utilisation d'organismes génétiquement modifiés (OGM).

* *
*

Nous avons vu dans ce chapitre les notions de génétique qui nous permettent d'appréhender les applications biologiques de notre algorithme d'alignement de cartes de minisatellite.

Les minisatellites, à cause de leur variabilité de longueur, sont utiles pour la cartographie génétique, les études de médecine légale et l'exploration de la diversité génétique et de la structure des populations. Mais ils sont également associés à des maladies graves. Leur étude, par le biais des cartes de minisatellite, qui donnent accès au processus de mutation des minisatellites à un degré plus précis, peut permettre de comprendre leur fonctionnement.

Chapitre 2

Rappels d'informatique

Sommaire

2.1 Complexité algorithmique 34
2.2 Alignements de séquences 35
2.2.1 Notations 36
2.2.2 Comparaison de séquences 37
2.2.2.1 Pourquoi comparer des séquences ? 37
2.2.2.2 Comment les séquences diffèrent-elles ? 38
2.2.2.3 Analyse des différences : trace, alignement et listing . . 38
2.2.3 Distances entre séquences 41
2.2.4 Alignements 43
2.2.4.1 Alignement global 44
2.2.4.1.1 Calcul de la distance 44
2.2.4.1.2 Calcul d'un alignement optimal 45
2.2.4.1.3 Analyse de complexité 47
2.2.4.2 Les différents types d'alignements 48
2.2.4.3 Plus longue sous-séquence commune 49
2.3 Graphes 51
2.3.1 Définitions et notations 51
2.3.2 Graphes d'intervalles 53
2.3.3 Graphes de chevauchement 54
2.3.4 Graphes de cordes 55

Dans ce chapitre, nous rappelons les concepts informatiques et les algorithmes qui nous seront utiles par la suite. Nous donnons tout d'abord quelques rappels élémentaires sur ce qu'est la complexité algorithmique. Ensuite, dans section 2.2, nous parlons de l'alignement de séquences par programmation dynamique. Enfin, la dernière section de ce chapitre est consacrée aux graphes et, en particulier, aux types de graphes que nous rencontrerons dans ce travail de thèse.

2.1 Complexité algorithmique

Nous donnons ici quelques rappels élémentaires concernant la complexité algorithmique. Cette section est réalisée à partir des informations contenues dans [Garey et Johnson, 1979], [Cormen et al., 1994] et [Muller, 2003].

Lorsqu'un problème peut être résolu par un certain algorithme, après la question de la résolution, se pose la question de la *complexité* en temps et en espace de la méthode utilisée. Par complexité en espace, nous entendons la place mémoire nécessaire à l'algorithme pour fonctionner. Par complexité en temps, nous entendons le nombre d'instructions nécessaires pour atteindre la solution. Nous nous intéressons dans ce cadre au comportement de l'algorithme lorsqu'il est appliqué à des problèmes de plus en plus considérables. Généralement, lorsque l'on parle de complexité sans préciser en temps ou en espace, c'est que l'on parle de la complexité en temps.

Temps de calcul Nous notons $t(n)$ le temps de calcul d'un algorithme en fonction de la taille n des données en entrée. Cette taille est variable suivant les instances du problème traitées.

Pour rester indépendant de la machine, on considère que les opérations élémentaires (additions, multiplications, comparaisons, etc.) prennent toutes le même temps. En général, on s'intéresse au temps maximum pour un n donné, c'est la complexité dans le pire des cas.

Si une opération élémentaire prend une microseconde, notée μs, voici un tableau donnant les temps de calcul suivant la taille n du problème.

$t(n)$ / n	10	20	30	40	50	60
$\log n$	1 μs	1,3 μs	1,5 μs	1,6 μs	1,7 μs	1,8 μs
n	10 μs	20 μs	30 μs	40 μs	50 μs	60 μs
$n \log n$	10 μs	26 μs	44 μs	64 μs	85 μs	107 μs
n^2	100 μs	400 μs	900 μs	1,6 ms	2,5 ms	3,5 ms
n^3	1 ms	8 ms	27 ms	64 ms	125 ms	216 ms
n^5	0,1 s	3,2 s	24,3 s	1,7 m	5,2 m	13 m
2^n	1 ms	1 s	18 m	13 j	36 a	366 siècles
3^n	59 ms	58 m	6 m	3855 siècles	2×10^8 siècles	$1,3 \times 10^{13}$ siècles

Ordre de grandeur Comme le temps précis d'exécution dépend de l'ordinateur utilisé, nous nous intéressons à son ordre de grandeur. Si le temps d'exécution d'un algorithme A est donné par $t(n) = an^2 + bn + c$, on ne considère que le premier terme de la formule, c'est-à-dire an^2, puisque les termes d'ordres inférieurs ne sont pas vraiment significatifs lorsque n est grand. On ignore également le coefficient constant du premier terme, puisque les facteurs constants sont moins significatifs que l'ordre de grandeur lorsqu'on détermine l'efficacité des calculs pour de grandes entrées. On écrira donc que l'algorithme A a un temps d'exécution en $\theta(n^2)$, ou que l'algorithme A a une complexité en temps en $\theta(n^2)$.

On considère qu'un algorithme est plus efficace qu'un autre si son temps d'exécution dans le pire des cas a un ordre de grandeur inférieur.

La notation θ borne une fonction asymptotiquement à la fois par excès et par défaut. Lorsqu'on ne dispose que d'une borne supérieure asymptotique, on utilise la notation O.

Classes de complexité On peut classer hiérarchiquement les problèmes en fonction de la complexité de leur algorithme. Certains problèmes sont plus difficiles que d'autres. Nous définissons tout d'abord ce qu'est un algorithme polynômial, nous donnons ensuite la définition de deux classes de problèmes, P et NP, que l'on rencontre fréquemment, et enfin nous évoquons les problèmes dits NP-complets.

Définition 1 (Algorithme polynômial) *Un algorithme est dit polynômial si quel que soit n, pour des données ne prenant pas plus de n octets, l'algorithme s'exécute en moins de $C.n^k$ opérations élémentaires, où les constantes C et k sont indépendantes de n.*

En pratique, les algorithmes polynômiaux sont les seuls à pouvoir être utilisés informatiquement pour de grandes valeurs de n, et ce, quelle que soit la puissance de la machine.

Définition 2 (Classe P) *On dit qu'un problème est dans la classe P s'il existe un algorithme polynômial pour le résoudre.*

Définition 3 (Classe NP) *On dit qu'un problème est dans la classe NP s'il existe un algorithme polynômial pour vérifier qu'une solution donnée convient.*

Autrement dit, être dans P, c'est trouver une solution en temps polynômial, tandis qu'être dans NP, c'est prouver en temps polynômial qu'une proposition de réponse est une solution du problème. Tout problème qui est dans P se trouve dans NP.

Définition 4 (NP-complet) *On dit qu'un problème est NP-complet si la résolution de ce problème en temps polynômial entraîne la résolution en temps polynômial de tout problème de NP.*

Les problèmes NP-complets sont donc les plus compliqués des problèmes de la classe NP. Les problèmes d'ordonnancement de tâches, d'emploi du temps ou d'affectation de ressources par exemple sont NP-complets.

2.2 Alignements de séquences

Une séquence est une suite de caractères sur un alphabet donné. Les alignements sont une des méthodes pour comparer les séquences ; ils sont basés sur des notions de distance ou de similarité. Calculer un alignement s'effectue généralement par programmation dynamique.

Nous établissons dans un premier temps les notations que nous allons utiliser dans tout le reste de cette thèse. Nous parlons ensuite de comparaison de séquences. Dans la section 2.2.3, nous détaillons quelques distances utilisées pour comparer des séquences. Nous expliquons ensuite le calcul d'une distance entre séquences à laquelle on peut associer un alignement entre ces séquences. Enfin, nous concluons la section sur les alignements en évoquant le problème analogue de la plus longue sous-séquence commune.

Pour cette section nous avons utilisé les livres suivants : [Crochemore et al., 2001], [Setubal et Meidanis, 1997], [Gusfield, 1999] et [Sankoff et Kruskal, 1999].

2.2.1 Notations

Soit $\mathbf{\Sigma}$ un alphabet de taille finie. Σ^* est l'ensemble des mots constructibles à partir de l'alphabet Σ.

Une **chaîne** s de longueur n sur Σ, ou autrement dit, une chaîne s telle que $s \in \Sigma^*$, est une séquence de n symboles de Σ, indexés de 1 à n. La longueur de s est notée $|s|$. Dans le reste de ce manuscrit, nous employons indifféremment chaîne ou séquence.

Nous notons le $i^{\text{ème}}$ symbole de s par $s[i]$ pour tout i tel que $1 \leq i \leq n$. Pour tous entiers i, j, tels que $1 \leq i \leq j \leq n$, $s[i..j] = s[i]s[i+1]\dots s[j]$ est appelée **sous-chaîne** ou **facteur** de s. Un **préfixe** de s est une sous-chaîne de s commençant à la position 1.

Une **sous-séquence** w de s est obtenue en ôtant de s un certain nombre de caractères. Plus formellement $w = s[u_1]s[u_2]\dots s[u_k]$ est une sous-séquence de s si et seulement si :

- $k \leq n$;
- et u est une suite *croissante* à valeurs dans $\mathbb{N}^*$ telle que $1 \leq u_1$ et $u_k \leq n$.

Nous attirons votre attention sur le fait que la notion de sous-chaîne est différente de celle de sous-séquence, alors que chaîne et séquence sont la même chose.

Nous notons l'opération de **concaténation** par un **.** ; si s et r sont deux chaînes de longueurs respectives n et m, $s.r = s[1]s[2]\dots s[n]r[1]\dots r[m-1]r[m]$.

Exemple *Notations*

Soient deux chaînes $s = abcd$ et $r = ef$;

on a alors : $s[1] = a$, $s[2] = b$, $s[4] = d$, $r[2] = f$;

$s[1..3] = abc$, $r[1..2] = r = ef$;

et $s.r = abcdef$.

bc est une sous-chaîne de s, abc est un préfixe de s et bd est une sous-séquence de s.

2.2.2 Comparaison de séquences

Comparer des séquences c'est mettre en évidence la ressemblance entre ces séquences. Dans un premier temps, nous montrons pourquoi il est utile de comparer des séquences, ensuite nous expliquons comment les séquences diffèrent, et enfin nous décrivons trois modes d'analyse de ces différences, les traces, les alignements et les listings.

2.2.2.1 Pourquoi comparer des séquences ?

Il est souvent nécessaire de comparer deux séquences ou plus et de mesurer jusqu'à quel point elles diffèrent. Les applications se divisent en deux grandes parties, discrètes et continues. Dans le cas discret, les objets étudiés sont des séquences ordinaires, généralement de tailles différentes. Ces séquences peuvent s'écrire sous la forme $s = s[1]s[2]\dots s[n]$, où les éléments $s[i]$ sont pris dans un alphabet Σ. Dans le cas continu, comme le chant des oiseaux par exemple, les objets sont des fonctions continues, mais nous ne détaillons pas ce cas là. À partir de maintenant nous considérons seulement le cas discret. Les différentes applications du cas discret se font dans les domaines suivants [Sankoff et Kruskal, 1999] :

Biologie moléculaire Dans les séquences biologiques (ADN, ARN ou séquences d'acides aminés), les séquences hautement similaires impliquent en général une origine évolutive, une structure ou une fonction similaire significative, mais l'inverse n'est pas vrai.

Les êtres vivants partagent une grande partie de leur matériel génétique. Redondance et similarité sont des phénomènes centraux en biologie. Mais la similarité a ses limites, les hommes et les mouches diffèrent à de nombreux égards. Ces différences font que des similarités conservées entre les génomes sont encore plus significatives, ce qui à leur tour font de la comparaison et de l'analogie des outils très puissants en biologie.

Aujourd'hui, la méthode la plus puissante pour inférer la fonction biologique d'un gène (ou de la protéine qu'il code) est la recherche de similarité de séquences dans les bases de données d'ADN et de protéines [Gusfield, 1999]. En effet, les séquences ayant une origine commune, et donc une fonction similaire, sont relativement conservées.

Informatique L'alignement est utilisé pour la comparaison de deux fichiers, où chaque ligne est traitée comme un seul symbole (`diff` sous unix par exemple). La deuxième application en informatique est le correcteur d'orthographe (recherche de mots les plus proches dans un dictionnaire), une extension prend en compte les inversions de caractères dues aux fautes de frappe.

Langage humain Reconnaissance de la parole ou de l'orateur. Le flot de paroles peut être échantillonné en syllabes. Il s'agit de reconnaître et d'isoler des mots d'un vocabulaire limité.

Autres applications Contrôles de codes et d'erreurs, strates géologiques, anneaux des troncs d'arbres et comparaison de textes.

2.2.2.2 Comment les séquences diffèrent-elles ?

En général, la différence entre deux séquences est décomposée en différences élémentaires, c'est-à-dire en différences sur les caractères. Ces différences élémentaires constituent des opérations de transformation élémentaires pour passer d'une séquence à une autre. Il existe plusieurs différences/opérations élémentaires, parmi lesquelles :

- les *substitutions*, que nous appelons aussi mutations, c'est-à-dire le remplacement d'un caractère par un autre ;
- les *délétions* et *insertions*, appelées *indels* (raccourci anglophone pour **in**sertion-or-**del**etion), il s'agit ici du retrait ou de l'ajout d'un caractère ;
- les *amplifications* et *contractions*, elles permettent à un caractère d'être amplifié, c'est-à-dire d'augmenter son nombre d'occurrences (ces occurrences restant côte à côte), ou à l'inverse à plusieurs caractères identiques adjacents d'être contractés, c'est-à-dire réduits à un seul ;
- les *inversions*, c'est-à-dire l'échange de deux caractères adjacents.

Exemple *Différences entre séquences*

Séquences	Différences
$POMME$ et $GOMME$	Substitution $P \rightarrow G$ à la position 1
$VAIN$ et VIN	Délétion du A à la position 2
$CHAZAM$ et $CHAAAZAM$	Amplification du A à la position 3
$PLIER$ et $PILER$	Inversion du L et du I aux positions 2 et 3
$EPELER$ et $APPELLE$	Substitution $E \rightarrow A$ à la position 1 Amplification du P à la position 2 Amplification du L à la position 4 Délétion du R à la position 6

En biologie, les différences classiquement considérées sont seulement la substitution, l'insertion et la délétion, aussi nous nous limitons à ces trois opérations dans la suite de ce chapitre. Dans ce travail de thèse, nous considérons en plus les opérations d'amplification et de contraction (cf. chapitres 4 et 6).

2.2.2.3 Analyse des différences : trace, alignement et listing

Comme on vient de le voir, pour analyser la différence totale entre deux séquences, on peut regarder l'ensemble des différences élémentaires entre caractères. Une opération transforme une séquence dite *source* en une séquence dite *cible*.

Il existe au moins trois modes différents de représentation de cette analyse de différences entre séquences : les traces, les alignements et les listings [Sankoff et Kruskal, 1999]. Des

Trace

Alignement

$$\begin{pmatrix} O & C & T & O & B & R & - & E \\ O & - & T & A & - & R & I & E \end{pmatrix}$$

Listing

$OCTOBRE$	
	Délétion du C
$OTOBRE$	
	Délétion du B
$OTORE$	
	Substitution du 2^{e} O en A
$OTARE$	
	Insertion du I
$OTARIE$	

FIG. 2.1 – *Trois modes d'analyse de différences entre séquences.*

exemples de ces trois types d'analyse sont données à la figure 2.1.

Trace Une trace de s à r est la séquence source s au dessus de la séquence cible r, avec des lignes joignant des caractères de la source et de la cible. Un caractère ne peut pas être adjacent à plus d'une ligne et les lignes ne peuvent pas se croiser. Si les caractères connectés par la même ligne sont identiques, on parlera d'un appariement exact[1], (les O de la première position par exemple) et sinon d'un mésappariement ou substitution[2], (le deuxième O de $OCTOBRE$ et le A de $OTARIE$). Un caractère source n'ayant pas de ligne montre une délétion (le C de $OCTOBRE$), un caractère cible n'ayant pas de ligne, une insertion(le I de $OTARIE$).

Alignement Un alignement entre s et r est une matrice de deux lignes. La première ligne est la source s dans laquelle des caractères nuls ont éventuellement été insérés. Nous représentons les caractères nuls par des $-$ (on peut également voir selon les auteurs $\varnothing$, λ ou un blanc). Un caractère nul est aussi appelé trou. Une suite de trous est une brèche[3]. La deuxième ligne de la matrice est la cible r dans laquelle des caractères

[1] *Match* en anglais.
[2] *Mismatch* en anglais.
[3] *Gap* en anglais.

nuls ont éventuellement été insérés. La colonne $\begin{pmatrix} - \\ - \end{pmatrix}$ de caractères nuls n'est pas autorisée. Chaque colonne, également appelée paire alignée, a une signification :

$\begin{pmatrix} x \\ - \end{pmatrix}$ avec un $-$ en bas, indique la délétion de x ;

$\begin{pmatrix} - \\ y \end{pmatrix}$ ayant un $-$ en haut, indique l'insertion de y ;

$\begin{pmatrix} x \\ y \end{pmatrix}$ sans $-$, est appelée un appariement exact si $x = y$ ou une substitution de x par y si $x \neq y$.

L'alignement de la figure 2.1 correspond à la trace de la même figure.

Comme mode d'analyse, les alignements sont plus riches que les traces dans le sens où ils font une distinction entre les indels adjacents. Il peut exister plusieurs alignements qui correspondent à la même trace. Par exemple, les alignements

$$\begin{pmatrix} S & A & - & P & - & I & N \\ S & A & V & - & O & - & N \end{pmatrix} \text{ et } \begin{pmatrix} S & A & P & I & - & - & N \\ S & A & - & - & V & O & N \end{pmatrix}$$

correspondent tous les deux à la même trace :

$$\begin{array}{ccccc} S & A & P & I & N \\ | & | & & & | \\ S & A & V & O & N \end{array}$$

Listing Un listing de s à r est une alternance de séquences et d'opérations élémentaires, commençant par la séquence source s et terminant par la séquence cible r, et qui satisfait la propriété de consistance suivante : deux séquences adjacentes dans le listing doivent différer seulement par l'application d'une opération élémentaire. Le listing est en fait un algorithme qui décrit comment changer la source en la cible. Le listing de la figure 2.1 correspond à l'alignement et à la trace de la même figure.

Les listings sont un mode d'analyse plus riche que les alignements et les traces, plusieurs opérations peuvent être appliquées à la même position, alors que dans les deux autres modes d'analyse ce n'est pas possible. De plus, les listings donnent l'ordre d'application des opérations et cet ordre peut être important. En effet, à une position i, la substitution $A \to B$ peut directement précéder mais ne peut pas directement suivre la substitution $B \to C$. D'un autre côté, les opérations qui ne se produisent pas sur la même position peuvent être effectuées dans un ordre quelconque.

Pour chacun de ces modes, plusieurs analyses pour les mêmes deux séquences sont possibles, ceci est illustré à la figure 2.2 en utilisant les traces.

La multiplicité de ces analyses alternatives est une des difficultés centrales de ce domaine. Une notion de parcimonie va être introduite dans la suite pour choisir la plus courte ou la moins coûteuse de ces analyses. Une partie de la multiplicité vient du simple fait que la substitution de a par b peut également être analysée comme un couple de délétion-insertion, délétion de a et insertion de b. L'analyse la plus plausible dépend alors du contexte, comme

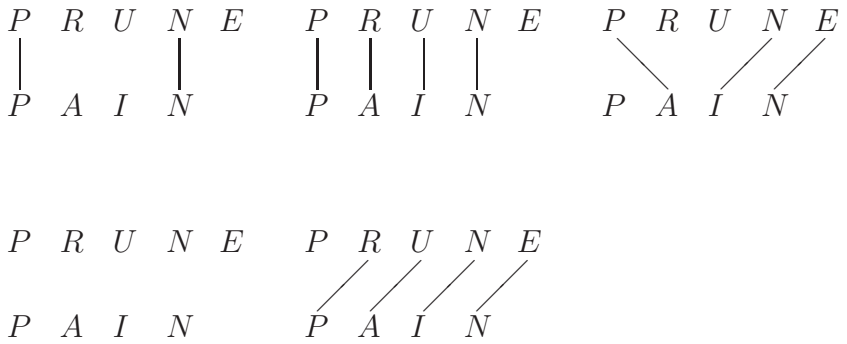


FIG. 2.2 – *Différentes analyses pour la même paire de séquences.*

par exemple des coûts donnés à ces opérations, on choisit dans ce cas la combinaison d'opérations la moins chère, ou des règles d'applications de ces opérations, et on choisit alors la combinaison qui n'enfreint aucune règle.

Dans les deux principaux domaines d'application, la biologie et l'informatique, les listings correspondent directement aux mécanismes naturels par lesquels les séquences sont supposées évoluer. Les différents types d'analyse que nous avons vus mènent au même résultat dans beaucoup de cas, mais les calculs basés sur les alignements ou les traces sont beaucoup plus rapides que les calculs basés sur les listings. Ainsi, les listings sont essentiellement d'un intérêt théorique, tandis que les alignements et les traces sont utilisés en pratique. En fait les traces et les alignements résument des listings de même coût.

2.2.3 Distances entre séquences

On dit qu'une fonction $d : \Sigma^* \times \Sigma^* \to \mathbb{R}$ est une distance sur Σ^* si les quatre propriétés suivantes sont respectées pour tous $s, r \in \Sigma^*$:

positivité : $d(s, r) \geq 0$;

séparation : $d(s, r) = 0 \Leftrightarrow s = r$;

symétrie : $d(s, r) = d(r, s)$;

inégalité triangulaire : $d(s, r) \leq d(s, t) + d(t, r) \quad \forall t \in \Sigma^*$.

On peut considérer plusieurs distances différentes entre les séquences. Certaines peuvent être appliquées seulement lorsque les séquences sont de longueurs égales, comme par exemple la distance de Hamming, qui est simplement le nombre de positions pour lesquelles les éléments correspondant dans chacune des séquences sont différents. Dans cette méthode, $s[i]$ correspond à $r[i]$ et la comparaison entre eux est exprimée par un terme qui vaut 0 si $s[i] = r[i]$ et 1 si $s[i] \neq r[i]$ [Sankoff et Kruskal, 1999].

Nous nous intéressons à une distance classique, permettant de comparer deux chaînes de longueurs différentes, la distance d'édition. Cette distance permet de transformer une

chaîne en une autre par une série d'opérations d'édition appliquées aux caractères. Les opérations d'édition permises sont :

- l'insertion ;
- la délétion ;
- la substitution.

On peut noter I l'opération d'insertion, D, l'opération de délétion, S, l'opération de substitution, R, la non-opération ou appariement exact, alors la chaîne $OCTOBRE$ peut être « éditée » pour devenir $OTARIE$ de la manière suivante :

R	D	R	S	D	R	I	R	
O	C	T	O	B	R		E	*(Les opérations d'édition sont indi-*
O		T	A		R	I	E	*quées sur la première ligne)*

Le O est inchangé, le C est délété, le T est inchangé, le O est substitué par le A, le B est délété, le R est inchangé, le I est inséré, le E est inchangé. Ici la distance est de 4 opérations. La distance entre deux séquences peut être vue comme un coût de transformation pour passer de l'une à l'autre.

Définition 5 (Distance d'édition) *La distance d'édition entre deux chaînes est définie comme le nombre minimal d'opérations d'édition (insertions, délétions, substitutions) nécessaires pour transformer la première séquence en la seconde.*

Nous notons la distance d'édition entre deux chaînes s et r par $Lev(s, r)$.

La distance d'édition est parfois référencée comme la distance de Levenshtein en reconnaissance de l'article [Levenshtein, 1966] par V. Levenshtein où la distance d'édition a été certainement discutée pour la première fois [Gusfield, 1999].

Une généralisation de la distance d'édition est d'autoriser l'association d'un poids, coût ou score arbitraire à chaque opération d'édition[4]. Pour $a, b \in \Sigma$, on note :

- $Sub(a, b)$ le coût de la substitution de la lettre a par la lettre b ;
- $Del(a)$ le coût de la délétion de la lettre a ;
- $Ins(b)$ le coût de l'insertion de la lettre b.

Ces coûts sont des réels positifs. On suppose implicitement que les coûts sont indépendants des positions auxquelles les opérations sont réalisées. Nous notons $T_{s,r}$ l'ensemble des suites d'opérations d'édition permettant de transformer s en r. Un élément $\sigma \in T_{s,r}$ est donc une suite d'opérations élémentaires transformant s en r. Le coût d'un tel élément est la somme des coûts des opérations qu'il contient.

Définition 6 (Distance d'édition généralisée) *La distance d'édition généralisée, que nous notons Edit, entre deux chaînes s et r est définie de la manière suivante :*

$$Edit(s, r) = \min\{\textit{coût de } \sigma \mid \sigma \in T_{s,r}\}$$

[4]Les termes poids et coût sont très utilisés en informatique alors que le terme score est plutôt employé en biologie.

La fonction $Edit$ est une distance sur Σ^* si et seulement si Sub est une distance sur Σ et si $Del(a) = Ins(a) > 0, \forall a \in \Sigma$. Le problème du calcul de $Edit(s, r)$ consiste à déterminer une suite d'opérations d'édition pour transformer s en r en minimisant le coût total des opérations utilisées. Toute solution, qui n'est pas nécessairement unique, peut s'énoncer comme une suite d'opérations élémentaires de substitution, délétion et insertion. Elle peut également se représenter sous forme d'un alignement [Crochemore et al., 2001].

2.2.4 Alignements

Un alignement est la manière de positionner une séquence au dessus de l'autre dans le but de mettre en évidence les correspondances entre les caractères similaires.

Nous avons vu une définition de l'alignement à la page 39 ainsi que les différents types de paires alignées, ou colonnes. Nous associons maintenant un coût à chaque paire alignée pour les deux distances que nous avons définies. Pour $a, b \in \Sigma$, on a :

Distance d'édition	Distance d'édition généralisée
$Lev\begin{pmatrix} a \\ b \end{pmatrix} = 1$	$Edit\begin{pmatrix} a \\ b \end{pmatrix} = Sub(a, b)$
$Lev\begin{pmatrix} a \\ - \end{pmatrix} = 1$	$Edit\begin{pmatrix} a \\ - \end{pmatrix} = Del(a)$
$Lev\begin{pmatrix} - \\ b \end{pmatrix} = 1$	$Edit\begin{pmatrix} - \\ b \end{pmatrix} = Ins(b)$

Le coût d'un alignement est alors défini comme la somme des coûts associés à chacune des paires alignées. Ceci est valable quelle que soit la distance d'édition utilisée.

Nombre d'alignements Le nombre d'alignements entre deux mots est exponentiel. Si l'on note $f(n)$ le nombre d'alignements différents entre deux mots de même longueur n, et $\begin{pmatrix} n \\ k \end{pmatrix}$ la combinaison de k éléments parmi n, on a :

$$f(n) = \sum_{k=0}^{n} \begin{pmatrix} n+k \\ k \end{pmatrix} \begin{pmatrix} n \\ k \end{pmatrix}$$

Ce résultat vient du fait que pour aligner deux mots, on peut commencer par placer k trous, $k \leq n$, dans le premier mot. On a $\begin{pmatrix} n+k \\ k \end{pmatrix}$ façons différentes de placer ces trous. Ensuite, pour que l'alignement soit valide, on doit placer les trous du deuxième mot sous les lettres du premier mot. Comme les deux mots sont de longueurs égales, il y a également k trous à placer dans le second mot, on a donc $\begin{pmatrix} n \\ k \end{pmatrix}$ choix possibles pour les placer. Il faut alors faire varier le nombre de trous de 0 à n et on obtient le résultat indiqué [Hamel, 2002,

pages 13-14]. S. Hamel, en utilisant un résultat de [Bergeron et al., 1997], montre également que :

$$f(n) \text{ est asymptotiquement équivalent à } (3+2\sqrt{2})^n.$$

Elle donne en exemple le nombre d'alignements entre deux protéines de longueur 300, qui est $f(300) \sim 10^{230}$.

Types d'alignement Il existe différents types d'alignement entre deux séquences : global, local, semi-global, avec brèches. On peut également comparer plus de deux séquences en faisant de l'alignement multiple. Nous évoquons les différences entre ces types d'alignement à la fin de cette section, nous détaillons seulement le calcul de l'alignement global. Dans la suite, nous utilisons la distance d'édition généralisée. Les mêmes résultats peuvent être obtenus pour la distance d'édition en posant :

- $\forall a, b \in \Sigma, Sub(a,b) = \begin{cases} 1 \text{ si } a \neq b \\ 0 \text{ sinon} \end{cases}$;
- $\forall a \in \Sigma, Del(a) = Ins(a) = 1$.

2.2.4.1 Alignement global

Dans cette section, nous nous tournons vers la question algorithmique du calcul de la distance d'édition généralisée et de l'alignement de deux chaînes. De nombreux algorithmes ont été publiés et tous emploient la programmation dynamique ([Needleman et Wunsch, 1970, Smith et Waterman, 1981] par exemple, voir aussi [Sankoff, 2000]). La programmation dynamique résout les problèmes en combinant les solutions de sous-problèmes. Cette méthode résout chaque sous-problème une seule fois et mémorise sa solution dans une matrice, épargnant ainsi le recalcul de la solution chaque fois que le sous-problème est rencontré [Cormen et al., 1994].

2.2.4.1.1 Calcul de la distance

À partir des deux chaînes s et $r \in \Sigma^*$ de longueurs respectives n et m, on définit la matrice $\mathcal{A}$ comportant $n+1$ lignes et $m+1$ colonnes (indexées à partir de 0) par :

$$\mathcal{A}(i,j) = Edit(s[1..i], r[1..j]) \quad \forall i \in [1,n], \forall j \in [1,m].$$

Autrement dit, $\mathcal{A}(i,j)$ est le coût minimal d'une suite d'opérations d'édition permettant de transformer les i premiers caractères de s en les j premiers caractères de r. En utilisant cette notation, la distance d'édition généralisée entre s et r est précisément la valeur $\mathcal{A}(n,m)$.

Nous remplissons la matrice $\mathcal{A}$ en deux phases :

1. Initialisation :

$$\mathcal{A}(0,0) = 0,$$

$$\mathcal{A}(i,0) = \mathcal{A}(i-1,0) + Del(s[i]) \quad \forall i \in [1,n],$$

$$\mathcal{A}(0,j) = \mathcal{A}(0,j-1) + Ins(r[j]) \quad \forall j \in [1,m] \ ;$$

2. Récurrence :

$$\mathcal{A}(i,j) = \min \begin{cases} \mathcal{A}(i-1,j-1) + Sub(s[i],r[j]) \\ \mathcal{A}(i-1,j) + Del(s[i]) \\ \mathcal{A}(i,j-1) + Ins(r[j]) \end{cases} \qquad \forall i \in [1,n], \forall j \in [1,m].$$

La valeur à la position (i,j) dans la matrice $\mathcal{A}$, avec $i,j > 0$, ne dépend ainsi que des valeurs aux positions $(i-1,j-1)$, $(i-1,j)$ et $(i,j-1)$ (comme illustré à la figure 2.3). Ces trois choix correspondent aux trois types de paires alignées qui permettent d'étendre un alignement : $\begin{pmatrix} x \\ y \end{pmatrix}$, $\begin{pmatrix} x \\ - \end{pmatrix}$ et $\begin{pmatrix} - \\ y \end{pmatrix}$. La validité de ce procédé de calcul est démontrée dans [Crochemore et al., 2001, Chapitre 7, p. 232-234].

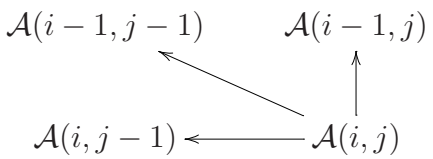


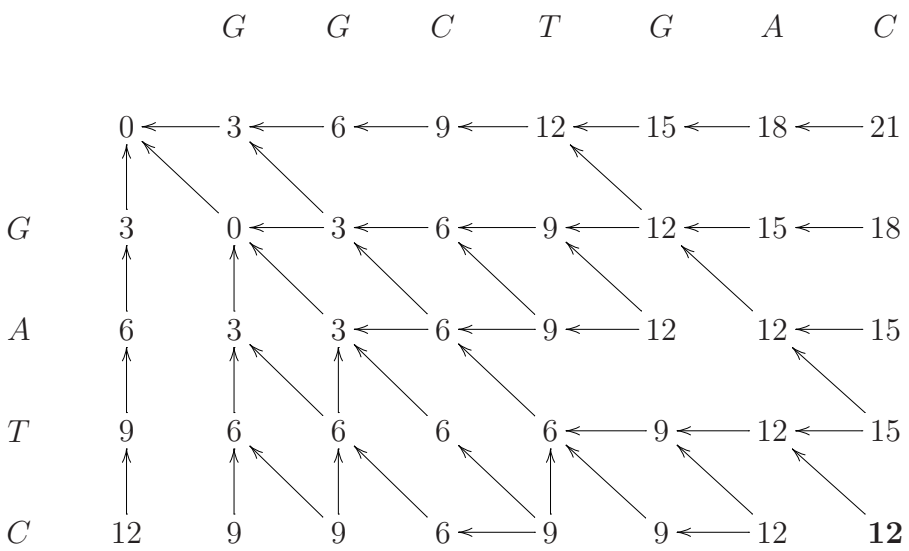
FIG. 2.3 – *La valeur $\mathcal{A}(i,j)$, pour $i,j > 0$, ne dépend que des valeurs aux trois positions voisines : $(i-1,j-1)$, $(i-1,j)$ et $(i,j-1)$.*

À la figure 2.4, nous donnons un exemple de calcul de matrice de programmation dynamique. Pour remplir la matrice, nous calculons d'abord les valeurs d'initialisation, la ligne et la colonne 0, ensuite nous procédons ligne à ligne, ainsi, lorsque nous remplissons une case, les valeurs voisines (au-dessus, en diagonale et à gauche) sont déjà calculées.

2.2.4.1.2 Calcul d'un alignement optimal

Nous venons de voir comment calculer le coût de transformation de s en r, mais comment lui associer un alignement optimal ? Le moyen le plus simple est de mettre des *pointeurs* dans la matrice au moment du calcul des valeurs.

En particulier, quand une cellule (i,j) est calculée, mettre un pointeur :
- de la cellule (i,j) vers la cellule $(i-1,j-1)$ si $\mathcal{A}(i,j) = \mathcal{A}(i-1,j-1) + Sub(s[i],r[j])$;
- de (i,j) vers $(i-1,j)$ si $\mathcal{A}(i,j) = \mathcal{A}(i-1,j) + Del(s[i])$;
- de (i,j) vers $(i,j-1)$ si $\mathcal{A}(i,j) = \mathcal{A}(i,j-1) + Ins(r[j])$.



Dans cet exemple $s = GATC$, $r = GGCTGAC$ et :
- $Sub(a, b) = 3$ si $a \neq b, Sub(a, b) = 0$ si $a = b$ $\forall a, b \in \Sigma$;
- et $Ins(a) = Del(a) = 3$, $\forall a \in \Sigma$.

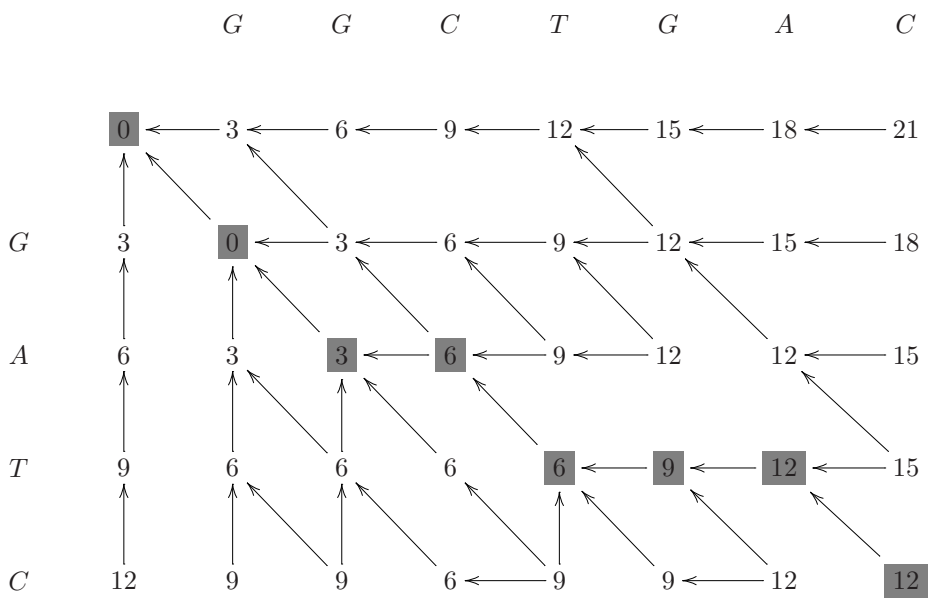
Fig. 2.4 – *Exemple de matrice de programmation dynamique. Les flèches, ou pointeurs, indiquent quelles ont été les valeurs utilisées pour remplir la case dont elles sont issues. Le chiffre en gras, 12, donne la distance entre s et r.*

Cette règle s'applique également aux cellules des lignes et colonnes 0. D'où chaque cellule de la ligne 0 pointe vers la cellule à sa gauche, et chaque cellule de la colonne 0, vers la cellule du dessus. Pour les autres cellules, il est possible (et commun) que plus d'un pointeur sortent de la case.

Les pointeurs permettent de retrouver facilement un alignement optimal, en effet, il suffit simplement de suivre n'importe quel chemin de la cellule (n, m) vers la cellule $(0, 0)$. À partir de la position (i, j), on visite, parmi les trois positions voisines, $(i - 1, j - 1)$, $(i - 1, j)$ et $(i, j - 1)$, l'une de celle dont la valeur associée a produit la valeur de $\mathcal{A}(i, j)$.

L'alignement est retrouvé à partir du chemin, en interprétant :
- chaque flèche diagonale, de (i, j) vers $(i - 1, j - 1)$, comme un appariement exact entre $s[i]$ et $r[j]$, si $s[i] = r[j]$, ou comme une substitution si $s[i] \neq r[j]$;
- chaque flèche verticale, de (i, j) vers $(i - 1, j)$, comme la délétion du caractère $s[i]$ de s (c'est-à-dire l'insertion d'un $-$ dans r) ;
- et chaque flèche horizontale, de la cellule (i, j) vers la cellule $(i, j - 1)$, comme une insertion du caractère $r[j]$ dans s (c'est-à-dire l'insertion d'un $-$ dans s).

Un exemple d'alignement associé à un chemin est donné à la figure 2.5. Toutes les cellules sauf $(0, 0)$ ont un pointeur sortant, donc aucun chemin ne peut se retrouver dans une impasse.



$$\begin{pmatrix} G & A & - & T & - & - & C \\ G & G & C & T & G & A & C \end{pmatrix}$$

FIG. 2.5 – *Parcours arrière dans la matrice de programmation dynamique de l'exemple de la figure 2.4. Nous partons de la case* $(4, 7)$*, en bas à droite, et nous remontons dans la matrice jusqu'à la case* $(0, 0)$ *en suivant les flèches, le chemin que nous empruntons est visualisé par les cases grisées. En suivant ce chemin, nous obtenons l'alignement montré en dessous de la matrice.*

Les pointeurs, construits en même temps que le calcul des valeurs de la matrice, permettent plus que retrouver un alignement optimal, ils permettent de retrouver *tous* les alignements optimaux.

Si l'on veut calculer tous les alignements optimaux entre s et r, il suffit de remonter par tous les chemins possibles. Par exemple, il existe deux autres alignements optimaux pour l'exemple des figures 2.4 et 2.5, car il existe deux autres chemins possibles pour remonter de la case $(4, 7)$ à la case $(0, 0)$:

$$\begin{pmatrix} G & - & A & T & - & - & C \\ G & G & C & T & G & A & C \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} - & G & A & T & - & - & C \\ G & G & C & T & G & A & C \end{pmatrix}$$

2.2.4.1.3 Analyse de complexité

L'algorithme d'alignement utilise $\theta(nm)$ unités de mémoire pour stocker la matrice de programmation dynamique. Pour calculer la valeur d'une cellule (i, j), il regarde seulement les valeurs de $(i-1, j-1)$, $(i-1, j)$ et $(i, j-1)$, ainsi que les caractères $s[i]$ et $r[j]$. D'où, remplir une cellule demande un nombre constant d'examens de cellule, d'opérations

arithmétiques et de comparaisons de caractères. Il y a $O(nm)$ cellules dans la matrice, donc en utilisant la programmation dynamique, la distance d'édition généralisée $\mathcal{A}(n, m)$ peut être calculée en temps $O(nm)$. Comme n'importe quel chemin de pointeurs de (n, m) à $(0, 0)$ donne un alignement optimal, un alignement optimal peut être trouvé en temps $O(n + m)$.

2.2.4.2 Les différents types d'alignements

Nous donnons ici succinctement quelques notions sur les différents types d'alignements.

Alignement global Celui que nous avons décrit dans la section précédente. Il permet de détecter les ressemblances générales entre deux séquences. Ce type d'alignement est nommé global car il prend en compte l'intégralité des deux séquences.

Alignement local Au lieu de s'intéresser à un alignement global entre s et r, il est souvent plus pertinent de déterminer un meilleur alignement entre un facteur de s et un facteur de r. La notion de distance n'est pas appropriée, on utilise plutôt une notion de similarité, pour laquelle les égalités entre caractères sont valuées positivement, et les inégalités, les insertions et les délétions sont valuées négativement. De la même manière que pour l'alignement global on remplit une matrice de programmation dynamique. Les différences par rapport à la récurrence de l'alignement global (p. 44-45) sont notées en gras :

1. Initialisation :

$$\mathcal{A}(0, 0) = 0,$$

$$\mathcal{A}(i, 0) = \mathbf{0} \quad \forall i \in [1, n],$$

$$\mathcal{A}(0, j) = \mathbf{0} \quad \forall j \in [1, m] \ ;$$

2. Récurrence :

$$\mathcal{A}(i, j) = \max \begin{cases} \mathbf{0} \\ \mathcal{A}(i-1, j-1) + Sub(s[i], r[j]) \\ \mathcal{A}(i-1, j) + Del(s[i]) \\ \mathcal{A}(i, j-1) + Ins(r[j]) \end{cases} \qquad \forall i \in [1, n], \forall j \in [1, m].$$

Le 0 dans la récurrence permet de démarrer l'alignement dans n'importe quelle case et ainsi, de sélectionner les facteurs de manière optimale. Pour trouver l'alignement local optimal, il suffit de rechercher la plus grande valeur dans la matrice $\mathcal{A}$. On retrace ensuite le chemin en suivant les pointeurs jusqu'à une case contenant un 0. On peut reconstruire l'alignement à partir du chemin comme expliqué à la section 2.2.4.1.2.

Alignement semi-global L'alignement semi-global est, comme son nom l'indique, très similaire à l'alignement global, à la différence qu'il ne pénalise pas les brèches au début et à la fin de l'alignement. Il permet donc de détecter le chevauchement ou l'inclusion de séquences.

Alignement avec brèches Dans les séquences génétiques, il est quelquefois souhaitable de pénaliser moins fortement les longues brèches. Les trous ne sont donc plus comptabilisés indépendamment les uns des autres. On définit une fonction de coût des brèches, qui donne le coût de la brèche en fonction de sa longueur. Cette fonction peut être *affine*, c'est-à-dire de la forme $c(t) = o + e \times (t-1)$, où o est le coût d'ouverture de la brèche, e le coût d'extension et t la longueur de la brèche (o et e sont des valeurs négatives). Avec une fonction de brèche affine, on peut calculer l'alignement optimal en temps $O(nm)$ [Gotoh, 1982].

Alignement multiple La notion d'alignement multiple est la généralisation naturelle du cas à deux séquences. Par exemple, si nous avons des séquences de protéines provenant d'espèces différentes et ayant des fonctions similaires, nous voulons savoir quelles parties des séquences sont similaires et quelles parties sont différentes.

Pour valuer un alignement multiple, on utilise par exemple le score SP (*Sum of Pairs*) pour donner un coût à chaque colonne, le coût de l'alignement étant la somme des coûts de chaque colonne.

$$\text{score}SP\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \sum_{1 \leq i < j \leq k} (Sub(x_i, x_j))$$

$$\text{où } x_1, x_2, \ldots, x_k \in \Sigma \cup \{-\} \text{ et } Sub(-,-) = 0.$$

L'approche exacte pour trouver l'alignement multiple de score SP maximum est de construire une matrice de programmation dynamique de dimension le nombre de séquences à aligner. S'il y a k séquences de taille n, la matrice sera de dimension n^k, le calcul d'une case dépendra des $2^k - 1$ cases voisines, et le temps de calcul de chaque score SP candidat sera $\frac{k(k-1)}{2}$, d'où une complexité totale en $O(n^k 2^k k^2)$. Le problème de décision associé est NP-complet. Il existe cependant des heuristiques [Higgins et Taylor, 2002, Chapitre 3], parmi elles :

- ClustalW [Thompson et al., 1994] qui fait de l'alignement itératif en prenant pour guide l'arbre phylogénétique construit à partir de la matrice de distance des séquences à aligner ;
- DCA [Stoye et al., 1997] qui utilise une approche « diviser pour régner » ;
- et DIALIGN 2 [Morgenstern, 1999] qui se base sur des alignements locaux.

2.2.4.3 Plus longue sous-séquence commune

Ce problème est connu sous l'acronyme anglophone *LCS* pour *Longest Common Subsequence*. La définition de sous-séquence est donnée page 36.

Soient w, s et r trois chaînes. w est une sous-séquence commune de s et r si w est à la fois une sous-séquence de s et une sous-séquence de r. w est la plus longue sous-séquence commune de s et r si :

1. w est une sous-séquence commune de s et r ;
2. Il n'existe aucune chaîne t telle que :
   - t est une sous-séquence commune de s et r,
   - $|t| > |w|$.

Deux chaînes peuvent avoir plusieurs plus longues sous-séquences communes.

Définition 7 (Problème LCS) *Soient deux chaînes s et r, le problème LCS est de trouver une plus longue sous-séquence commune de s et r.*

On note $LCS(s,r)$ une plus longue sous-séquence commune de s et r et $lcs(s,r)$ sa longueur. Par exemple, si $s = GCTAT$ et $r = CGATTA$, alors $LCS(s,r) = GTT$ et $lcs(s,r) = 3$.

Le problème LCS est connexe au problème de la distance d'édition généralisée. Si l'on donne les coûts suivants aux opérations élémentaires :

- $Sub(a,b) = \begin{cases} 0 & \text{si } a = b \\ 2 & \text{si } a \neq b \end{cases}$;
- $Del(a) = 1$;
- $Ins(a) = 1$.

Alors $Edit(s,r) = |s| + |r| - 2 * |lcs(s,r)|$. Le problème LCS, comme le problème de la distance d'édition généralisée, trouve des applications en informatique et en biologie moléculaire.

Un algorithme simple de programmation dynamique trouve la longueur de $LCS(s,r)$. Pour i et j tels que $0 \leq i \leq n$ et $0 \leq j \leq m$, il remplit une matrice de programmation dynamique $\mathcal{M}$ par la récurrence suivante :

$$\mathcal{M}(i,j) = \begin{cases} 0 & \text{pour } i = 0 \text{ ou } j = 0 \\ \mathcal{M}(i-1,j-1) + 1 & \text{si } s[i] = r[j] \\ \max(\mathcal{M}(i-1,j), \mathcal{M}(i,j-1)) & \text{si } s[i] \neq r[j] \end{cases}$$

La case $\mathcal{M}(n,m)$ donne la longueur de $LCS(s,r)$. Pour trouver $LCS(s,r)$, il suffit de placer des pointeurs lors du calcul de la matrice, d'une manière similaire à celle expliquée à la section 2.2.4.1.2. Ensuite, il faut suivre un chemin de pointeurs de la case (n,m) à la case $(0,0)$, en relevant seulement les caractères correspondant aux cases (i,j) dont la valeur est $\mathcal{M}(i-1,j-1)+1$.

2.3 Graphes

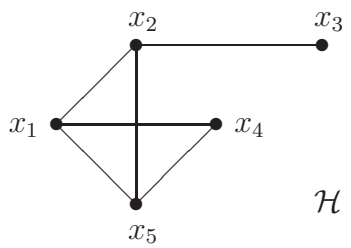
Dans cette section, nous rappelons tout d'abord quelques notions élémentaires concernant les graphes et donnons les notations utilisées. Dans les sections 2.3.2, 2.3.3 et 2.3.4 nous définissons les trois classes de graphes dont nous nous servons dans la suite. Le site de Lorna Stewart [`Stewart`] donne un aperçu des différentes classes de graphes existantes.

2.3.1 Définitions et notations

Définition 8 (Graphe) *Un graphe $\mathcal{G}$ est un ensemble de sommets, noté V, et un ensemble d'arêtes, noté E.*

Définition 9 (Arête) *Une arête est une paire non ordonnée de sommets. Une arête $e \in E$ peut s'écrire de la manière suivante : $e = \{x, y\}$ avec $x, y \in V$.*

EXEMPLE *Graphe*



Le graphe $\mathcal{H} = (V, E)$ représenté ci-dessus est défini de la manière suivante :
$V = \{x_1, x_2, x_3, x_4, x_5\}$;
$E = \{\{x_1, x_2\}, \{x_1, x_4\}, \{x_1, x_5\}, \{x_2, x_3\}, \{x_2, x_5\}, \{x_4, x_5\}\}$.

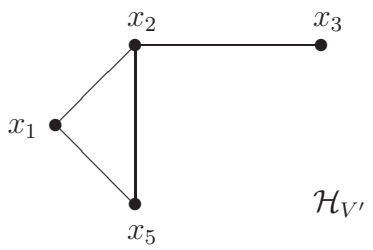
Une *boucle* est une arête de la forme $\{x, x\}$. Un graphe est dit *simple* s'il est sans boucle et s'il n'existe pas plus d'une arête entre deux sommets quelconques.

En donnant un sens aux arêtes d'un graphe, on obtient un *graphe orienté*. Un graphe orienté $\mathcal{D} = (V, A)$ est défini par l'ensemble V de ses sommets et par l'ensemble A de ses arêtes orientées, appelées *arcs*. Un arc est défini par une paire ordonnée de sommets.

Dans la suite, nous considérons des graphes simples, définis par un ensemble fini de sommets.

Définition 10 (Sous-graphe induit) *Soient $\mathcal{G} = (V, E)$ un graphe et $V' \subseteq V$ un sous-ensemble de sommets. L'ensemble des arêtes $e \in E$ dont les sommets extrémités appartiennent à V' est noté $E|V'$. Le graphe $(V', E|V')$ est appelé sous-graphe induit par V' et est noté $\mathcal{G}_{V'}$.*

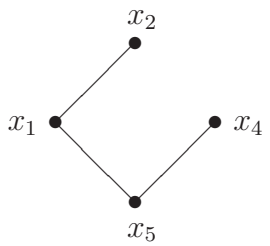
Exemple *Sous-graphe induit*



Sous-graphe du graphe $\mathcal{H}$ de l'exemple précédent induit par $V' = \{x_1, x_2, x_3, x_5\}$

Définition 11 (Sous-graphe) *Un graphe $\mathcal{G} = (V', E')$ est un sous-graphe de $\mathcal{G}$ si $V' \subseteq V$ et $E' \subseteq E$.*

Exemple *Sous-graphe*



Un sous-graphe de $\mathcal{H}$

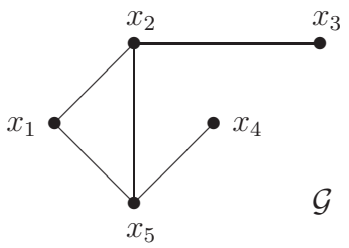
Définition 12 (Stable) *Un* **stable** *est un sous-ensemble de sommets $V' \subseteq V$ non connectés deux à deux.*

Autrement dit, le sous-graphe induit par un stable est sans arête.

Définition 13 (Nombre de stabilité) *Le* **nombre de stabilité** *du graphe $\mathcal{G}$ est la cardinalité maximale d'un stable de ce graphe $\mathcal{G}$. Le nombre de stabilité est notée $\alpha(\mathcal{G})$.*

Un stable de cardinalité maximale est appelé **stable max**. Pour les graphes $\mathcal{G}$ où chaque sommet est pondéré par la fonction de poids ω, on note $\alpha_\omega(\mathcal{G})$ le poids d'un stable de poids maximal, le poids d'un ensemble étant la somme des poids des sommets qu'il contient. Il peut exister plusieurs stables max ainsi que plusieurs stables de poids maximal dans un même graphe.

EXEMPLE *Stable max et stable de poids maximal*



Fonction de poids ω :
$\omega(x_1) = 1$
$\omega(x_2) = 3$
$\omega(x_3) = 1$
$\omega(x_4) = 1$
$\omega(x_5) = 3$

On a $\alpha(\mathcal{G}) = 3$, un stable max de $\mathcal{G}$ est $\{x_1, x_3, x_4\}$;
et $\alpha_\omega(\mathcal{G}) = 4$, un stable de poids maximal de $\mathcal{G}$ pondéré par ω est par exemple $\{x_2, x_4\}$.

Dans un graphe quelconque, le problème de trouver un stable max est un problème NP-complet [Garey et Johnson, 1979].

2.3.2 Graphes d'intervalles

Définition 14 (Graphe d'intervalles) *Un graphe $\mathcal{G} = (V, E)$ est un graphe d'intervalles si et seulement s'il existe une bijection φ de V vers une famille I d'intervalles de la droite réelle telle que $\{x, y\} \in E \Leftrightarrow \varphi(x) \cap \varphi(y) \neq \varnothing$.*

Autrement dit si :

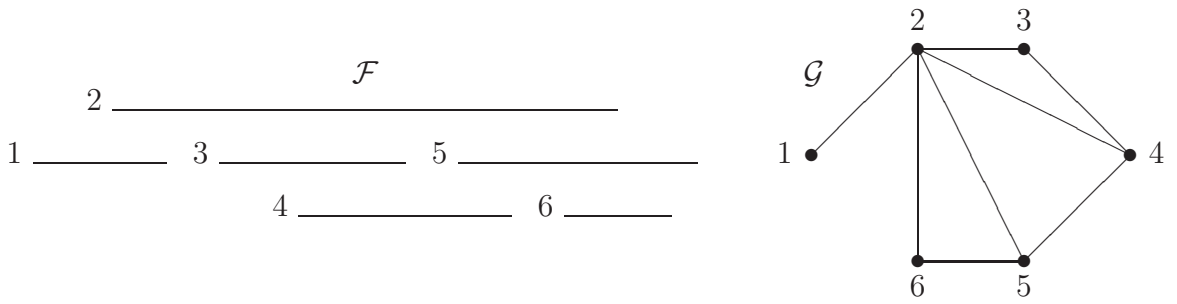
1. Chacun de ses sommets peut être mis en correspondance avec un intervalle de la droite réelle ;
2. Il existe une arête entre deux sommets si et seulement si les intervalles associés ont une partie commune. REMARQUE : Deux intervalles ont une partie commune s'ils se chevauchent ou si l'un contient l'autre.

Une **famille d'intervalles** est un ensemble d'intervalles de la droite réelle. Nous montrons un exemple de famille ci-après. Nous représentons chaque intervalle par un segment, le nom de l'intervalle est le numéro placé à gauche du segment. Pour des raisons de lisibilité, nous plaçons les intervalles les uns au dessus des autres et non pas sur une droite.

Définition 15 (Densité d'une famille d'intervalles) *La* **densité** *d'une famille d'intervalles est le nombre maximum d'intervalles couvrant un point de la droite réelle.*

Dans l'exemple suivant, la densité de la famille $\mathcal{F}$ est 3.

Exemple *Graphe d'intervalles*



On montre ici une famille d'intervalles $\mathcal{F}$ et le graphe d'intervalles $\mathcal{G}$ associé.

2.3.3 Graphes de chevauchement

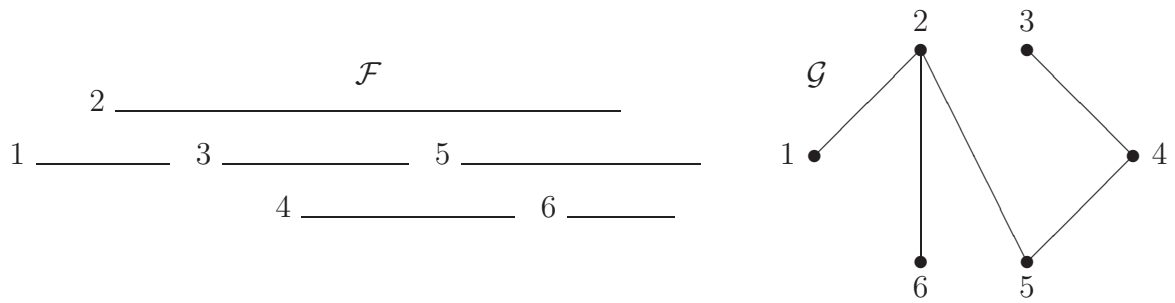
Définition 16 (Graphe de chevauchement) *Un graphe $\mathcal{G} = (V, E)$ est un graphe de chevauchement si et seulement s'il existe une bijection φ de V vers une famille I d'intervalles telle que :*

$$\{x, y\} \in E \Leftrightarrow \begin{cases} \varphi(x) \cap \varphi(y) \neq \varnothing \\ \text{non } (\varphi(x) \subseteq \varphi(y) \text{ ou } \varphi(y) \subseteq \varphi(x)) \end{cases}$$

Autrement dit si :

1. Chacun de ses sommets peut être mis en correspondance avec un intervalle de la droite réelle ;
2. Il existe une arête entre deux sommets si et seulement si les intervalles associés se chevauchent (c'est-à-dire s'ils ont une partie commune mais que l'un n'est pas inclus dans l'autre).

EXEMPLE *Graphe de chevauchement*



On reprend la famille $\mathcal{F}$ d'intervalles et on lui associe son graphe de chevauchement $\mathcal{G}$.

On remarque que les arêtes $\{2,3\}$, $\{2,4\}$ et $\{5,6\}$ ont disparues par rapport au graphe d'intervalles de la famille $\mathcal{F}$, en effet les intervalles 3 et 4 sont inclus dans l'intervalle 2 et l'intervalle 6 est inclus dans 5.

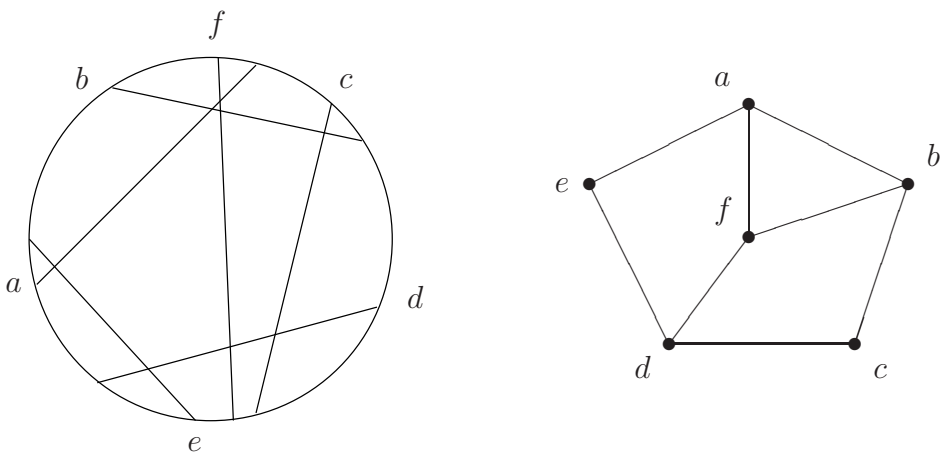
2.3.4 Graphes de cordes

Définition 17 (Graphe de cordes) *Un graphe $\mathcal{G} = (V, E)$ est un graphe de cordes*[5]*, si et seulement s'il existe une bijection φ de V vers un ensemble I de cordes d'un cercle telle que $\{x, y\} \in E \Leftrightarrow \varphi(x)$ et $\varphi(y)$ s'intersectent.*

Autrement dit si :

1. Chacun de ses sommets peut être mis en correspondance avec une corde d'un cercle ;
2. Il existe une arête entre deux sommets si et seulement si les cordes associées s'intersectent.

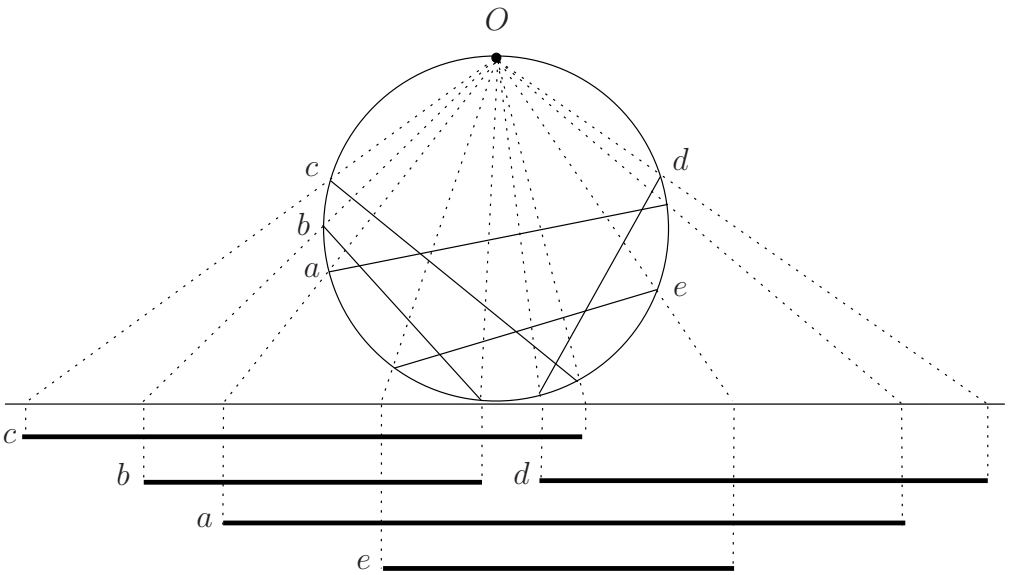
EXEMPLE *Graphe de cordes*



Cordes dans un cercle et graphe de cordes associé, exemple tiré de [Gavril, 1973].

[5] *Circle graph* en anglais.

Équivalence entre graphe de chevauchement et graphe de cordes Un résultat de [Gavril, 1973] indique qu'un graphe est un graphe de cordes si et seulement s'il est un graphe de chevauchement. La figure 2.6 montre cette équivalence. Ainsi tous les algorithmes définis sur les graphes de cordes sont directement applicables sur les graphes de chevauchement et inversement.



En plaçant un point O sur le cercle, on peut projeter les cordes sur la droite réelle pour former une famille d'intervalles. Le graphe ci-dessous est à la fois le graphe de cordes du cercle et le graphe de chevauchement de la famille d'intervalles obtenue.

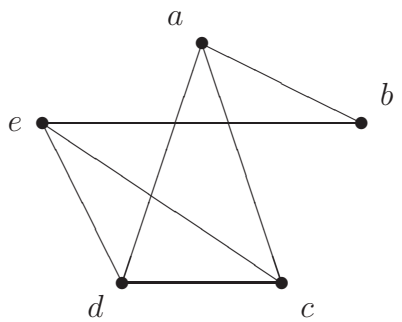


Fig. 2.6 – *Équivalence entre graphe de chevauchement et graphe de cordes.*

Première partie

Informatique

Chapitre 3

État de l'art

Sommaire

3.1 Problème du stable max 59
3.1.1 Stable max dans les graphes de chevauchement 60
3.1.2 Stable max dans les graphes de 2-intervalles 67
3.1.3 Stable max dans les graphes de croisement 69
3.2 Séquences répétées en tandem 72
3.2.1 Reconstruction de l'histoire des répétitions en tandem 72
3.2.2 Alignement de séquences contenant des répétitions en tandem . 74

Dans ce chapitre, nous faisons le point sur les deux principaux thèmes abordés pendant ce travail de thèse : les problèmes de stable max dans certaines classes de graphes, que nous utilisons dans notre méthode d'alignement, et les séquences répétées, qui sont les séquences que nous cherchons à aligner.

Comme nous l'avons vu au chapitre 2, le problème du stable max est de trouver dans un graphe $\mathcal{G} = (V, E)$, le plus grand ensemble de sommets $\mathcal{S} \subseteq V$ tel qu'il n'y ait pas d'arête entre les sommets de $\mathcal{S}$. $\mathcal{S}$ est appelé stable max de $\mathcal{G}$ et le cardinal de $\mathcal{S}$ est notée $\alpha(\mathcal{G})$.

Les problèmes ayant pour thème les séquences répétées sont la reconstruction de l'histoire de leurs duplications et leur alignement.

3.1 Problème du stable max

Le problème du stable max a été montré NP-complet pour les graphes quelconques, mais polynômial pour certaines classes de graphes comme par exemple les graphes bipartis, les graphes arêtes, les graphes triangulés et les graphes de cordes[1] [Garey et Johnson, 1979]. Nous nous intéressons plus particulièrement à trois classes de graphes : les graphes de

[1] *Circle graphs* en anglais.

chevauchement[2], les graphes de 2-intervalles et les graphes de croisement. Pour ces trois classes, le problème du stable max possède une solution en temps polynômial.

3.1.1 Stable max dans les graphes de chevauchement

Dans notre travail de thèse, pour aligner des séquences répétées en tandem sous le modèle d'évolution simple SSE, que nous présentons au chapitre 4, nous avons besoin de résoudre le problème suivant : trouver un stable max dans un graphe de chevauchement. La première solution au problème du stable max dans un graphe de chevauchement a été apportée par Gavril dans [Gavril, 1973] ; une amélioration de la complexité a été trouvée plus tard par Apostolico et ses collaborateurs [Apostolico et al., 1992].

[Gavril, 1973]

Dans cet article, F. Gavril décrit un algorithme en $O(n^3)$ pour résoudre le problème du stable max dans un graphe de cordes, où n est le nombre de sommets de ce graphe. Il montre qu'un graphe est un graphe de cordes si et seulement s'il est un graphe de chevauchement.

Sa solution considère la famille d'intervalles, notée $\mathcal{F}$, associée au graphe de chevauchement, noté $\mathcal{G} = (V, E)$. Pour chaque sommet $v \in V$, on note $\bar{v}$ l'intervalle correspondant dans la famille $\mathcal{F}$. À chaque sommet v de $\mathcal{G}$, Gavril associe :

- un ensemble U_v contenant les sommets associés aux intervalles inclus dans $\bar{v}$, c'est-à-dire $U_v = \{u \mid \bar{u} \subset \bar{v}\}$;
- et un graphe $\mathcal{G}_v$ qui est le sous-graphe de $\mathcal{G}$ induit par U_v.

Il construit le graphe d'intervalles $\tilde{\mathcal{G}}$ de la famille $\mathcal{F}$ dans lequel chaque sommet v reçoit un poids $\omega(v) = \alpha(\mathcal{G}_v) + 1$. C'est-à-dire que chaque sommet v reçoit un poids égal au cardinal d'un stable max des sommets de U_v plus 1 (lui-même). Ceci est illustré par un exemple à la figure 3.1.

L'idée de rechercher un stable max dans $\mathcal{G}_v$ est de résoudre le problème pour un sous-intervalle de la droite réelle. Les sommets v sont traités par étape de manière à ce que dans une même étape les intervalles correspondants ne soient pas inclus les uns dans les autres. Ainsi, à l'étape suivante, on pourra combiner les stables max des intervalles ne se chevauchant pas ou ne s'incluant pas. Ces relations sont codées dans les graphes d'intervalles. En associant $w(v)$ à chaque nœud v dans ce graphe, rechercher le stable de poids maximal revient à trouver l'union maximale des stables max correspondant à des sous-intervalles disjoints, stables max que l'on peut donc rassembler à l'étape i pour former un stable max à l'étape $i + 1$.

On note $\{v_1, \ldots, v_r\}$ un stable de poids maximal de $\tilde{\mathcal{G}}$ et D_{v_j} un stable max de $\mathcal{G}_{v_j}$, on a donc $D_{v_j} \subseteq U_{v_j}$. Gavril montre les deux propositions suivantes :

1. $\alpha(\mathcal{G}) = \alpha_\omega(\tilde{\mathcal{G}})$;
2. $\bigcup_{j=1}^{r}(v_j \cup D_{v_j})$ est un stable max de $\mathcal{G}$.

[2] *Overlap graph.*

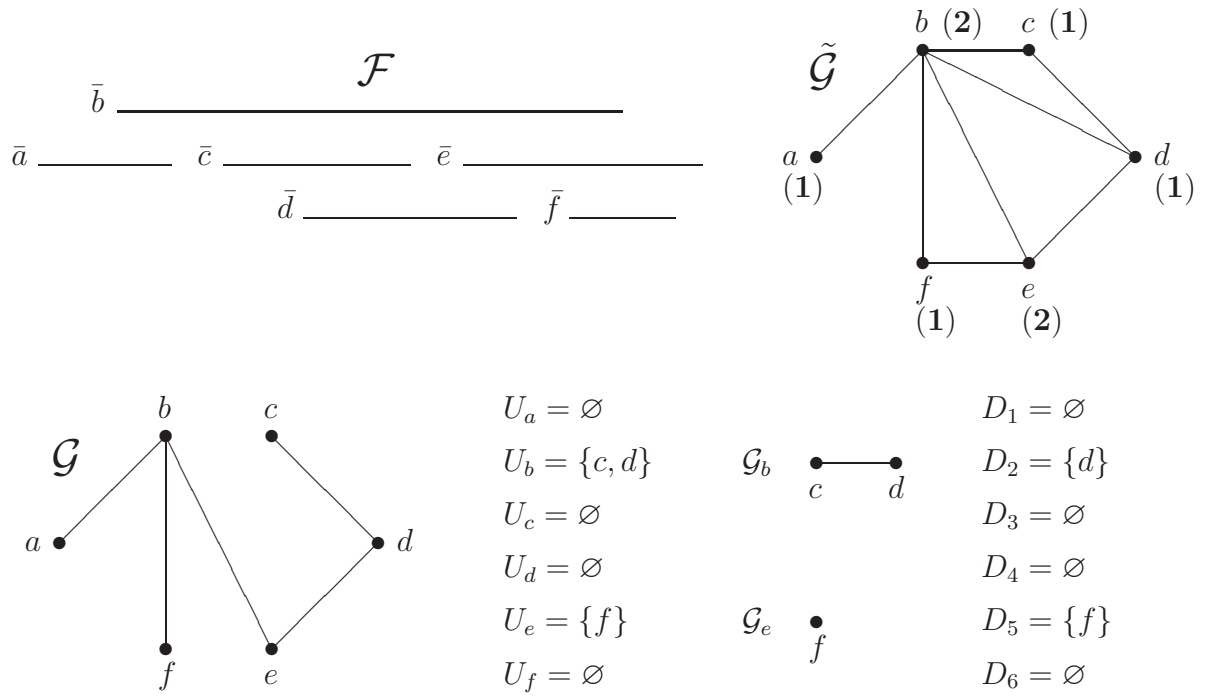


FIG. 3.1 – *Exemple d'application de l'algorithme de Gavril. Nous montrons ici une famille d'intervalles $\mathcal{F}$ et les graphes d'intervalles et de chevauchement associés, $\tilde{\mathcal{G}}$ et $\mathcal{G}$ respectivement. Nous donnons les ensembles U_v, les sous-graphes induits $\mathcal{G}_v$ non vides et les ensembles D_v comme décrits dans l'algorithme de Gavril. Les chiffres gras entre parenthèses dans le graphe d'intervalles $\tilde{\mathcal{G}}$ représentent les poids associés à chaque sommet.*

La première proposition signifie que le cardinal d'un stable max du graphe de chevauchement $\mathcal{G}$ est égal au poids du stable de poids maximal du graphe d'intervalles associé $\tilde{\mathcal{G}}$ pondéré par ω. Pour expliquer la deuxième proposition, on pose $B_j = v_j \cup D_{v_j}, \forall j \in [1..r]$; c'est-à-dire que B_j est l'association d'un sommet v_j et d'un stable max dans le sous-graphe de $\mathcal{G}$ induit par U_{v_j}. L'union de tous les B_j pour $j \in [1..r]$ est, d'après la deuxième proposition, un stable max de $\mathcal{G}$.

Dans l'exemple de la figure 3.1, le poids du stable de poids maximal de $\tilde{\mathcal{G}}$ est 4, l'ensemble $\{a, c, e\}$ est un stable de poids maximal de $\tilde{\mathcal{G}}$. On a $B_a = \{a\}$, $B_c = \{c\}$ et $B_e = \{e, f\}$. D'après la proposition 2, $B_a \cup B_c \cup B_e = \{a, c, e, f\}$ est un stable max de $\mathcal{G}$.

Pour affecter un poids $\omega(v)$ et un ensemble D_v à chaque sommet v de manière à ce que $\omega(v) = \alpha(\mathcal{G}_v) + 1$ et D_v est soit stable max de $\mathcal{G}_v$, Gavril considère les intervalles par niveau d'inclusion. Il commence par les intervalles qui n'en contiennent aucun autre (pour les sommets associés, $\omega(v) = 1$ et $D_v = \varnothing$), puis les intervalles qui contiennent seulement des intervalles déjà traités et ainsi de suite. Ainsi les poids sont affectés progressivement aux sommets de $\tilde{\mathcal{G}}$. On remarque que si deux intervalles sont traités à la même phase, ils

ne sont pas inclus l'un dans l'autre et donc soit ils se chevauchent, soit ils sont disjoints.

Dans l'exemple de la figure 3.1, on traite dans une première phase les intervalles $\bar{a}$, $\bar{c}$, $\bar{d}$ et $\bar{f}$, on traite ensuite dans une seconde phase $\bar{b}$ et $\bar{e}$.

Pour trouver un stable max dans un graphe de chevauchement $\mathcal{G}_v$, et ainsi affecter l'ensemble D_v, Gavril calcule un stable de poids maximal dans le graphe d'intervalles associé $\tilde{\mathcal{G}}_v$ et en déduit, par la proposition 2, un stable max de $\mathcal{G}_v$. En effet, trouver un stable de poids maximal dans un graphe d'intervalles est plus rapide que trouver un stable max dans un graphe de chevauchement. À chaque phase P, on calcule pour chaque sommet $v \in P$, un stable de poids maximal de $\tilde{\mathcal{G}}_v$, que l'on note $\{u_1, \ldots, u_t\}$; tous les sommets de $\tilde{\mathcal{G}}_v$ ont été traités lors de phases précédentes. D'après la proposition 1, on a $\alpha(\mathcal{G}_v) = \alpha_\omega(\tilde{\mathcal{G}}_v)$. On affecte donc $\omega(v)$ avec $\alpha_\omega(\tilde{\mathcal{G}}_v)+1$ et D_v avec $\bigcup_{j=1}^{t}(u_j \cup D_{u_j})$. Il assigne progressivement les poids ainsi obtenus aux sommets de $\tilde{\mathcal{G}}$. En trouvant un stable de poids maximal de $\tilde{\mathcal{G}}$ totalement valué, il en déduit un stable max de $\mathcal{G}$.

En 1973, trouver un stable de poids maximal dans le graphe d'intervalles $\tilde{\mathcal{G}}_v$ se faisait en $O(n^2)$, où n est le nombre de sommets de $\tilde{\mathcal{G}}_v$. Gavril cherche un stable de poids maximal pour chacun des n sommets de $\mathcal{G}$, donc la complexité de la procédure décrite ci-dessus est en $O(n^3)$.

La première solution en complexité linéaire pour résoudre le problème du stable de poids maximal dans un graphe d'intervalles a été proposée par A. Franck [Frank, 1976] en 1976, soit trois ans après l'article de Gavril précédemment cité [Erlebach et Spieksma, 2002]. Aussi, l'adaptation de l'algorithme de Gavril avec la méthode de Franck pour trouver un stable de poids maximal dans les $\tilde{\mathcal{G}}_v$ conduit à une complexité en $O(n^2)$.

[Apostolico et al., 1992]

En 1992, Apostolico et ses collaborateurs [Apostolico et al., 1992] donnent deux procédures pour résoudre le problème du stable max dans un graphe de chevauchement. La première est en $O(min\{n^2, d^2n\})$, où d est la densité de la famille d'intervalles associée (cf. définition 15, page 54). La seconde en $O(nlogn + dn)$ utilise des structures de données spécifiques. La première solution permet de stocker les stables max de tous les sous-graphes induits et nous intéresse plus particulièrement ; nous allons donc la détailler ici. Dans un premier temps, nous donnons les notations utilisées, nous décrivons ensuite l'algorithme et nous concluons en discutant de la complexité.

Soit $\mathcal{F}$ une famille de n intervalles sur la droite réelle, telle que deux intervalles ne partagent pas la même extrémité. Un intervalle i est représenté par la paire ordonnée (g_i, d_i) de ses extrémités sur la droite réelle, $g_i < d_i$. La contrainte sur la famille $\mathcal{F}$ peut alors s'écrire :

$$\forall i, j \in \mathcal{F}, i \neq j \Rightarrow (g_i \neq g_j \text{ ET } g_i \neq d_j \text{ ET } d_i \neq g_j \text{ ET } d_i \neq d_j).$$

Cette condition est nécessaire à l'application de l'algorithme de Apostolico et ses collaborateurs. Remarque : On peut toujours artificiellement modifier les extrémités des intervalles pour que $\mathcal{F}$ respecte cette condition

Le poids d'un intervalle i est noté ω_i, $\forall i \in [1 \dots n]$. Il est toujours possible, par un tri en $O(nlogn)$, d'obtenir les intervalles numérotés de 1 à n selon l'ordre naturel de leur extrémité gauche, c'est-à-dire triés tels que $i < j$ si et seulement si $g_i < g_j$.

Soient i, j deux intervalles tels que $i < j$, c'est-à-dire $g_i < g_j$;

– l'intervalle i *contient* l'intervalle j si $d_j < d_i$;

– les intervalles i et j sont *disjoints* si $d_i < g_j$;

– enfin, les intervalles i et j *se chevauchent* s'ils ne sont ni disjoints, ni que l'un d'eux est contenu dans l'autre.

La famille $\mathcal{F}$ peut être associée à un graphe de chevauchement $\mathcal{G} = (V, E)$ tel que $|V| = n$ et $(i, j) \in E$ si et seulement si $i \cap j \neq \varnothing$ et ($i \nsubseteq j$ et $j \nsubseteq i$), c'est-à-dire si i chevauchement proprement j.

En temps $O(nlogn)$, la famille d'intervalles $\mathcal{F}$ peut être représentée comme une chaîne $\alpha = \alpha_1\alpha_2 \dots \alpha_{2n}$ qui est appelée *codage* α de $\mathcal{F}$. La chaîne α est une permutation de $\{1, 1, 2, 2, 3, 3, \dots, n, n\}$. Deux occurrences de i dans α représentent les extrémités de l'intervalle i. Notons que la première occurrence de i dans α précède la première occurrence de $i + 1$. Par cette transformation, les extrémités de chaque intervalle sont codées par des entiers. On garde la notation (g_i, d_i) pour désigner les extrémités de l'intervalle i. On note E_i l'ensemble des sommets associés aux intervalles inclus dans i plus i lui-même. Avec les notations de l'article précédent, on a $E_i = U_i \cup \{i\}$.

Remarque : Nous appelons stable max d'une famille d'intervalles $\mathcal{F}$ associée à un graphe de chevauchement $\mathcal{G}$, un ensemble d'intervalles de $\mathcal{F}$ associés aux sommets d'un stable max de $\mathcal{G}$. Un stable max d'une famille d'intervalles est donc un plus grand ensemble d'intervalles deux à deux compatibles au sens de la relation de non chevauchement, c'est-à-dire disjoints ou contenus l'un dans l'autre. Dans l'exemple 3.1 de la section précédente, un stable max de $\mathcal{G}$, $\{a, c, e, f\}$ peut également être vu sur la famille d'intervalles comme le plus grand ensemble d'intervalles deux à deux compatibles, c'est-à-dire $\{\bar{a}, \bar{c}, \bar{e}, \bar{f}\}$.

Dans la suite, par abus de langage on confondra le sommet du graphe et l'intervalle associé dans la famille.

Comme dans l'algorithme de Gavril, pour tout intervalle v de la famille on calcule le stable max de $\mathcal{G}_v$. Supposons qu'un intervalle f couvre toute la famille $\mathcal{F}$, alors calculer le stable max de $\mathcal{F}$ revient à calculer celui du graphe induit $\mathcal{G}_f$. L'idée du premier algorithme de [Apostolico et al., 1992] est que le calcul du stable max sur un intervalle v peut se faire en lisant le codage α de gauche à droite. En effet, lorsque l'on parvient à l'extrémité droite de v, on connaît tous les intervalles inclus dans v. La procédure nous donne donc le poids de ce stable et l'on peut alors calculer, pour tout u tel que $v \subset u$, le stable max compris entre l'extrémité gauche de u et l'extrémité droite de v.

Le premier algorithme de [Apostolico et al., 1992] est donné ci-dessous. Il associe à chaque intervalle i un poids stocké dans $CMIS[i]$, dont la valeur finale correspond au cardinal d'un stable max du sous-graphe induit par E_i. L'ensemble $OPEN$ contient les intervalles *ouverts*, c'est-à-dire les intervalles dont on a rencontré l'extrémité gauche, mais pas encore l'extrémité droite.

À chaque intervalle i correspond une entrée dans $CMIS$ et une entrée dans *clist* :

– $CMIS[i]$ contient le cardinal d'un stable max parmi les intervalles contenus dans i

et dont on a déjà rencontré l'extrémité droite. Quand on rencontre l'extrémité droite de i, l'ajout de 1 dans $CMIS[i]$ (ligne **7**) permet de prendre en compte i et d'obtenir le cardinal d'un stable max dans le sous-graphe induit par E_i ;
– $clist[i]$ contient des paires composées des extrémités droites des intervalles contenus dans i et d'un poids. Le poids d'une paire (d_j, w) correspond au cardinal d'un stable max des intervalles compris entre g_i et d_j.

Algorithme 1: *Calcul du cardinal d'un stable max d'un graphe de chevauchement [Apostolico et al., 1992].*

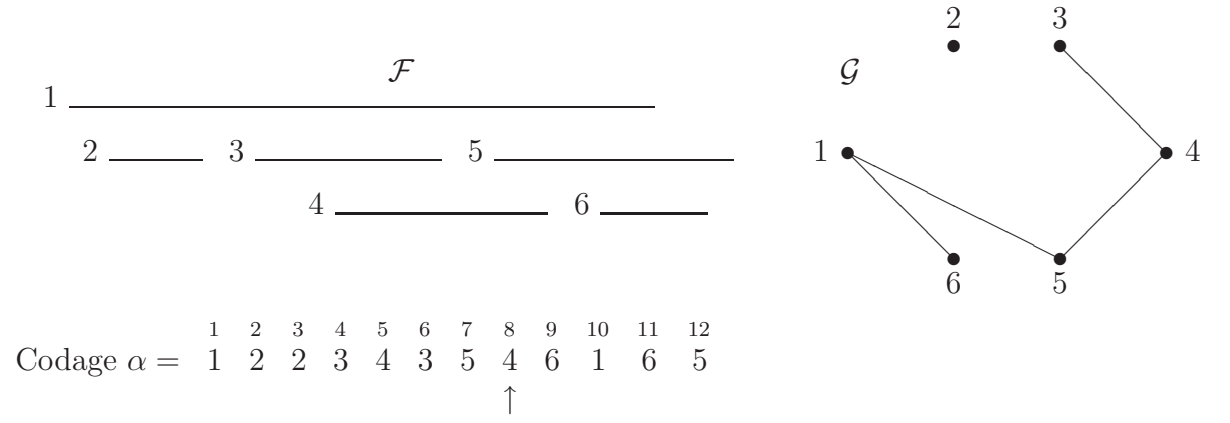
Données : Un codage α de dimension $2n$, où n est le nombre d'intervalles.
Résultat : Le cardinal d'un stable max parmi les n intervalles.

```
   /* 1ère phase : déterminer les entrées de CMIS */
 1 pour p = 1 à 2n faire
 2     si p est le début d'un intervalle i (p = g_i) alors
 3         Ajouter i dans OPEN ;
 4         CMIS[i] = 0 ;
 5         clist[i] = ∅ ;
       sinon
           /* p correspond à la fin d'un intervalle i (p = d_i) */
 6         Retirer i de OPEN ;
 7         CMIS[i] = CMIS[i] + 1 ;
           /* mise à jour des intervalles de OPEN */
 8         pour chaque x ∈ OPEN tel que g_x < g_i faire
 9             Choisir dans clist[x] la paire (d, w) telle que d < g_i et w maximum ;
10             si une telle paire n'existe pas alors w ← 0 ;
11             clist[x] ← (d_i, w) ;
12             CMIS[x] = max(CMIS[x], CMIS[i] + w) ;
13             poids du dernier élément de clist[x] ← CMIS[x] ;

   /* 2ème phase : déterminer le cardinal d'un stable max */
14 MIS[0] = 0 ;
15 pour p = 1 à 2n faire
16     si p est le début d'un intervalle i alors
17         MIS[p] = MIS[p − 1] ;
       sinon
           /* p correspond à la fin d'un intervalle i */
18         MIS[p] = max(MIS[p − 1], MIS[g_i − 1] + CMIS[i]) ;

19 Retourner MIS[2n] ;
```

Le cœur de l'algorithme est dans la mise à jour des intervalles contenus dans $OPEN$



$OPEN = \{1\ 2\ 3\ 4\ 5\}$

$CMIS$			
1	0	1	2
2	0	1	
3	0	1	
4	0	1	
5	0		
6			

clist				
1	$\varnothing$	$(3, 0\ 1)$	$(6, 1\ 2)$	$(8, 1\ 2)$
2	$\varnothing$			
3	$\varnothing$			
4	$\varnothing$			
5	$\varnothing$			
6				

FIG. 3.2 – *Nous montrons ici une étape de l'algorithme de Apostolico. Nous représentons une famille d'intervalles $\mathcal{F}$ et son codage α associé, avec en dessous l'ensemble $OPEN$ et les tables $CMIS$ et clist à l'état succédant l'examen de la position 8 du codage α (indiquée par la flèche verticale). Les chiffres grisés correspondent aux valeurs qui ont été remplacées ou effacées au cours de l'algorithme.*

(lignes **8-13**). L'algorithme se déroule en deux phases. Dans la première, on calcule les entrées de $CMIS$ en une seule passe sur le codage α (de gauche à droite), dans la seconde, on détermine le cardinal d'un stable max de la famille.

Dans la figure 3.2, nous montrons un exemple de déroulement de l'algorithme 1. La famille $\mathcal{F}$, légèrement différente de celle de l'exemple précédent, contient six intervalles numérotés de 1 à 6 ; chaque intervalle est associé à un sommet du graphe de chevauchement $\mathcal{G}$. Nous montrons dans cet exemple l'état de l'ensemble $OPEN$ et des tables $CMIS$ et *clist* à la fin de l'examen de la position 8 du codage α. L'ensemble $OPEN$ contient les intervalles 1 et 5 dont l'extrémité gauche se situe avant la position 8 et l'extrémité droite après. $CMIS[2]$, $CMIS[3]$ et $CMIS[4]$ contiennent leurs valeurs finales, c'est-à-dire le cardinal des stables max des sous-graphes induits $\mathcal{G}_{E_2}$, $\mathcal{G}_{E_3}$ et $\mathcal{G}_{E_4}$ respectivement[3]. La valeur de $CMIS[1]$ est mise à jour trois fois jusqu'à l'examen de la position 8 :

1. À la fin de l'intervalle 2 (position 3 du codage α), $CMIS[2] \leftarrow 1$; l'intervalle 1 est dans $OPEN$ et $g_1 < g_2$, comme $clist[1]$ est vide, $w \leftarrow 0$ et la paire $(2, 0)$ est ajoutée

[3]On remarque ici qu'étant donné que $\mathcal{G}_{E_2}$, $\mathcal{G}_{E_3}$ et $\mathcal{G}_{E_4}$ ne contiennent qu'un seul sommet, le cardinal de leur stable max est 1.

à $clist[1]$. $CMIS[1]$ est mis à jour et reçoit la valeur $\max(0,1) = 1$ et le poids de la dernière paire de $clist[1]$ est remplacée par 1 ;

2. À la fin de l'intervalle 3 (position 6 du codage α), $CMIS[3] \leftarrow 1$, de la même manière que précédemment, la valeur de $CMIS[1]$ est mise à jour, à la différence, que maintenant il existe dans $clist[1]$ une paire (d, w) telle que $d < g_3$, c'est la paire $(3,1)$. D'où $w \leftarrow 1$, $CMIS[1] \leftarrow 2$ et la paire $(6,1)$ est ajoutée à $clist[1]$ puis modifiée en $(6,2)$;
3. À la fin de l'intervalle 4 (position 8 du codage α), $CMIS[4] \leftarrow 1$, la paire de $clist[1]$ répondant à la condition $d < g_4$ est $(3,1)$, d'où $w \leftarrow 1$, $CMIS[1]$ reste inchangé et la paire $(8,1)$ est ajoutée à $clist[1]$ puis modifiée en $(8,2)$.

À la fin de l'intervalle 1, c'est-à-dire lors de l'examen de la position 10, $CMIS[1]$ est affecté avec la valeur 3 qui est le cardinal du stable max de $\mathcal{G}_{E_1}$ ($E_1 = \{1,2,3,4\}$). Une fois la première phase de l'algorithme terminé, on a :

$CMIS$	
1	3
2	1
3	1
4	1
5	2
6	1

	$clist$			
1	$\varnothing$	$(3,1)$	$(6,2)$	$(8,2)$
2	$\varnothing$			
3	$\varnothing$			
4	$\varnothing$			
5	$\varnothing$	$(11,1)$		
6	$\varnothing$			

Après la deuxième phase, la table MIS est remplie de la manière suivante :

MIS	0	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	0	1	1	1	2	2	2	2	3	3	4

Le cardinal d'un stable max du graphe se trouve dans la case $MIS[2n]$. Dans notre exemple, le cardinal d'un stable max de $\mathcal{G}$ est donc 4.

On remarque que pour tout i, $clist(i)$ contient les extrémités droites des intervalles inclus dans i. À partir de $clist(i)$ on peut trouver un stable max parmi les intervalles inclus dans i. Pour cela, il suffit de modifier légèrement l'algorithme pour permettre de garder en mémoire quelles sont les paires de $clist(i)$ impliquées dans le calcul du poids de ces paires. Ainsi, en partant d'une paire de poids maximal et en remontant jusqu'à une paire de poids 1, on trouve un stable max des intervalles inclus dans i, que nous notons D_i. Par exemple, pour $i = 1$, nous sélectionnons la paire $(8,2)$ de poids maximal poids, le poids de cette paire a été calculé à partir de $(3,1)$, de poids 1. Les intervalles correspondant à ces deux paires sont 2 et 4. Un stable max des intervalles inclus dans 1 est donc $D_1 = \{2,4\}$.

Pour retrouver les sommets appartenant au stable max de la famille, on modifie de la même manière l'algorithme pour permettre de garder en mémoire quels sont les sommets impliqués dans le calcul de la valeur finale de MIS. Notons les $v_1 \dots v_s$. Un stable max de la famille est alors :

$$\bigcup_{i=1}^{s} v_i \cup D_i,$$

où les D_i sont calculés à partir de $clist(i)$ comme expliqué ci-dessus.

Dans notre exemple, les valeurs de MIS contribuant à la valeur finale sont augmentées aux positions 3, 6 et 12 (la position 10 n'est pas comptée, car lorsque l'on affecte la position 12 avec la valeur 4, c'est le résultat $MIS[6]+2$). Cela correspond aux intervalles 2, 3 et 5. On a $D_2 = D_3 = \varnothing$ et $D_5 = \{6\}$, d'où un stable max de la famille $\mathcal{F}$ est $\{2, 3, 5, 6\}$.

L'algorithme 1 a une complexité en $O(d^2n)$, où n est le nombre de sommets du graphe et d la densité de la famille d'intervalles associée. En effet, la complexité de la première phase est en $O(d^2n)$: l'ensemble $OPEN$ des intervalles ouverts a au plus d éléments et la liste $clist$ a également au plus d éléments ; la deuxième phase est en $O(n)$. Dans la complexité $O(min\{n^2, d^2n\})$ annoncée par Apostolico, n^2 provient de l'adaptation de la solution de Gavril avec une méthode similaire à celle que l'on vient de décrire et d^2n de l'algorithme 1. Cette complexité ne prend pas en compte le coût $O(nlogn)$ du tri des intervalles.

3.1.2 Stable max dans les graphes de 2-intervalles

Lorsque nous avons étendu notre modèle d'évolution pour l'alignement de séquences répétées en tandem, nous nous sommes retrouvés à manipuler des objets qui sont en fait des 2-intervalles et non plus des intervalles simples (voir chapitre 6). Notre problème est devenu : trouver un stable max dans un graphe de 2-intervalles défini par une relation de compatibilité entre 2-intervalles spécifique à notre modèle. Cela nous a amené à nous intéresser aux travaux de Stéphane Vialette [Vialette, 2001] qui traitent de ce genre de problème. [Vialette, 2001] fait référence à un article qui a particulièrement retenu notre attention, [Felsner et al., 1997], où une nouvelle classe de graphes est introduite.

[Vialette, 2001]

Dans le cadre de la prédiction de structure secondaire d'ARN, Stéphane Vialette étudie des problèmes algorithmiques sur les graphes d'intersection d'un ensemble de 2-intervalles.

Définition 18 (2-intervalle) *Un 2-intervalle est un couple d'intervalles (I, J), tels que I et J ne se chevauchent pas.*

Un graphe de 2-intervalles est le graphe d'intersection d'une famille de 2-intervalles. Les graphes de 2-intervalles sont une généralisation des graphes d'intervalles. Dans sa thèse, Stéphane Vialette s'intéresse à différents problèmes de recherche de sous-familles de 2-intervalles comparables.

Soient deux 2-intervalles $D_1 = (I_1, I_1')$ et $D_2 = (I_2, I_2')$, ils sont *comparables* s'ils sont dans l'une des trois configurations de la figure 3.3 ou les configurations symétriques (I_1 à la place de I_2).

Un *modèle de comparaison* $\mathcal{R}$ permet de restreindre le nombre de ces configurations. Plus précisément, un modèle de comparaison est un sous-ensemble non vide de l'ensemble des relations $\{<, \sqsubset, \sqcap\!\!\sqsubset\}$ (décrites à la figure 3.3) tel que deux 2-intervalles sont $\mathcal{R}$-comparables s'ils sont comparables par l'une des relations de $\mathcal{R}$. S. Vialette a étudié deux problèmes sur les familles de 2-intervalles en relation avec les structures secondaires d'ARN :

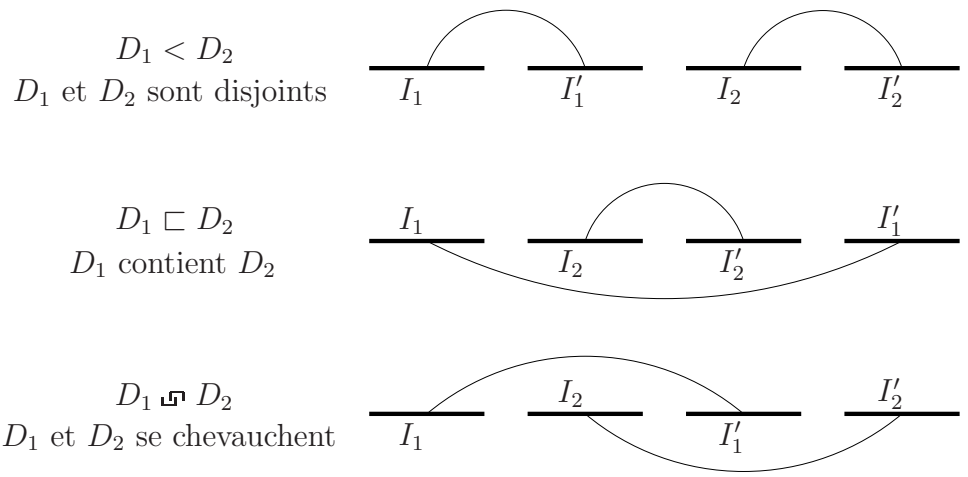


Fig. 3.3 – *Compatibilité entre deux 2-intervalles [Vialette, 2001].*

1. Calculer la plus grande sous-famille de 2-intervalles $\mathcal{R}$-comparables relativement à un modèle de comparaison $\mathcal{R}$ donné ;
2. Rechercher un motif dans une famille de 2-intervalles, c'est-à-dire l'extension du problème de recherche d'un motif dans un texte au cas des familles de 2-intervalles.

Nous nous intéressons à la première famille de problèmes lorsque $\mathcal{R}=\{<, \sqsubset\}$. Dans ce cas, Stéphane Vialette donne une solution en $O(n^2)$ (voir [Vialette, 2001, chapitre 6, pages 180-3]). Pour cela, il utilise les graphes de croisement de doubles intervalles définis par [Felsner et al., 1997].

Définition 19 (Double intervalle) *Un double intervalle est une paire (I_1, I_2) d'intervalles sur la droite réelle, où I_2 est inclus dans I_1.*

Felsner et ses collaborateurs montrent le théorème suivant :

Théorème 1 *Le calcul d'un stable max dans un graphe de croisement est soluble en temps $O(n^2)$, où n est le nombre de doubles intervalles.*

Notons g_I l'extrémité gauche de l'intervalle I et d_I son extrémité droite. Soit $D = (I, J)$ un 2-intervalle, l'intervalle extérieur de D, noté $ext(D)$, est défini par $ext(D) = [g_I, d_J]$ et l'intervalle intérieur de D, noté $int(D)$, est défini par $int(D) =]d_I, g_J[$. Pour tout 2-intervalle D, $int(D) \subset ext(D)$. Autrement dit, le couple $(ext(D), int(D))$ est un double intervalle. Étant donné une famille $\mathcal{F} = \{D_i \mid 1 \leq i \leq n\}$ de 2-intervalles, la famille $\tilde{\mathcal{F}}$ est définie par $\tilde{\mathcal{F}} = \{\tilde{D} \mid D \in \mathcal{F}\}$ avec $\tilde{D} = (ext(D), int(D))$. $\tilde{\mathcal{F}}$ est une famille de doubles intervalles. Stéphane Vialette montre alors le lemme suivant :

Lemme 1 *Soit $\mathcal{F} = \{D_i \mid 1 \leq i \leq n\}$ une famille de 2-intervalles. Il existe une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$, avec $|\mathcal{F}'| = K$, de 2-intervalles $\{<, \sqsubset\}$-comparables si et seulement s'il existe un stable de taille K dans le graphe de croisement de $\tilde{\mathcal{F}}$.*

Le théorème 1 et le lemme 1 lui permettent de conclure quant à la complexité du problème.

Bien que nous ayons la même définition des 2-intervalles, notre relation de compatibilité autorise des intervalles de 2-intervalles différents à se chevaucher, alors que pour cette configuration, le modèle proposé par Stéphane Vialette les rend incomparables. Cette différence dans la relation de compatibilité nous empêche d'utiliser la méthode de Stéphane Vialette pour résoudre notre problème (voir chapitre 6, section 6.5.1.1).

3.1.3 Stable max dans les graphes de croisement

Dans le cadre d'une recherche d'un modèle de compatibilité entre 2-intervalles qui corresponde à notre problème, nous nous sommes intéressés à l'article de Felsner et ses collaborateurs.

[Felsner et al., 1997]

Felsner et ses collaborateurs introduisent une nouvelle classe de graphes, les graphes de trapézoïdes circulaires. Cette classe contient comme sous-classe, les graphes de trapézoïdes, les graphes de cordes et les graphes d'arcs circulaires. Ils montrent que le problème du stable max sur cette classe de graphes est encore polynômial.

Un trapézoïde circulaire est une région du cercle qui est comprise entre deux cordes non sécantes et un graphe de trapézoïdes circulaires est le graphe d'intersection d'une famille de trapézoïdes circulaires sur un cercle. Ceci est illustré à la figure 3.4.

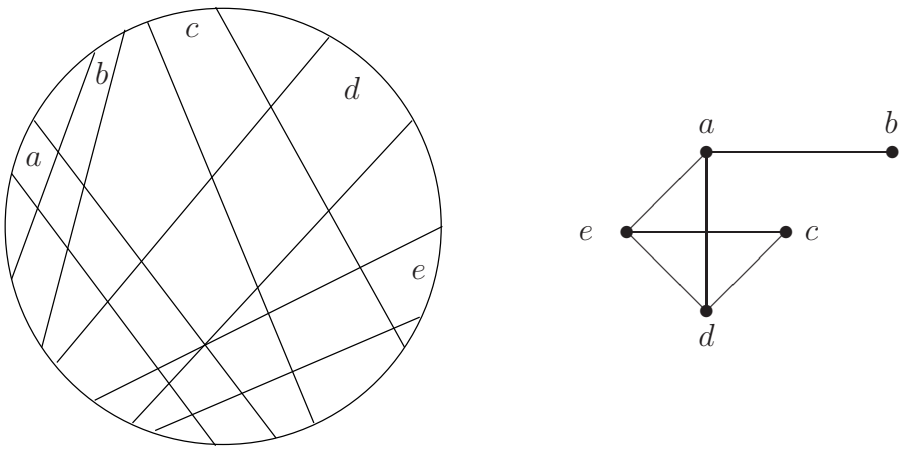


FIG. 3.4 – *À gauche un cercle contenant* 5 *trapézoïdes circulaires : a, b, c, d et e. À droite le graphe de trapézoïdes circulaires associé.*

Pour résoudre le problème du stable max dans les graphes de trapézoïdes circulaires, ils utilisent des graphes de croisement qu'ils définissent sur des doubles intervalles (cf. définition 19).

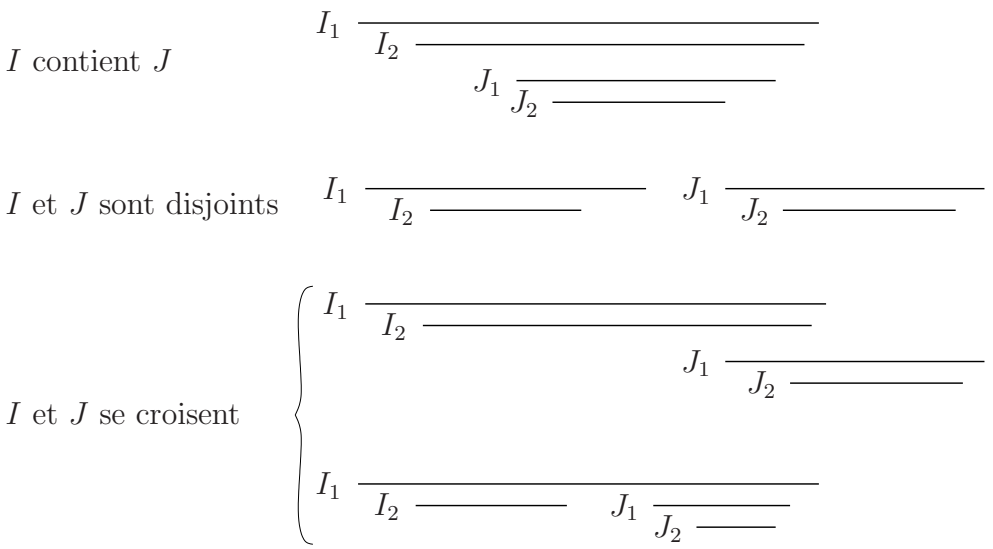


FIG. 3.5 – *Compatibilité entre deux doubles intervalles, $I = (I_1, I_2)$ et $J = (J_1, J_2)$.*

Soient $I = (I_1, I_2)$ et $J = (J_1, J_2)$ deux doubles intervalles :
- I *contient* J si $J_1 \subset I_2$;
- I et J sont *disjoints* si $I_1 \cap J_1 = \varnothing$;
- I et J *se croisent* s'ils ne sont ni disjoints, ni que l'un d'eux contient l'autre.

Ces relations sont illustrées à la figure 3.5.

Définition 20 (Graphe de croisement) *Un graphe $\mathcal{G} = (V, E)$ est un graphe de croisement si ses sommets peuvent être mis en correspondance un à un avec un ensemble de doubles intervalles tel que deux sommets de $\mathcal{G}$ sont adjacents si et seulement si leurs doubles intervalles associés se croisent.*

La classe des graphes de croisement contient à la fois les graphes de trapézoïdes et les graphes de chevauchement.

Un trapézoïde circulaire $\mathcal{T}$ peut être représenté par un double intervalle, il suffit en effet de définir un point sur le cercle, ce point est nommé O, et un sens de rotation. À partir de O, nous parcourons le cercle dans le sens choisi, nous rencontrons donc les quatre sommets du trapézoïde $\mathcal{T}$, le premier et le quatrième délimitent un intervalle, que l'on nomme I_1, le deuxième et le troisième, un intervalle I_2. Par construction, $I_1 \subset I_2$. (I_1, I_2) est donc bien un double intervalle qui représente $\mathcal{T}$.

Les auteurs donnent un algorithme en $O(n^2)$, où $n = |V|$, pour trouver un stable max dans un graphe de croisement $\mathcal{G} = (V, E)$ induit par une famille $\mathcal{F}$ de doubles intervalles. Le processus est le suivant :

1. D'abord on extrait l'*ordre de contenance* $\mathcal{P}_C = (V, P_C)$ et l'*ordre de précédence* $\mathcal{P}_I = (V, P_I)$ correspondant à $\mathcal{F}$. Les ordres $\mathcal{P}_C$ et $\mathcal{P}_I$ sont des graphes orientés définis sur l'ensemble de sommets V :

  - dans $\mathcal{P}_C$, il existe un arc d'un sommet x vers un sommet y si et seulement si le double intervalle correspondant à x est contenu dans le double intervalle correspondant à y,
  - dans $\mathcal{P}_I$, il existe un arc de x vers y si et seulement si le double intervalle correspondant à x est strictement à gauche du double intervalle correspondant à y ;
2. On calcule ensuite l'extension linéaire de $\mathcal{P}_C$, $L_C = v_1, \ldots, v_n$. L'extension L_C est une manière d'ordonner les sommets à examiner, assurant que lorsqu'on examine un sommet associé à un double intervalle D, tous les sommets associés aux doubles intervalles contenus dans D ont déjà été examinés. C'est-à-dire que si v_i contient v_j alors $j < i$ dans L_C ;
3. Enfin, tous les sommets $v \in V$ sont affectés d'un poids (égal à 1 si on cherche le stable max, c'est-à-dire $w(v) = 1$) et on prolonge artificiellement $\mathcal{P}_C$ et L_C par un élément v_{n+1} de poids 0, tel que $v_{n+1} > v_i \; \forall i \in [1, n]$.

Ce prétraitement peut être accompli en $O(n^2)$. L'algorithme 2, MAXINDSET-CROSS, donne le principe du calcul d'un stable max. On remarque que U_i contient seulement des éléments v_j tels que $j < i$ dans $\mathcal{P}_C$. Donc les poids $W(v_j)$ de tous les éléments de U_i ont été calculés avant la $i^{\text{ème}}$ boucle. On peut prouver facilement l'invariant suivant à la fin de chaque boucle :

Pour tout $j \leq i$, le poids $W(v_j)$ est le poids d'un stable max, noté $I(v_j)$, des éléments $v \in \{v_j\} \cup U_j$, c'est-à-dire des éléments tels que $v \leq v_j$ dans $\mathcal{P}_C$.

Cet invariant implique directement que $I(v_{n+1})$ est un stable max de $\mathcal{G}$.

Algorithme 2: *MAXINDSET-CROSS [Felsner et al., 1997].*

Données : $\mathcal{P}_C$, $\mathcal{P}_I$ et L_C d'un graphe de croisement $\mathcal{G} = (V, E)$.
Résultat : Un stable max du graphe $\mathcal{G}$.
pour $i = 1$ *à* $n + 1$ **faire**
1 $\quad U_i \leftarrow \{v_j \mid v_j < v_i \text{ dans } \mathcal{P}_C\}$;
2 $\quad C \leftarrow$ chaîne de poids maximal dans $\mathcal{P}_I$ des éléments de U_i pondérés par W ;
3 $\quad W(v_i) \leftarrow w(v_i) + \sum_{v \in C} W(v)$;
4 $\quad I(v_i) \leftarrow \{v_i\} \cup \bigcup_{v \in C} I(v)$;
5 Retourner $I(v_{n+1})$;

L'algorithme 2 calcule un stable max d'un graphe de croisement $\mathcal{G} = (V, E)$, $|V| = n$, en $O(n^2)$ si la procédure de recherche de chaîne de poids maximal (ligne **2**) est accomplie en $O(n)$. Une chaîne dans un graphe orienté est une suite de sommets reliés par des arcs. Le poids de la chaîne est la somme des poids des sommets qui la composent.

Ce modèle des doubles intervalles est adaptable à nos données. Par contre, encore une fois, c'est notre relation de compatibilité qui n'est pas adaptable à ce modèle (voir chapitre 6, section 6.5.1.2).

* *

*

Nous avons présenté dans la première partie de cet état de l'art les différentes classes de graphes auxquelles nous nous sommes intéressés pour résoudre le problème du stable max. Pour chacune de ces classes, nous avons donné les méthodes connues de calcul d'un stable max. Nous présentons maintenant l'état de l'art sur l'étude des répétitions en tandem.

3.2 Séquences répétées en tandem

L'étude des séquences répétées en tandem est le thème central de ce travail de thèse. Le tout premier intérêt que nous avons porté à ces séquences provient du problème de la reconstruction de l'histoire de leurs duplications [Bérard, 2000]. Nous nous sommes basés pour cela sur l'article de Benson et Dong [Benson et Dong, 1999].

Nous nous sommes ensuite intéressés au problème de l'alignement de séquences répétées en tandem. Ce problème a été abordé d'une manière différente de celle que nous avons adoptée dans [Benson, 1997]. Enfin, nous avons spécialisé notre travail sur l'alignement de cartes de minisatellite, pour lequel nous avons produit un algorithme [Bérard et Rivals, 2002], [Bérard et Rivals, 2003]. Un article ultérieur au nôtre résout notre problème avec une complexité différente [Behzadi et Steyaert, 2003]. Cet article est présenté au chapitre 6, page 130.

3.2.1 Reconstruction de l'histoire des répétitions en tandem

Le problème est le suivant : pour une séquence de répétitions en tandem donnée, trouver l'histoire de sa construction qui contient le moins d'opérations d'édition (insertions, délétions ou substitutions d'un caractère) et d'amplifications d'une sous-chaîne. L'opération d'amplification d'une sous-chaîne a la même définition que l'opération d'amplification d'un caractère définie page 38, cette opération est également appelée duplication en tandem.

[Benson et Dong, 1999]

Dans un article de 1999, Benson et Dong proposent plusieurs formalisations plus ou moins restrictives du problème de reconstruction de l'histoire des répétitions en tandem et étudient des approches heuristiques [Benson et Dong, 1999]. Ils supposent que la séquence est constituée de n copies approximatives d'un motif de base de longueur k et ils alignent les n copies dans une matrice de n lignes et k colonnes. Ci-dessous un exemple de représentation :

Répétition en tandem : $CGA\ CGG\ CG\ CGG\ CGG\ CGA$

Matrice correspondante :
$$\begin{bmatrix} C & G & A \\ C & G & G \\ C & - & G \\ C & G & G \\ C & G & G \\ C & G & A \end{bmatrix}$$

Pour retrouver l'histoire de la séquence, ils appliquent à cette matrice une opération algorithmique de *contraction* qui remplace plusieurs copies par une seule, la *copie consensus*, jusqu'à ce que la matrice n'ait plus qu'une seule ligne. Ils donnent des règles pour produire cette copie consensus. Chaque contraction admet un coût déterminé par une *fonction de coût*. Une série de contractions qui réduit la matrice de départ à une seule ligne représente une histoire possible pour la séquence. Le motif restant dans la matrice est potentiellement le motif ancêtre. Le coût total d'une telle histoire est la somme des coûts des contractions qui la composent.

Les auteurs n'étudient pas la complexité, ni ne donnent un algorithme exact du problème général. D'autres travaux montrent que le nombre d'histoires possibles pour une séquence donnée est exponentiel [Gascuel et al., 2003] et que le problème de trouver l'histoire de coût optimal est NP-complet [Jaitly et al., 2002]. Il y a plusieurs manières de restreindre ce problème, les restrictions abordées dans l'article de Benson et Dong portent principalement sur :

- le nombre de copies contractées en un événement de contraction, qui est fixé à 2 ou variable ;
- la taille des copies à contracter, qui est soit de même longueur que le motif de base, c'est-à-dire la largeur de la matrice, soit un multiple de cette longueur ;
- le début de la première copie à contracter, qui est soit dans la colonne 1 de la matrice, soit n'importe où.

Les auteurs donnent des bornes inférieures et supérieures de coût en nombre de mutations pour le problème restreint le plus simple, c'est-à-dire qui traite des contractions binaires, dont la taille des copies à contracter est la taille du motif de base et dont la limite gauche de ces copies se situe dans la colonne 1 de la matrice. Ils donnent également des algorithmes d'approximation.

Le problème de reconstruction de l'histoire des duplications en tandem est, comme on vient de le voir, un problème difficile. D'autres auteurs ont publié des résultats intéressants autour de ce thème : [Fitch, 1977], [Elémento et al., 2002], [Elémento et Gascuel, 2002], [Elémento et Gascuel, 2003], [Jaitly et al., 2002] et [Tang et al., 2002].

3.2.2 Alignement de séquences contenant des répétitions en tandem

Les premiers algorithmes d'alignement et de comparaison de séquences biologiques étaient basés exclusivement sur un modèle d'évolution qui suppose que les modifications de la séquence sont dues aux substitutions, insertions et délétions. Benson nomme ce modèle SI pour Substitutions et Indels. Ce modèle fonctionne convenablement mais il est devenu clair que d'autres événements mutationnels se produisent sur les séquences d'ADN, tels que les duplications en tandem. L'un des premiers articles à traiter le problème de l'alignement avec un modèle étendu aux duplications en tandem est l'article de Benson [Benson, 1997].

Un article de Varré et collaborateurs, [Varré et al., 1999], que nous n'étudions pas ici, définit une famille de distances appelées *distances de transformation*. Ces distances quantifient la dissimilarité entre deux séquences en termes d'événements basés sur des segments. Elles permettent aussi de prendre en comptent des duplications, des inversions et des insertions d'un segment d'ADN.

[Benson, 1997]

Dans cet article, l'auteur introduit un nouveau modèle d'évolution, le modèle DSI, qui en plus du modèle SI, inclut l'événement de Duplication en tandem (la duplication est similaire à l'événement d'amplification que nous avons défini page 38, à la différence qu'il peut s'appliquer sur plusieurs caractères à la fois). En utilisant le modèle DSI, Benson développe de nouveaux algorithmes exacts et heuristiques pour comparer et aligner des séquences d'ADN pouvant contenir des répétitions en tandem.

Dans son alignement, pour calculer les coûts des brèches[4], il utilise une fonction affine (voir chapitre 2, page 49). L'auteur se restreint, dans cet article, à l'alignement local.

Sous le modèle SI, pour calculer l'alignement local entre deux séquences s et r, avec une fonction de brèche affine, le score optimal d'alignement des préfixes $s[1..i]$ et $r[1..j]$ est le suivant :

$$\mathcal{A}(i,j) = max \begin{cases} E(i,j) \\ F(i,j) \\ M(i,j) \\ 0 \end{cases}$$

où

- $E(i,j)$ est le meilleur score possible pour un alignement avec une délétion à l'extrémité droite de $r[1..j]$;
- $F(i,j)$ est le meilleur score possible pour un alignement avec une délétion à l'extrémité droite de $s[1..i]$;
- $M(i,j)$ est le meilleur score possible pour un alignement finissant par un appariement exact ou une substitution entre $s[i]$ et $r[j]$;

[4] *Gaps* en anglais.

– 0 permet aux sous-séquences participant à l'alignement d'être optimalement sélectionnées.

Pour le modèle DSI, Benson ajoute une nouvelle option : l'option de duplication. Ainsi $Dup(i,j)$ est le meilleur score possible pour un alignement finissant par une duplication à l'extrémité droite de $s[1..i]$ ou à l'extrémité droite de $r[1..j]$. Il ne subdivise pas Dup comme E et F. La nouvelle récurrence pour le modèle DSI est donc de la forme suivante :

$$\mathcal{A}(i,j) = max \begin{cases} Dup(i,j) \\ E(i,j) \\ F(i,j) \\ M(i,j) \\ 0 \end{cases} \tag{3.1}$$

La principale difficulté est de calculer efficacement Dup, l'auteur utilise pour cela une variante du *Wraparound Dynamic Programming* (WDP). La méthode du WDP permet d'aligner une séquence a avec une répétition en tandem de motif b aussi longue que nécessaire, dans une matrice de programmation dynamique de taille $|a| \times |b|$. Cette méthode a été introduite pour la première fois par Myers et Miller [Myers et Miller, 1989] et indépendamment développée et simplifiée par Fischetti et ses collaborateurs [Fischetti et al., 1992].

Benson fait deux suppositions :

1. Pas de délétion d'une copie dans une région de répétitions en tandem ;
2. Les duplications se produisent avant les autres types de mutations (indels et substitutions).

Sans ces hypothèses, le problème serait équivalent au problème précédent de reconstruction de l'histoire des duplications. En effet, si les duplications et les autres types de mutation se produisent de manière entremêlée, un alignement entre deux séquences ne peut pas trouver le coût optimal sans retrouver l'histoire des événements.

Soit u une sous-chaîne de r et t une sous-chaîne de s. Pour calculer $Dup(i,j)$, le coût d'un alignement finissant par une duplication de motifs u alignée avec t, l'auteur se sert de γ, le coût d'initiation d'une duplication, et de $dupcost(u,t)$ le meilleur score d'alignement pour dupliquer u et aligner le résultat de cette duplication avec t. $dupcost(u,t)$ est composé d'un coût d'extension de duplication δ, pour chaque copie de u utilisée dans l'alignement, et du coût sous le modèle SI de la transformation des copies de u en t (γ et δ sont des valeurs négatives). Cet exemple d'alignement est illustré à la figure 3.6. La matrice Dup se remplit de la manière suivante :

$$Dup(i,j) = \max_{(u,t)} \begin{cases} \mathcal{A}(i-h, j-k) + \gamma + dupcost(u,t) - \delta, \\ \qquad \text{avec } t = s[i-h+1..i] \\ \qquad\qquad u = r[j-k+1..j] \\ \mathcal{A}(i-k, j-h) + \gamma + dupcost(u,t) - \delta, \\ \qquad \text{avec } u = s[i-k+1..i] \\ \qquad\qquad t = r[j-h+1..j] \end{cases} \tag{3.2}$$

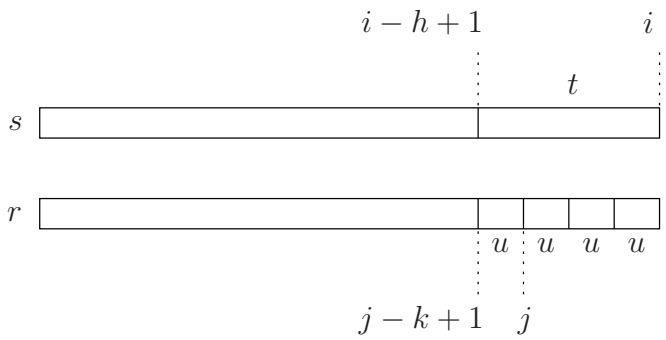


Fig. 3.6 – *Exemple d'alignement entre s et r terminant par une duplication du motif u alignée avec t. Les indices indiqués sont ceux de l'équation (3.2).*

Ici, $\mathcal{A}(i,j)$ est le score de l'alignement local sous le modèle DSI. Le terme $-\delta$ prend en compte le fait que si le nombre de copies de u est c, le coût d'extension doit être seulement de $(c-1)\times\delta$ car une copie n'est pas dupliquée. Étant donné que l'auteur formule *dupcost* pour inclure un δ par copie de u, la définition précédente inclus le terme de correction $-\delta$.

Notez que le terme γ ne doit pas être inclus si une seule copie de u est alignée avec t puisqu'il n'y a pas de duplication. L'auteur ne modifie pas le score de *Dup* pour prendre en considération cette possibilité car la valeur correcte sera prise en compte dans le calcul du score sous le modèle SI (cf. équation (3.1)).

L'algorithme 3 montre le squelette sur lequel l'auteur construit ses deux algorithmes exacts. Pour toutes les positions i dans s et pour toutes les positions j dans r, l'algorithme affecte $\mathcal{A}(i,j)$ avec le minimum entre :

- la valeur de l'alignement entre $s[1..i]$ et $r[1..j]$ sous le modèle SI (ligne **3**) ;
- la valeur d'un alignement entre $s[1..i]$ et $r[1..j]$ terminant par une duplication à la droite de r (ligne **10**) ;
- et la valeur d'un alignement entre $s[1..i]$ et $r[1..j]$ terminant par une duplication à la droite de s (ligne **17**).

La différence entre les deux algorithmes exacts proposés dans cet article est la manière de calculer *Dup*. Ces deux algorithmes sont respectivement en $O(n^5)$ et $O(n^4)$. Ils sont restreints à l'utilisation d'un nombre intégral de copies de u. Le second utilise une astuce de calcul pour faire descendre la complexité en temps mais du coup augmente la complexité en espace, on passe de $O(n^2)$ à $O(n^3)$. L'auteur propose également plusieurs heuristiques pour réduire la complexité en temps, comme réduire le nombre de paires (i,j) utilisées pour calculer *dupcost*, réduire le nombre de sous-séquences u utilisées, ou encore réduire la taille des sous-séquences, tout en essayant de ne pas sacrifier des informations biologiques pertinentes.

Algorithme 3: *Calcul du meilleur alignement local de deux séquences sous le modèle* DSI *[Benson, 1997].*

Données : Deux chaînes $s = [1..n]$ et $r = [1..m]$.

Résultat : Score du meilleur alignement local entre s et r sous le modèle DSI.

```
1  pour i = 1 à n faire
2      pour j = 1 à m faire
3          A(i,j) = maxscore{options du modèle SI (substitutions et indels)};
4          pour k = 1 à j faire
5              Sélectionner u tel que u = r[j-k+1..j];
6              pour h = 1 à i faire
7                  Sélectionner t tel que t = s[i-h+1..i];
8                  /* calculer dupoption avec u et t */
9                  Dup_r = A(i-h, j-k) + γ + dupcost(u,t) - δ;
10                 A(i,j) = max{A(i,j), Dup_r};
11         pour k = 1 à i faire
12             Sélectionner u tel que u = s[i-k+1..i];
13             pour h = 1 à j faire
14                 Sélectionner t tel que t = r[j-h+1..j];
15                 /* calculer dupoption avec u et t */
16                 Dup_s = A(i-k, j-h) + γ + dupcost(u,t) - δ;
17                 A(i,j) = max{A(i,j), Dup_s};
18 Trouver le meilleur score A(i,j);
19 Faire un parcours arrière et tracer l'alignement;
```

La méthode de Benson permet d'aligner des séquences qui peuvent contenir des répétitions en tandem. Pour prendre en compte les possibles événements de duplications, son algorithme doit identifier des répétitions locales dans les séquences, ce qui amène à une complexité élevée. Sous le modèle DSI, le coût d'une duplication est calculé en comparant la nouvelle unité à une unité dans l'autre séquence. Aussi, le modèle DSI ne peut pas prendre en compte le fait qu'une unité répétée et sa copie soient dans la même séquence, ce qui se produit en réalité dans l'évolution des séquences.

Chapitre 4

Alignement avec amplification et contraction sous le modèle SSE

Sommaire

4.1 Modèle d'évolution SSE 80
4.1.1 Événements évolutifs 80
4.1.2 Distance induite par le modèle SSE 81
4.2 Arches 84
4.2.1 Définition 84
4.2.2 Alignement et arches 85
4.2.3 Ordre des événements 87
4.3 Algorithme d'alignement 87
4.3.1 Calcul de la matrice de programmation dynamique 88
4.3.2 Coût de génération et compression d'arche 92
4.3.3 Prétraitement 95
4.3.3.1 Prétraitement à base de graphe 96
4.3.3.2 Prétraitement sans décalage 103
4.3.4 Complexité de l'algorithme 106
4.4 Autres alignements optimaux 106
4.4.1 Arches fantômes 107
4.4.2 Récurrence 108
4.4.3 Calcul des coûts de compression d'arche fantôme 109
4.4.4 Prétraitement 111
4.4.5 Complexité 111

Dans ce chapitre, nous proposons un algorithme d'alignement de séquences qui prend en compte les événements de délétion, d'insertion, de mutation, d'amplification et de contraction. Cet algorithme est approprié pour la comparaison de cartes de minisatellites.

La difficulté de ce problème d'alignement vient du fait que les opérations ne sont pas commutatives, l'ordre d'application de celles-ci est donc important. Par exemple, l'amplification d'un caractère a à partir d'un a adjacent peut s'effectuer avant que ce dernier ne soit muté en un caractère b, mais pas après. L'algorithme que nous présentons à la section 4.3 est le premier algorithme à prendre en compte les événements d'amplification et de contraction d'un caractère. Il calcule une distance entre deux séquences en temps polynômial. Cet algorithme est basé sur la programmation dynamique et la résolution de problèmes de stable max. Il est fondé sur un modèle d'évolution symétrique et unaire que nous appelons sse. Cet algorithme a donné lieu à deux publications : [Bérard et Rivals, 2002] et [Bérard et Rivals, 2003]. Dans la partie Biologie, au chapitre 8, nous appliquons cette méthode pour comparer des cartes du minisatellite MSY1 provenant de (Jobling et al., 1998). Cet algorithme a également été utilisé par É. Fontanillas pour une étude du minisatellite MS205 (Fontanillas, 2002).

Le chapitre est organisé de la manière suivante : nous présentons le modèle sse et prouvons qu'il induit une distance dans la section 4.1. Dans la section 4.2, nous introduisons un type de facteur particulier sur les séquences que nous appelons arche et dont nous nous servons pour notre algorithme d'alignement. Cet algorithme est détaillé à la section 4.3. Enfin, dans la section 4.4, nous expliquons comment retrouver tous les alignements optimaux entre deux séquences.

Dans tout ce chapitre, s et r sont deux chaînes, par exemple deux cartes de minisatellite, sur un alphabet Σ, de longueurs respectives, n et m.

4.1 Modèle d'évolution SSE

Dans cette section, nous décrivons le modèle évolutif nommé sse pour *Single Step Evolutionary model*, nous définissons une distance à partir de ce modèle et nous prouvons que cette distance est une distance au sens mathématique.

4.1.1 Événements évolutifs

Le modèle sse considère cinq d'événements évolutifs sur les variants : mutation, insertion, délétion, amplification et contraction. L'événement mutation sur un motif/variant est identique à l'événement substitution sur un nucléotide dans le sens où il remplace un caractère par un autre. Mutation, insertion, et délétion sont les événements classiques considérés dans les alignements de séquences en biologie (voir chapitre 2, page 38) sauf qu'ici ils sont appliqués à des variants au lieu d'être appliqués à des nucléotides. Voici une illustration de ces trois opérations sur une séquence abc :

Mutation de b en d :	$abc \to adc$.
Insertion de d à la position 3 :	$abc \to abdc$;
Délétion de b :	$abc \to ac$;

La mutation de a en b transforme le variant a en variant b. L'insertion insère un variant à une position donnée dans la carte, tandis que la délétion, son événement inverse, supprime un variant à une position donnée.

Les deux événements spécifiques du modèle SSE sont l'amplification et la contraction. Nous considérons dans ce chapitre des amplifications et contractions d'un seul variant à la fois et ne produisant/supprimant qu'une seule copie (c'est-à-dire d'ordre 1 et d'arité 1 seulement, voir définitions 25 et 26, page 125). Par exemple, la carte *abc* subit successivement une amplification du variant b, puis sa contraction :

$$\begin{array}{ll} \text{Amplification :} & abc \to abbc\,; \\ \text{Contraction :} & abbc \to abc. \end{array}$$

Dans une carte s, l'amplification du variant $s[i]$ donne comme résultat l'insertion d'une nouvelle copie de $s[i]$ à la position i+1. La contraction est l'opération inverse : si $s[i]$ apparaît à deux positions successives, c'est-à-dire si $s[i] = s[i+1]$, l'opération de contraction à la position $i+1$ retire $s[i+1]$.

Le modèle SSE est dit unaire (*Single Step*), car l'amplification, resp. la contraction, ajoute, resp. retire, un seul variant à la fois. L'amplification non plus d'un variant mais d'un bloc de variants, par exemple *ab* de la séquence *abc* donnerait *ababc*, correspond à une amplification d'ordre supérieur à 1 et n'est pas autorisée ici. Dans le chapitre 6, nous considérons aussi des modèles plus étendus où les triplications ou quadruplications d'un variant peuvent arriver en une fois, cela correspond à des amplifications d'arité supérieure à 1.

4.1.2 Distance induite par le modèle SSE

Pour compléter le modèle, nous avons besoin d'un critère quantitatif pour juger de la similarité des cartes. Pour cela, à chaque opération est associé un coût réel strictement positif. Une série d'événements qui transforme s en r est appelée un alignement. Le coût de l'alignement est la somme des coûts des opérations qui le composent. Nous désignons chaque coût par l'initiale majuscule de l'opération correspondante : M, I, D, A et C.

Ici, nous considérons un modèle symétrique où les opérations inverses ont le même coût : $I = D$ et $A = C$. La fréquence des amplifications et des contractions sur les cartes de minisatellite étant plus élevée que celle des autres opérations, ces deux événements ont un coût plus faible : $A, C < M, D, I$. Pour unifier les notations de la mutation et de l'identité, nous utilisons $M(a, b)$ qui est égal à 0 si $a = b$ et à M sinon.

Pour des raisons pratiques, nous considérons que toutes les mutations possibles ont le même coût, sans tenir compte de la séquence nucléotidique des variants. C'est une hypothèse raisonnable pour l'application aux minisatellites car les variants sont longs et diffèrent l'un de l'autre par un petit nombre de paires de bases (voir le cas de MSY1 au chapitre 7, section 7.3). Par exemple, soit v_1, v_2 et v_3 trois variants d'une carte de minisatellite, respectivement égaux à *cggcgat*, *cggcga***c** et *cgg***a***gat*. Dans notre modèle,

muter le variant v_1 en variant v_2 ou muter le variant v_2 en variant v_3 coûtent la même chose, M, alors que cela nécessite une substitution de nucléotide dans le premier cas, et deux dans le second.

Notons qu'une délétion peut également être obtenue par une mutation suivie d'une contraction et une insertion par une amplification suivie d'une mutation. Suivant le coût affecté aux opérations, on préférera une solution à l'autre. En effet, comme notre modèle est symétrique, nous avons soit :

- H1 : ($D > M + C$ et $I > A + M$) ;
- H2 : ($D \leq M + C$ et $I \leq A + M$).

Sans perte de généralité, nous supposons la première hypothèse, H1. Cela influence le coût des générations/compressions d'arche (cf. section 4.3.2) et modifie légèrement l'algorithme. Sous l'hypothèse H1 un variant est « inséré » par une amplification+mutation, ou « délété » par une mutation+contraction. Nous notons ces opérations par A_M et M_C, leurs coûts sont $A + M$ et $M + C$ respectivement. A_M et M_C sont des opérations élémentaires dans le sens où elles s'appliquent à un seul variant. Le modèle sse contient donc sept opérations élémentaires, M, I, D, A, C, A_M et M_C.

Le coût d'alignement que nous venons de définir est une métrique ; ceci est important pour l'utilisation du coût en tant que distance, par exemple en reconstruction phylogénétique.

Théorème 2 Distance *Le coût d'alignement définit une métrique sur* Σ^*.

Preuve (**Distance métrique**) Nous voulons montrer que notre coût d'alignement est une distance au sens mathématique, c'est-à-dire qu'il vérifie les propriétés données au chapitre 2, page 41 :

(i) Positivité ;

(ii) Séparation ;

(iii) Symétrie ;

(iv) Inégalité triangulaire.

Nous construisons notre preuve sur l'analyse des propriétés métriques des distances d'alignement de [Sankoff et Kruskal, 1999, Chapitre 9, p. 307-8].

Quelques propriétés des coûts élémentaires sont directement transmises à la distance entre les cartes :

(i) La positivité, puisque tous les coûts élémentaires sont positifs ;

(ii) La séparation, puisque seule l'identité coûte zéro ;

(iii) La symétrie, puisque les coûts des opérations inverses sont les mêmes.

Il reste à prouver (iv), l'inégalité triangulaire. Soit r, s, t trois cartes. Notons d notre distance d'alignement entre deux cartes. Nous devons montrer que $d(r, t) \leq d(r, s) + d(s, t)$. L'inégalité triangulaire peut ne pas être respectée si et seulement si nous ne pouvons pas combiner l'alignement de r à s et de s à t en un alignement de coût inférieur de r à t.

Première	Seconde	Résultat	Première	Seconde	Résultat
$A(-,a)$	$C(a,-)$	*pop*	$C(a,-)$	$A(-,a)$	$M(a,a)$
$A(-,a)$	$D(a,-)$	*pop*	$C(a,-)$	$I(-,a)$	$M(a,a)$
$A(-,a)$	$M(a,b)$	$A_M(-,b,a)$	$C(a,-)$	$I(-,b)$	$M(a,b)$
$A_M(-,b,a)$	$D(b,-)$	*pop*	$C(a,-)$	$A_M(-,b,a)$	$M(a,b)$
$A_M(-,b,a)$	$M(b,c)$	$A_M(-,c,a)$	$M_C(a,-,b)$	$A(-,b)$	$M(a,b)$
$A_M(-,b,a)$	$M(b,a)$	$A(-,a)$	$M_C(a,-,b)$	$I(-,a)$	$M(a,a)$
$A_M(-,b,a)$	$M_C(b,-,a)$	*pop*	$M_C(a,-,b)$	$I(-,b)$	$M(a,b)$
$I(-,a)$	$D(a,-)$	*pop*	$M_C(a,-,b)$	$I(-,c)$	$M(a,c)$
$I(-,a)$	$C(a,-)$	*pop*	$M_C(a,-,b)$	$A_M(-,c,b)$	$M(a,c)$
$I(-,a)$	$M_C(a,-,b)$	*pop*	$M_C(a,-,b)$	$A_M(-,a,b)$	$M(a,a)$
$I(-,a)$	$M(a,b)$	$I(-,b)$	$M(a,b)$	$C(b,-)$	$M_C(a,-,b)$
$D(a,-)$	$A(-,a)$	$M(a,a)$	$M(a,b)$	$D(b,-)$	$D(a,-)$
$D(a,-)$	$I(-,b)$	$M(a,b)$	$M(a,b)$	$M_C(b,-,a)$	$C(a,-)$
$D(a,-)$	$I(-,a)$	$M(a,a)$	$M(a,b)$	$M_C(b,-,c)$	$M_C(a,-,c)$
$D(a,-)$	$A_M(-,b,a)$	$M(a,b)$	$M(a,b)$	$M(b,a)$	$M(a,a)$
			$M(a,b)$	$M(b,c)$	$M(a,c)$

TAB. 4.1 – *Toutes les combinaisons possibles de paires d'opérations élémentaires et pour chacune une opération équivalente, où pop signifie « pas d'opération » et M(a,a) est mis pour un appariement exact.*

Ceci ne peut être le cas que si deux opérations élémentaires successives appliquées à la même position dans l'alignement coûtent moins cher qu'une simple opération. Nous avons donc besoin de vérifier toutes les paires possibles d'opérations élémentaires. Pour cela, nous avons besoin d'introduire des notations légèrement plus complexes. Tout d'abord, l'alphabet d'alignement est $\Sigma \cup \{-\}$ où $-$ est le symbole d'absence de variant. Si E est une opération élémentaire, nous notons $E(a,b)$ l'opération E qui transforme le *symbole source*, a, en *symbole destination*, b, où $a, b \in \Sigma \cup \{-\}$. Ainsi, si a, b, c sont trois variants distincts de Σ, nous notons par $A(-,a)$ l'amplification (elle transforme $-$ en a à partir d'un a adjacent), par $C(a,-)$ la contraction, par $A_M(-,b,a)$ l'amplification+mutation (elle amplifie le a adjacent et ensuite le mute en b, ici le troisième argument est le variant adjacent se trouvant à gauche), par $M_C(b,-,a)$ la mutation+contraction (elle mute b en a et contracte la position en un a adjacent, ce qui produit un $-$), par $M(a,b)$ la mutation si $a \neq b$ et l'identité si $a = b$, par $I(-,a)$ l'insertion d'un a, et par $D(a,-)$ la délétion d'un a. Maintenant, certaines paires d'opérations ordonnées sur la même position d'alignement sont impossibles :

- une fois que la position a été insérée, amplifiée, ou amplifiée+mutée de r à s, elle ne peut pas être à nouveau insérée, amplifiée ou amplifiée+mutée de s à t ;
- une fois que la position est présente, c'est-à-dire après une identité ou une mutation, la position ne peut pas être insérée, amplifiée ou amplifiée+mutée ;

– une fois que la position a été supprimée de r à s, elle ne peut pas être supprimée à nouveau de s à t ; ainsi, ni deux délétions, contractions ou M_C successives, ni une des précédentes suivie par une mutation ou une identité est possible ;
– quels que soient a et $b \in \Sigma$, $A(-,a)M_C(a,-,b)$ est impossible car amplifier un a nécessite un a à la position gauche et muter et contracter un a en b nécessite un b sur la gauche ; pour la même raison, $A_M(-,b,a)C(b,-)$ n'est pas admis.

Dans la table 4.1, nous montrons que toutes les paires possibles restantes peuvent être remplacées par une seule opération ou pas d'opération du tout (ce que nous avons noté *pop*). Dans de telles paires, le symbole destination de la première opération doit être le même que le symbole source de la deuxième. Cette preuve est indépendante de l'hypothèse H1, c'est pourquoi nous incluons des paires qui ne sont pas toujours optimales, ni possible suivant les variants voisins. Il est direct de vérifier que les opérations élémentaires coûtent moins que les paires, cela prouve l'inégalité triangulaire et termine la preuve. □

* *

*

Le problème que nous traitons dans ce chapitre est le suivant :

Définition 21 (Alignement avec amplification et contraction sous le modèle sse**)** *Soient deux séquences s et r de longueurs respectives n et m. Le problème de l'alignement de séquences avec amplification et contraction est de trouver un alignement global optimal de score minimum entre s et r, sous le modèle* sse*.*

4.2 Arches

Dans cette section, nous introduisons la notion d'arche. Dans un premier temps, nous donnons la définition des arches. Nous montrons ensuite un alignement entre séquences sous le modèle sse en utilisant les arches. Enfin nous discutons de l'ordre des opérations induit par les arches dans l'alignement que nous calculons.

4.2.1 Définition

Regardons pas à pas l'exemple artificiel d'évolution du même minisatellite dans deux individus ; ceci est montré à la figure 4.1.

L'alphabet des variants est $\{a,b,c,d,e\}$. L'état ancêtre a la carte *aaaaa*. Les cartes résultantes chez les individus 1 et 2 sont $r = aeaaa$ et $s = aaabbcbddba$. Un alignement de ces deux cartes est donné à la figure 4.2.

Chez l'individu 2, le variant b est amplifié cinq fois. Ces variants amplifiés subissent ensuite d'autres amplifications et mutations. Cela résulte en une sous-chaîne dans laquelle les variants proviennent du même variant ancêtre, qui est appelé la **graine** de la sous-chaîne. Nous pouvons délimiter cette sous-chaîne par le fait que les variants à ses extrémités sont

Individu 1

Événement	Séquence
	a a a a a
mutation	*a* **e** *a a a*
	1 2 3 4 5

Individu 2

Événement	Séquence
	a a a a a
mutation	*a a a* **b** *a*
5*amplification	*a a a b* **b b b b b** *a*
mutation	*a a a b b* **c** *b b b a*
mutation	*a a a b b c b* **d** *b a*
amplification	*a a a b b c b d* **d** *b a*
	1 2 3 4 5 6 7 8 9 10 11

FIG. 4.1 – *Exemple d'évolution d'un minisatellite chez 2 individus.*

identiques. Dans notre exemple, la sous-chaîne de $s[4..10] = bbcbddb$ a pour graine le b de la position 4. Pendant l'évolution de cette sous-chaîne, le variant final à chaque position n'apparaît pas dans l'ordre de la séquence (gauche-droite ou inversement) : à un moment donné, la position 8 a toujours l'état ancêtre b et pas son état final d, tandis que la position 10 a déjà son état final, un b. Par conséquent, de telles sous-chaînes peuvent être obtenues par plusieurs séries d'opérations, mais la solution optimale sera dépendante de l'ordre de ces opérations. Si nous calculons incrémentalement l'alignement pour des préfixes de plus en plus longs, comme cela est fait dans les algorithmes d'alignement classiques, c'est-à-dire en ajoutant une opération à la fois, nous ne pouvons pas trouver l'ordre optimal des événements. Pour notre exemple de sous-chaîne, si la position 6 devient d'abord un c, il n'est pas possible d'obtenir un b à la position 7 par une amplification. Ainsi, pour retrouver la série optimale d'opérations qui conduit à une telle sous-chaîne, nous devons la considérer comme un tout et non pas variant après variant. Cela nous mène à la notion d'arche.

Définition 22 (Arche) *Soit s une carte de longueur n et i, j deux entiers tels que $1 \leq i < j \leq n$. La sous-chaîne $s[i..j]$ est une* **arche** *de s si et seulement si $s[i] = s[j]$.*

Dans une arche, nous différencions les variants externes ou extrémaux, qui par définition sont identiques, de tous les autres, que nous appelons internes. Tous les variants d'une arches ne sont pas forcément identiques, et une arche peut contenir récursivement d'autres arches, appelées *arches internes*. La graine d'une arche interne peut être différente de la graine de l'arche qui l'englobe, comme *dd* dans *bbcbddb*. En effet, une fois qu'une position interne a été mutée, elle peut subir une amplification, qui va créer une arche interne.

4.2.2 Alignement et arches

Nous montrons ici un alignement sous le modèle SSE qui prend en compte les arches des séquences. La figure 4.2 montre un exemple d'un tel alignement. Cet alignement peut se lire de la manière suivante :

1. Le e à la position 2 de r est muté en a ;

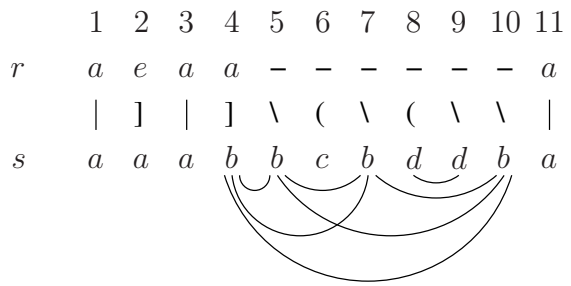


Fig. 4.2 – *Alignement des cartes des individus 1 et 2 de la figure 4.1. L'arche bbcbddb et ses arches internes sont représentées par des arcs en dessous de s. Dans la ligne du milieu,* '|', ']', '\', '(' *représentent respectivement un appariement exact, une mutation, une amplification et une* A_M.

2. Le a à la position 4 de r est muté en b ;
3. Ce b est ensuite amplifié cinq fois ;
4. Le b de la position 6 est muté en c ;
5. Le b de la position 8 est muté en d ;
6. Enfin, le d de la position 8 est amplifié une fois.

L'ordre des événements est important, l'événement 3 ne peut pas se produire avant le 2, de même l'événement 6 ne peut pas se produire avant le 5. Les événements 4 et 5 sont interchangeables mais doivent se produire après le 3. L'événement 1 peut se placer n'importe où.

Dans l'alignement de la figure 4.2, on peut voir qu'aucune position correspondant à l'arche *bbcbddb* de s existe dans r à l'exception de la position de la graine. Pour imiter l'évolution, nous voulons aligner une arche de s avec un seul variant de r, celui à la position originale de la graine.

Nous pouvons voir l'arrivée d'une arche comme un seul événement évolutif avec une fonction de coût spécifique. Selon le sens dans lequel on observe l'évolution, de s à r ou de r à s, nous nommons cette événement **compression** d'arche ou **génération** d'arche. Dans notre approche par programmation dynamique, les générations/compressions d'arche sont vues comme une seule opération qui exprime une dépendance entre deux cases non adjacentes de la matrice de programmation dynamique (voir figure 4.4, page 89).

La génération et la compression d'arche sont symétriques. Sous l'hypothèse H1, celle que nous avons choisie, la génération d'arche correspond à une série d'amplifications et de mutations qui donne naissance à la sous-chaîne associée à partir du variant graine (sous l'hypothèse H2, ce serait une série d'amplifications et d'insertions). Symétriquement, la compression sous H1 est une série de contractions et de mutations qui « rembobine » la sous-chaîne en sa graine.

Supposons que l'on veuille générer une arche dans s. Comme la génération n'implique pas de caractère de r, son coût optimal peut être calculé indépendamment de r. Nous

notons $G(t)$ et $K(t)$ les coûts de génération et de compression d'une arche t. À partir de maintenant nous ne considérons plus que les compressions d'arche à cause de la symétrie. Dans la section 4.3.2, nous expliquons comment calculer le coût d'une compression d'arche et montrons que cela amène à une économie d'au moins M sur toutes les autres séries d'opérations.

Une question se pose : est-ce que nous avons besoin de considérer des séquences d'opérations dans lesquelles l'ordre importe, pour des sous-chaînes qui ne sont pas des arches ? Si nous voulons retrouver tous les alignements de coût optimal, la réponse est oui ; la section 4.4 est dédiée à cela. Cependant, si nous voulons seulement calculer la distance entre deux cartes, la réponse est non ; nous prouvons cela dans le lemme 2, page 91.

4.2.3 Ordre des événements

La discussion sur les arches et l'exemple de la figure 4.1 soulèvent plusieurs points.

Premièrement, dans une seule colonne d'un alignement, plusieurs événements, au plus deux, peuvent apparaître. C'est le cas dans la colonne 6 de la figure 4.2 qui est créée par une amplification du variant voisin b et une mutation de b en c.

Deuxièmement, il est clair à partir de cet exemple que l'ordre des opérations est important. Notamment, l'amplification qui crée le b à la position 10 doit être effectuée avant la mutation qui change le b de la position 8 en d. En terme mathématique, les opérations ne sont pas *commutatives*.

Troisièmement, les alignements ne représentent pas bien toutes les informations qui sont contenues dans la transformation d'une séquence à l'autre. En réalité, les **listings**, comme définis dans [Sankoff et Kruskal, 1999] et rappelés page 40 de cette thèse, sont de meilleurs représentants pour notre modèle d'évolution. Les deux tables présentées à la figure 4.1 sont des exemples de listings. Étant donné deux cartes, notre algorithme calcule le score d'un listing optimal.

Quatrièmement, la non-commutativité nous force à considérer l'ordre des opérations pour calculer les compressions et générations optimales d'une arche. Cela explique pourquoi nous avons besoin d'une procédure spécifique pour calculer le coût d'une compression/génération d'arche, et pourquoi cette procédure ne peut pas considérer l'arche variant après variant, mais doit l'utiliser comme un tout. En fait, la non-commutativité accroît la complexité du calcul.

4.3 Algorithme d'alignement

Dans cette partie, nous voulons calculer l'alignement global optimal entre deux cartes de minisatellite sous le modèle SSE. Dans la section 4.3.1, nous présentons notre approche par programmation dynamique en supposant que les générations et compressions d'arche sont des opérations unitaires. Dans la section 4.3.2, nous montrons comment calculer le coût d'une génération ou compression d'arche. Pour cela nous avons besoin de trouver un ensemble maximal d'arches compatibles. Ensuite, dans la section 4.3.3, nous décrivons deux

$$\text{Initialisation : } \begin{array}{l} \mathcal{A}(0,0) = 0 \\ \mathcal{A}(1,0) = D \\ \mathcal{A}(0,1) = I \end{array}$$

Pour toutes les autres entrées, c'est-à-dire telles que $i + j > 1$, la récurrence est :

$$\mathcal{A}(i,j) = \min \begin{cases} \mathcal{A}(i-1,j-1) + M(s[i], r[j]) & \text{Mutation ou Appariement Exact} \\ & \text{si } i > 0 \text{ et } j > 0 \\ \mathcal{A}(i-1,j) + C & \text{Contraction} \\ & \text{si } i > 1 \text{ et } (s[i] = s[i-1] \text{ ou } s[i] = r[j]) \\ \mathcal{A}(i-1,j) + M + C & \text{Mutation+Contraction} \\ & \text{si } i > 1 \\ \mathcal{A}(i,j-1) + A & \text{Amplification} \\ & \text{si } j > 1 \text{ et } (r[j] = r[j-1] \text{ ou } r[j] = s[i]) \\ \mathcal{A}(i,j-1) + A + M & \text{Amplification+Mutation} \\ & \text{si } j > 1 \\ \mathcal{A}(l,j) + K(s[l..i]) & \text{Compression d'arche} \\ & \text{si } i > 2, \forall l \in [1, i-2] \text{ tel que } s[l] = s[i] \\ \mathcal{A}(i,l') + G(r[l'..j]) & \text{Génération d'arche} \\ & \text{si } j > 2, \forall l' \in [1, j-2] \text{ tel que } r[l'] = r[j] \end{cases}$$

Fig. 4.3 – *Équation de récurrence.*

manières de prétraiter les séquences dans le but de calculer ces ensembles compatibles pour toutes les arches considérées pendant le calcul. Enfin, dans la section 4.3.4, nous établissons la complexité de l'algorithme.

4.3.1 Calcul de la matrice de programmation dynamique

Nous notons $\mathcal{A}$ la matrice de programmation dynamique de dimension $(n+1) \times (m+1)$. Ses lignes et ses colonnes sont indexées de 0 à n et de 0 à m respectivement. La case $\mathcal{A}(i,j)$ donne la distance d'alignement entre les préfixes de s et r de longueurs i et j respectivement, c'est-à-dire entre $s[1..i]$ et $r[1..j]$. La case $\mathcal{A}(n,m)$ donne la distance entre les cartes $s = s[1..n]$ et $r = r[1..m]$, ce que nous voulons calculer. Les cartes s et r sont indexées de 1 à n et de 1 à m respectivement. La figure 4.3 montre l'équation de récurrence sous l'hypothèse H1.

Tout d'abord, notons que l'insertion et la délétion ne sont utilisées que pour calculer

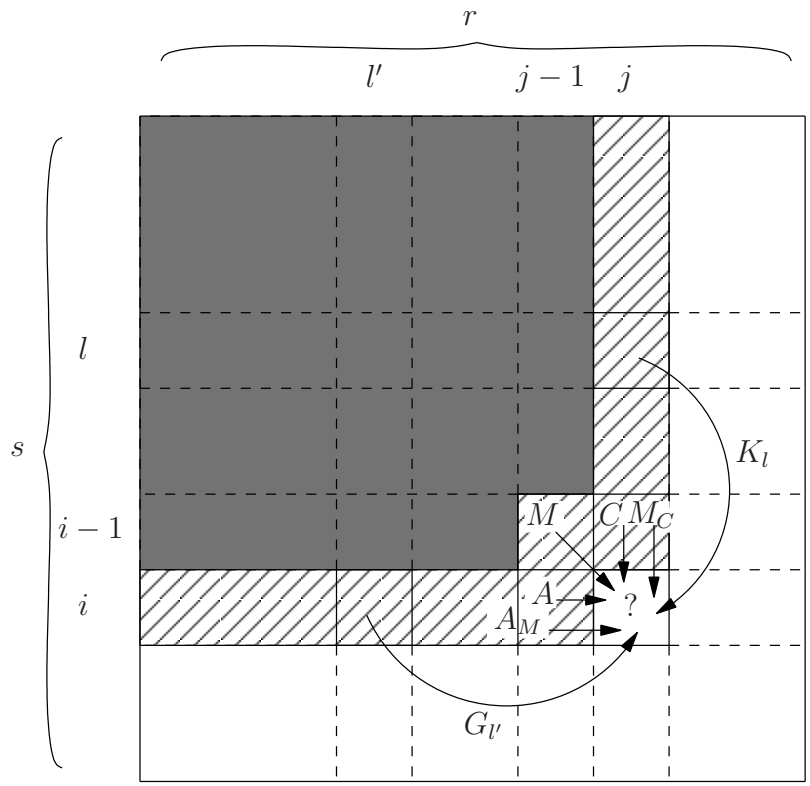


FIG. 4.4 – *Dépendances dans la matrice de programmation dynamique sous l'hypothèse H1. Pour calculer la cellule $\mathcal{A}(i,j)$, nous avons besoin de cellules de la partie rayée mais pas de la partie gris foncé. Les dépendances sont indiquées par des flèches.*

$\mathcal{A}(1,0)$ et $\mathcal{A}(0,1)$. C'est parce que l'amplification n'est pas autorisée dans une chaîne vide et que la contraction n'est pas autorisée dans une chaîne ne contenant qu'un seul variant. Pour toutes les autres entrées, quand une insertion ou une délétion est nécessaire, on utilise A_M ou M_C à cause de H1. La récurrence de programmation dynamique prend le minimum entre les sept termes montrés à la figure 4.3. Ces termes expriment des dépendances, illustrées par des flèches, entre cases de la matrice de programmation dynamique montrée à la figure 4.4.

Les cinq premiers termes sont dus aux opérations élémentaires et les autres termes aux générations et compressions d'arche. Ces derniers dépendent des positions de début, l ou l', de l'arche compressée ou générée. L'ensemble des positions pour l et l' est défini par les conditions données dans la récurrence et ainsi, les flèches de la figure 4.4 étiquetées $G_{l'}$ ou K_l représentent plusieurs termes, au plus $(j-1)$ et $(i-1)$ respectivement.

Une arche de longueur 2 peut être compressée par une simple contraction. Symétriquement, les générations d'arche de longueur 2 sont réductibles à une amplification. De telles compressions et générations d'arche ne sont donc pas considérées dans les deux dernières lignes de la récurrence. C'est pourquoi, la dernière position d'une arche doit être dans $[1, i-2]$ avec $i > 2$ pour les compressions et dans $[1, j-2]$ avec $j > 2$ pour les générations.

Un appariement exact ou une mutation ne peut pas se produire dans la première ligne ou la première colonne de la matrice, il s'ensuit que $i > 0$ et $j > 0$.

Pour n'importe quel terme impliquant une contraction ou une amplification, il est nécessaire qu'il y ait deux variants adjacents identiques, ce qui est possible seulement si $i > 1$ ou $j > 1$.

Considérez un alignement entre $s[1..i]$ et $r[1..j]$ dans lequel $s[i]$ est contracté. $s[i]$ peut être contracté soit en $s[i-1]$, soit en $r[j]$. En effet, quand on calcule $\mathcal{A}(i,j)$, $\mathcal{A}(i-1,j)$ est déjà calculé et ainsi nous connaissons une série optimale d'opérations Ω qui transforme $s[1..i-1]$ en $r[1..j]$. Pour obtenir un alignement entre $s[1..i]$ et $r[1..j]$ où $s[i]$ est contracté en $s[i-1]$, nous contractons d'abord $s[i]$ en $s[i-1]$ et ensuite nous appliquons Ω ; si nous voulons que $s[i]$ soit contracté en $r[j]$, nous appliquons d'abord Ω et ensuite nous contractons $s[i]$ en $r[j]$. C'est pourquoi la condition pour la contraction est $s[i] = s[i-1]$ ou $s[i] = r[j]$. La condition pour l'amplification est symétrique : $r[j] = r[j-1]$ ou $r[j] = s[i]$.

Sous l'hypothèse H2, l'équation de récurrence est légèrement modifiée. Les lignes Mutation+Contraction et Amplification+Mutation sont remplacées par les lignes de Délétion et Insertion :

$$\mathcal{A}(i-1,j) + D \qquad \text{Délétion si } i > 1$$

$$\mathcal{A}(i,j-1) + I \qquad \text{Insertion si } j > 1$$

L'algorithme 4 synthétise le calcul que nous venons de décrire.

Algorithme 4: *Alignement avec amplification et contraction sous le modèle* sse.

Données : Deux cartes s et r de longueurs respectives n et m.
Résultat : Le coût d'alignement optimal entre s et r.
$\mathcal{A}(0,0) = 0$; $\mathcal{A}(1,0) = D$; $\mathcal{A}(0,1) = I$;
pour $i = 0$ *à* n **faire**
  pour $j = 0$ *à* m **faire**
    si $i + j > 1$ **alors** $\mathcal{A}(i,j) =$ résultat de la récurrence p. 88 ;
Retourner $\mathcal{A}(n,m)$;

Nous démontrons maintenant que l'algorithme 4 est correct.

Théorème 3 Correction de l'algorithme 4 *L'algorithme 4 calcule le coût d'alignement optimal entre deux cartes.*

Preuve (**Correction de l'algorithme 4**)

Initialisation : L'alignement optimal entre deux cartes vides coûte zéro, entre une carte vide et une carte contenant un seul variant coûte I ou symétriquement D.

Hypothèse d'induction : Nous supposons que $\forall p, q$ tels que $(p \leq i)$ et $(q \leq j)$ et $(p \neq i$ ou $q \neq j)$, $\mathcal{A}[p,q]$ est égal au coût optimal d'alignement entre $s[1..p]$ et $r[1..q]$.

Nous prouvons que l'équation de récurrence (fig. 4.3, p. 88) calcule correctement $\mathcal{A}[i,j]$. Tous les alignements entre $s[1..i]$ et $r[1..j]$ peuvent être décomposés en un alignement de plus petits préfixes plus une opération finale. La récurrence prend le minimum parmi tous ces alignements possibles.

Pour compléter la preuve, nous avons besoin de prouver que pour toutes les sous-chaînes $s[p..i]$ autres que des arches, il est inutile de considérer un autre ordre d'opérations élémentaires que celui induit par la programmation dynamique. Ceci est montré dans le lemme 2 et conclut la preuve. □

Lemme 2 *Lorsque que l'on calcule $\mathcal{A}(i,j)$, la compression de la sous-chaîne $s[p..i]$, où $p \in [1, i-1]$ et $s[p] \neq s[i]$, mène à un coût supérieur ou égal au résultat de l'équation de récurrence.*

Corollaire *Pour calculer le coût de l'alignement, il n'est donc pas nécessaire de considérer des compressions de sous-chaînes qui ne sont pas des arches.*

Preuve (**Lemme 2**) Nous supposons l'hypothèse d'induction suivante :

$$\forall p, q \text{ tels que } (p \leq i) \text{ et } (q \leq j) \text{ et } (p \neq i \text{ ou } q \neq j), \mathcal{A}[p,q] \text{ est optimal.}$$

Nous examinons les différentes possibilités de compresser la sous-chaîne $s[p..i]$, $s[p] \neq s[i]$, en un seul variant, soit $s[p]$, soit $s[i]$, ce dernier étant aligné avec $r[j]$.

Il y a 3 cas : **1.** $s[i] = r[j]$ **2.** $s[p] = r[j]$ **3.** $s[i] \neq r[j]$ et $s[p] \neq r[j]$.

1. $s[i] = r[j]$: Nous alignons $s[i]$ avec $r[j]$, et il y a trois options pour compresser la sous-chaîne $s[p..i]$:

(a) Muter $s[p]$ en $s[i]$, supprimer les $(i-p-1)$ variants internes au coût optimal R, et contracter $s[p]$ en $s[i]$. Cette option coûte $\mathcal{A}(p-1, j-1) + M + R + C$;

(b) Muter et contracter $s[p]$ en $s[p+1]$, et supprimer de manière optimale les $(i-p-1)$ variants internes au coût R. Cela mène au coût $\mathcal{A}(p-1, j-1) + M + C + R$.

(c) Si $s[p] = s[p+1]$, contracter $s[p+1]$ en $s[p]$, et supprimer de manière optimale les $(i-p-1)$ variants internes au coût R. Cela mène au coût $\mathcal{A}(p,j) + C + R$.

Les options (a) et (b) coûtent $\mathcal{S} = \mathcal{A}(p-1, j-1) + R + M + C$ et nous avons :
– $\mathcal{A}(p,j) \leq \mathcal{A}(p-1, j-1) + M$ (condition de récurrence) ;
– $\mathcal{A}(i-1, j) \leq \mathcal{A}(p,j) + R$ (hypothèse d'induction).
À partir de ces deux inégalités nous obtenons :

$$\mathcal{A}(i-1, j) + C \leq \mathcal{A}(p-1, j-1) + R + M + C = \mathcal{S}$$

Comme $\mathcal{A}(i-1, j) + C$ est un terme de l'équation de récurrence, $\mathcal{A}(i,j) \leq \mathcal{S}$, ce qui montre qu'il est inutile de considérer ces deux options de compression.

En ce qui concerne l'option (c), on a $\mathcal{A}(i-1, j) \leq \mathcal{A}(p,j) + R$ par hypothèse d'induction, donc $\mathcal{A}(i-1, j) + C \leq \mathcal{A}(p,j) + C + R$. Comme $\mathcal{A}(i-1, j) + C$ est un terme de l'équation de récurrence, nous n'avons pas à considérer cette option ;

2. $s[p] = r[j]$: Ce cas cas est symétrique au cas **1.** où p joue le rôle de i et *vice versa*. Nous pouvons muter $s[i]$ en $s[p]$, supprimer les $(i-p-1)$ variants internes au coût optimal R, et contracter $s[i]$ en $s[p]$. Cela mène à un coût $\mathcal{S} = \mathcal{A}(p-1, j-1) + M + R + C$. Comme précédemment, nous avons :
– $\mathcal{A}(p, j) \leq \mathcal{A}(p-1, j-1)$ (condition de récurrence avec $s[p] = r[j]$) ;
– $\mathcal{A}(i-1, j) \leq \mathcal{A}(p, j) + R$ (hypothèse d'induction).
En combinant avec l'équation de récurrence, cela donne :

$$\mathcal{A}(i, j) \leq \mathcal{A}(i-1, j) + M + C \leq \mathcal{A}(p-1, j-1) + R + M + C = \mathcal{S}$$

et montre qu'il est inutile de considérer cette option de compression.
Le symétrique de l'option (c) du cas précédent se présente si $s[i] = s[i-1]$, il faut alors contracter $s[i]$ en $s[i-1]$ et supprimer les $(i-p-1)$ variants internes au coût optimal R. Cela mène à un coût $\mathcal{A}(p-1, j-1) + R + C \geq \mathcal{A}(i-1, j) + C$. Comme $\mathcal{A}(i-1, j) + C$ est un cas de la récurrence, il est inutile de considérer cette option ;

3. $s[i] \neq r[j]$ et $s[p] \neq r[j]$: Nous pouvons supprimer les $(i-p-1)$ variants internes au coût optimal R. Nous avons alors deux possibilités symétriques, muter $s[p]$ en $r[j]$ et muter+contracter $s[i]$, ou bien muter $s[i]$ en $r[j]$ et muter+contracter $s[p]$. Dans les deux cas, cela mène au coût $\mathcal{S} = \mathcal{A}(p-1, j-1) + R + 2M + C$.
– $\mathcal{A}(p, j) \leq \mathcal{A}(p-1, j-1) + M$ (condition de récurrence) ;
– $\mathcal{A}(i-1, j) \leq \mathcal{A}(p, j) + R$ (hypothèse induction).
Ainsi

$$\mathcal{A}(i-1, j) + M + C \leq \mathcal{A}(p-1, j-1) + R + 2M + C = \mathcal{S}$$

Comme $\mathcal{A}(i-1, j) + M + C$ est un terme de l'équation de récurrence, il est inutile de considérer cette option de compression.
Supposons que $s[p] = s[p+1]$ et $s[i] = s[i-1]$, pour compresser la sous-chaîne $s[p..i]$ on pourrait alors contracter $s[p+1]$ en $s[p]$ et $s[i]$ en $s[i-1]$ puis supprimer de manière optimale les $(i-p-1)$ variants internes, cela donne un coût de $\mathcal{A}(p, j) + 2C + R \geq \mathcal{A}(i-1, j) + 2C > \mathcal{A}(i-1, j) + C$. Or $\mathcal{A}(i-1, j) + C$ est un terme de la récurrence, donc nous n'avons pas à considérer cette option même si $s[p] = s[p+1]$ et $s[i] = s[i-1]$.

Comme énoncé, la compression d'une sous-chaîne $s[p..i]$ qui n'est pas une arche donne un coût supérieur ou égal au coût calculé par la récurrence. □

4.3.2 Coût de génération et compression d'arche

Les arches considérées dans la récurrence principale (fig. 4.3, p. 88) sont des séquences de variants dans lesquelles les variants extrémaux sont identiques. Un des variants extrémaux est supposé être l'ancêtre de toute l'arche. Nous notons $A(s)$ le nombre d'arches sur une carte s, si s est de longueur n, $A(s)$ est de l'ordre de $O(n^2)$. Le maximum est atteint lorsque la carte est composée d'un seul et même variant, dans ce cas, il y a $A(s) = \frac{n(n-1)}{2}$ arches ($n-1$ arches commençant à la position 1, $n-2$ arches commençant à la position 2, etc.).

Définition 23 (Arche simple et complexe) *Une arche de longueur $k > 1$ est* **simple** *si elle ne contient pas d'arche interne, et* **complexe** *sinon. En d'autres termes, une arche est simple si chaque variant apparaît seulement une fois sauf celui de l'extrémité qui apparaît deux fois.*

Considérons tout d'abord la compression d'une arche simple de longueur $k > 1$. Il y a deux possibilités de compresser cette arche. Ces possibilités sont montrées à la figure 4.5. L'option (i) mute et contracte le dernier variant, et retire ensuite les $(k-2)$ variants internes au coût optimal R, tandis que l'option (ii) supprime d'abord les $(k-2)$ variants internes au coût optimal R et contracte les variants des extrémités en un seul. Dans les deux cas, nous parvenons au seul variant graine. L'option (i) coûte $R+M+C$ et l'option (ii) $R+C$; ainsi, l'option (ii) est optimale. L'économie obtenue en choisissant la compression d'arche, c'est-à-dire l'option (ii), vaut M.

(i)

Événement	Séquence				
	$s[i-k+1]$	$s[i-k+2]$	$\ldots$	$s[i-1]$	$s[i]$
mutation	$s[i-k+1]$	$s[i-k+2]$	$\ldots$	$s[i-1]$	$s[i-1]$
contraction	$s[i-k+1]$	$s[i-k+2]$	$\ldots$	$s[i-1]$	
suppression	$s[i-k+1]$				

(ii)

Événement	Séquence				
	$s[i-k+1]$	$s[i-k+2]$	$\ldots$	$s[i-1]$	$s[i]$
suppression	$s[i-k+1]$	$s[i]$			
contraction	$s[i-k+1]$				

FIG. 4.5 – *Deux manières de compresser l'arche $s[i-k+1..i]$, avec $i > k$. Notez que $s[i-k+1] = s[i]$ et que l'événement* suppression *ôte tous les variants internes (ceux dans le rectangle) au coût optimal R.*

Considérons maintenant une arche complexe de longueur k. Par définition, elle contient des arches internes. La compression optimale utiliserait l'option (ii) récursivement pour compresser toutes les arches, c'est-à-dire l'arche externe et les arches internes, en allant des plus internes vers l'arche externe. Mais toutes les arches ne peuvent pas être compressées en même temps. En effet, quand deux arches, par exemple u, v, se chevauchent, la graine de u est un variant interne de v et *vice versa*. Il est donc impossible de compresser u et v, puisque cela impliquerait de supprimer la graine de u et la graine de v. La compression optimale d'une arche complexe doit donc maximiser le nombre d'arches compressées. Pour cela nous avons besoin de calculer un ensemble contenant le plus grand nombre d'arches pouvant être compressées ensemble ; nous les appelons compatibles.

Définition 24 (Arches incompatibles) *Deux arches sont* **incompatibles** *si elles se chevauchent par strictement plus d'un variant et que l'une ne contient pas l'autre, ou si elles partagent la même première ou la même dernière position.* **Compatible** *est le contraire de incompatible.*

On peut diviser la relation de compatibilité en deux, *précédence* et *contenance*. Soit deux arches $a = s[i..j]$ et $b = s[k..l]$, avec $1 \leq i < j \leq n$ et $1 \leq k < l \leq n$. On a :

– a précède b si et seulement si $j \leq k$;

– a contient b si et seulement si $i < k < l < j$.

Des exemples d'arches sont montrés à la figure 4.6. Chaque arche est symbolisée par un arc au dessus duquel est placé son nom. Ce nom est composé du variant extrémité de l'arche suivi d'un numéro quand il peut y avoir confusion. Dans l'exemple, les arches a_1, c et d sont simples, tandis que a_2 et a_3 sont complexes. Les paires d'arches suivantes sont incompatibles : (a_1, c), (a_1, a_2), (c, a_3), (a_2, a_3), tandis que (a_2, c), (a_2, d), (a_1, a_3), (a_1, d), (a_3, d) et (c, d) sont compatibles. Pour (a_2, c), (a_2, d) et (a_3, d), la première arche de la paire contient la seconde ; pour (a_1, a_3), (a_1, d) et (c, d), la première arche de la paire précède la seconde.

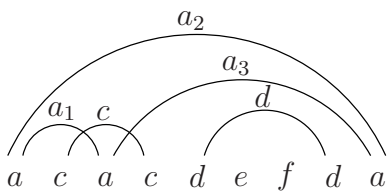


Fig. 4.6 – *Arches simples, complexes, compatibles ou incompatibles.*

En biologie, la notion de compatibilité a un sens. En effet par définition, les variants dans une arche proviennent tous d'un même variant ancêtre. Il est donc logique que deux arches ne peuvent pas se chevaucher, sinon cela voudrait dire que les variants se situant dans la partie commune ont deux ancêtres différents.

La condition « ne sont pas compatibles si elles partagent la même première ou la même dernière position » dans la définition 24 peut s'expliquer en regardant les arches a_1, a_2 et a_3 de la figure 4.6. Avec cette condition, a_2 n'est compatible ni avec a_1, ni avec a_3 tandis que a_1 et a_3 sont compatibles. En effet, ces trois arches ne peuvent pas être compressées en même temps. Par exemple, si nous compressons a_3, alors il reste seulement une arche à compresser, a_1 et a_2 étant devenues identiques. Parmi a_1, a_2 et a_3, il n'y a pas trois arches que l'on peut compresser, mais seulement deux, et nous choisissons celles qui ne partagent pas la même première ou la même dernière position. Cette « incompatibilité artificielle » est nécessaire pour gérer les arches de même nom (c'est-à-dire possédant le même motif à leur extrémité) et se propage bien lorsque leur nombre augmente (par exemple lorsqu'il y a 4 ou 5 caractères a dans une séquence, cela crée respectivement 6 et 10 arches de même nom a).

Lemme 3 Coût de compression d'une arche *Soit a une arche de k variants et p le nombre maximum d'arches internes compatibles parmi toutes les arches, y compris a. Alors, le coût de compression optimal de a est : $(k-1) \times C + (k-1-p) \times M$.*

Preuve (**Lemme 3**) Dans l'arche a, tous les variants sont contractés sauf la graine ; le coût pour les contractions est donc $(k-1) \times C$. Un des variants extrémaux de chaque arche compatible n'est pas muté car il est contracté dans l'autre extrémité. Il y a p arches internes compatibles, donc nous avons besoin de $(k-1-p)$ mutations et leur coût est $(k-1-p) \times M$. Cela donne un coût total de $(k-1) \times C + (k-1-p) \times M$ comme annoncé. □

Nous remarquons ici que sans l'hypothèse selon laquelle le coût des mutations est indépendant de la séquence nucléotidique des variants, le calcul du coût d'une arche s'apparenterait à un problème de parcimonie et serait NP-complet.

Lemme 3 bis Coût de compression d'une arche sous l'hypothèse H2 *Soit a une arche de k variants et p le nombre maximum d'arches internes compatibles parmi toutes les arches, y compris a. Alors, le coût de compression optimal de a est : $p \times C + (k-1-p) \times D$.*

La preuve du lemme 3 *bis* est similaire à celle du lemme 3.

4.3.3 Prétraitement

Nous venons de voir que pour calculer les coûts de compression ou de génération d'une arche a, nous avons besoin de connaître le nombre maximal p d'arches compatibles parmi toute les arches internes à a, y compris a (cf. lemme 3).

Nous souhaitons calculer, en une seule passe, les ensembles maximaux d'arches compatibles pour chaque arche d'une séquence. Ces ensembles sont utilisés lors du calcul des coûts de compression ou de génération des arches. Il est donc intéressant de les connaître avant le calcul de programmation dynamique car d'une part cela évite de recalculer les ensembles maximaux d'une arche à chaque fois que celle-ci est testée, et d'autre part, cela réduit le temps de calcul pour chaque entrée $\mathcal{A}(i,j)$, puisque les coûts de compression et génération d'arche sont déjà calculés.

Le prétraitement consiste donc à calculer et stocker un ensemble maximal d'arches compatibles de toutes les arches de s et r. Pour cela avons deux possibilités :

1. Transformer ce problème en terme de graphe et appliquer un algorithme connu de résolution, en l'occurrence [Apostolico et al., 1992]. Cet algorithme nécessite de décaler les segments correspondant aux arches ;
2. S'inspirer de cet algorithme et travailler sans décalage en gérant les extrémités communes selon la relation de compatibilité entre arches.

La suite de cette section est consacrée à l'étude de chacune de ces possibilités.

4.3.3.1 Prétraitement à base de graphe

Dans cette section, nous expliquons comment, grâce à un algorithme de graphe, nous pouvons obtenir le nombre maximal d'arches deux à deux compatibles pour toutes les arches d'une séquence.

4.3.3.1.1 Calcul d'un ensemble maximal d'arches compatibles

Comme une sous-chaîne, une arche est associée à un intervalle d'indices. La relation d'incompatibilité entre arches définit une relation de chevauchement entre ces intervalles comme vu au chapitre 2, section 2.3.3.

Soit $\mathcal{G} = (V, E)$ le graphe de chevauchement des intervalles associés aux arches de s. Une arête relie deux sommets si les arches associées sont incompatibles. Le problème de trouver le nombre maximal d'arches compatibles est équivalent à trouver un stable max de $\mathcal{G}$. Un exemple de transformation d'un problème en l'autre est donné à la figure 4.7. Chaque arche est associée à un intervalle lui-même associé à un sommet du graphe. Trouver un stable max du graphe de chevauchement nous donne un ensemble maximal d'arches compatibles sur la séquence. Dans l'exemple, il y a deux stables max dans le graphe, $\{a_1, a_3, d\}$ et $\{a_2, c, d\}$.

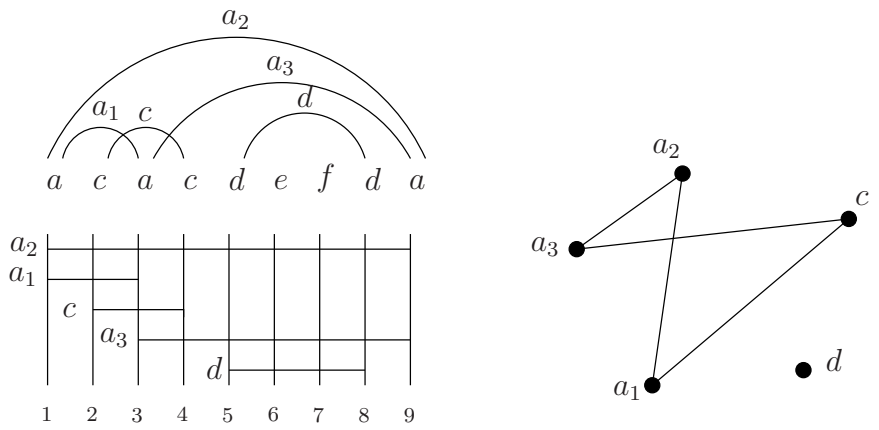


Fig. 4.7 – *Transformation du problème de l'ensemble maximal d'arches compatibles en problème de stable max dans un graphe de chevauchement.*

La transformation directe des arches en intervalles fait que certains de ces intervalles partagent la même extrémité et ceci pose un problème lorsque l'on veut utiliser l'algorithme de [Apostolico et al., 1992] (voir chapitre 3, page 62). En effet, dans ce dernier, il faut construire un codage α de longueur $2k$, où k est le nombre d'arches/intervalles, dans lequel chaque position du codage est soit le début, soit la fin d'un unique intervalle. En modifiant légèrement les intervalles associés aux arches, nous pouvons forcer deux intervalles à ne pas partager les mêmes extrémités, tout en conservant la relation de chevauchement/incompatibilité et ainsi construire un codage α. Nous expliquons la méthode de construction du codage α dans la section suivante (section 4.3.3.1.2).

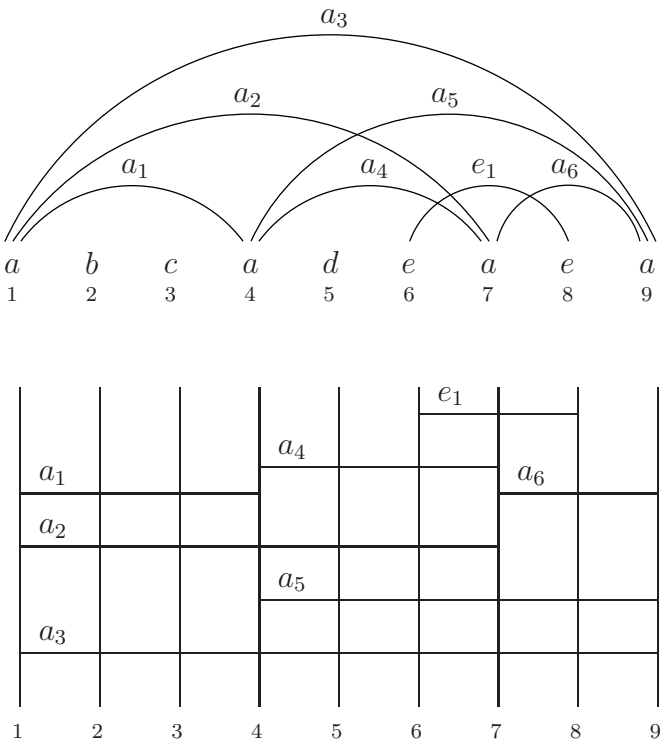


FIG. 4.8 – *Transformation directe des arches en intervalles.*

Le nombre d'intervalles est de l'ordre de $O(k^2)$ pour une arche de longueur k. Apostolico et ses collaborateurs [Apostolico et al., 1992] donnent un algorithme en $O(|V|^2)$ (voir section 3.1.1) pour trouver un stable max dans un graphe de chevauchement. En fait, ils donnent une meilleure complexité qui dépend de la densité de la famille d'intervalles mais dans notre cas, comme la densité peut être égale à $|V|$, ce n'est pas intéressant. Cet algorithme calcule aussi pour tous les intervalles $\mathcal{I}$, le stable max du graphe $\mathcal{G}$ induit par l'ensemble des intervalles inclus dans $\mathcal{I}$ plus $\mathcal{I}$ lui-même. On peut alors avoir le nombre maximal d'arches compatibles pour toutes les arches de la séquence en une seule passe.

Ainsi, en appliquant une seule fois l'algorithme de Apostolico sur les séquences s et r, et en stockant les résultats, nous disposons du nombre maximal d'arches compatibles pour toutes les arches que nous considérons dans l'algorithme d'alignement.

4.3.3.1.2 Construction du codage α

Nous construisons un codage α de l'ensemble des arches d'une séquence s pour pouvoir utiliser l'algorithme polynômial de Apostolico et ses collaborateurs. Cet algorithme nous permet d'obtenir pour chaque arche a de la séquence, un ensemble maximal d'arches compatibles parmi les arches incluses dans a. Le cardinal de cet ensemble est utilisé pour calculer le coût de génération ou de compression de a.

Les arches de s représentent des intervalles qui peuvent partager les mêmes extrémités. Le principe de notre construction du codage α est de « décaler » les intervalles partageant une même extrémité, de manière à ce qu'ils se chevauchent si les arches qu'ils représentent sont incompatibles, et au contraire de manière à ce que leurs extrémités soient distinctes sinon. L'application de notre méthode conduit à la construction d'un codage α pour s. Nous représentons le codage α dans une liste nommée α, sa longueur dépend du nombre d'arches présents dans s, $|\alpha| = 2 \times A(s)$[1].

Nous expliquons maintenant étape par étape le procédé de construction du codage α. Le principe est de relever l'emplacement de chaque variant dans la séquence, puis de créer les arches présentes sur s et enfin construire le codage α pour s. Les algorithmes que nous donnons ne sont pas optimisés par souci de lisibilité. Il faut noter que les ajouts dans une liste se font en fin de liste. Nous montrons l'application de ce procédé sur une séquence $s = abcadeaea$. Les arches de s, ainsi que leur transformation directe en intervalles sont montrées à la figure 4.8.

La méthode de construction du codage α se déroule en trois étapes. Les algorithmes 5, 6 et 7 décrivent séparément ces étapes. La 1re étape consiste à parcourir la séquence afin de relever l'emplacement de chaque variant.

Dans l'algorithme 5, v est le variant en cours de lecture, *app* est un booléen permettant de tester l'appartenance de v à L et L est une liste d'objets, où chaque objet contient un variant de la séquence ($L.variant$) et la liste des positions auxquelles ils apparaît ($L.listepos$). Nous appelons cette liste, liste des positions. La figure 4.9 donne la liste L après l'application de la 1re étape sur la séquence s (fig. 4.8).

Propriété 1 *$\forall v \in s$, $\exists i$ tel que $L[i].variant = v$ et $L[i].listepos$ contient les positions des occurrences de v dans s par ordre* croissant.

Algorithme 5: 1re étape - *Parcours de la séquence s de longueur n.*

```
pour i = 1 à n faire
    v ← s[i] ;
    app ← faux ;
    m ← longueur(L) ;
    pour j = 1 à m faire
        si v = L[j].variant alors
            L[j].listepos ← i ;
            app ← vrai ;
    si app = faux alors
        L[m + 1].variant ← v ; /* Création d'un nouvel élément */
        L[m + 1].listepos ← i ;
```

[1]Nous rappelons que $A(s)$ est le nombre d'arches sur la séquence s.

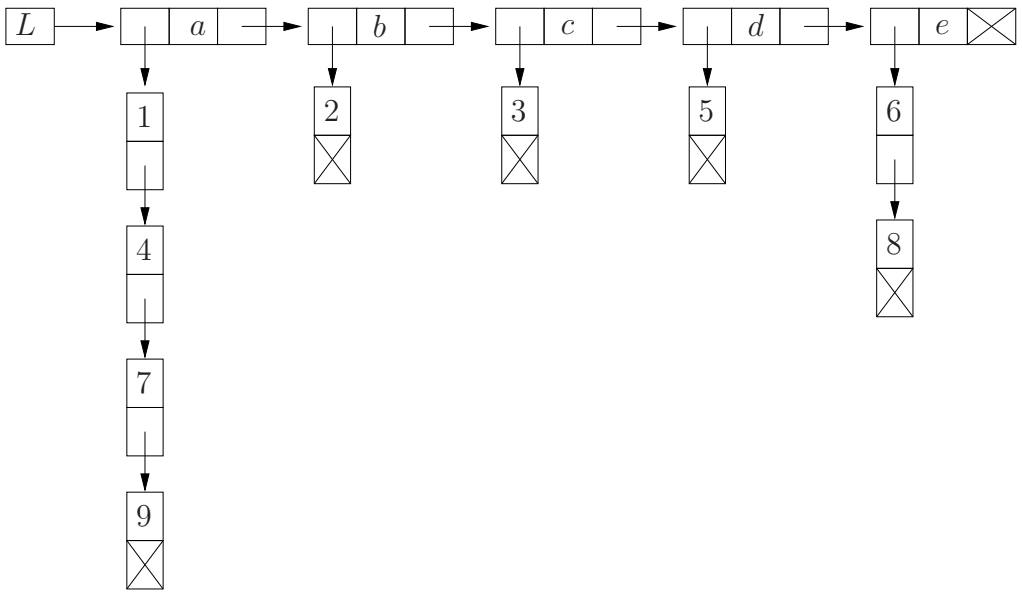


FIG. 4.9 – *Liste des positions associée à la séquence de la figure 4.8.*

La 2e étape consiste à parcourir la liste des positions pour recenser les arches existantes sur la séquence.

Algorithme 6: 2e étape - *Recensement des arches.*

```
pour i = 1 à longueur(L) faire
    v ← L[i].variant;
    si longueur(L[i].listepos) ≥ 2 alors
        l ← L[i].listepos ; /* l est la liste des positions auxquelles v apparaît */
        c ← 1 ;
        pour j = 1 à longueur(l) − 1 faire
            pour k = j + 1 à longueur(l) faire
                /* s[l[j]..l[k]] est une arche */
1               P[l[j]] ← v_c ;
2               P[l[k]] ← v_c ;
                c ← c + 1 ;
```

L'algorithme 6 parcourt la liste des positions L. Pour chaque variant v de L apparaissant au moins deux fois dans la séquence ($longueur(L[v]) \geq 2$), nous parcourons la liste l des positions auxquelles ce variant apparaît. Si v apparaît k fois, cela va créer $\frac{k(k-1)}{2} = p$ arches, que l'on nomme v_1, v_2, ..., v_p. P est un tableau de listes, où chaque ligne correspond à une position de s et contient les arches qui y ont une extrémité. Les lignes numérotées 1 et 2 de l'algorithme 6 remplissent P : à la ligne 1 on ajoute l'extrémité gauche ou début

de l'arche, à la ligne 2, l'extrémité droite ou fin de l'arche.

Le schéma ci-dessous montre le tableau P après le parcours de la liste des positions de la figure 4.9 :

	P		
1	a_1	a_2	a_3
2	$\varnothing$		
3	$\varnothing$		
4	a_1	a_4	a_5
5	$\varnothing$		
6	e_1		
7	a_2	a_4	a_6
8	e_1		
9	a_3	a_5	a_6

Pour toute arche a, nous notons g_a la position gauche de a dans s, et d_a, sa position droite. Par définition des arches, $g_a < d_a$. Chaque arche apparaît deux fois dans α et deux fois dans P, cela correspond à ses deux extrémités. Par commodité, nous distinguons la première occurrence d'une arche a dans α et dans P de la deuxième en les nommant a' et a''.

Propriété 2 *Pour toute arche a, a' est dans la ligne $P[g_a]$ et a'' dans la ligne $P[d_a]$.*

Les propriétés 3, 4 et 5 concernent l'ordre de rangement des positions dans P.

Propriété 3 *Pour toutes arches a et b telles que $g_a = g_b$, a' est avant b' dans $P[g_a]$ si et seulement si $d_a < d_b$. Autrement dit, parmi les arches qui commencent à la même position, les plus courtes sont rangées avant les plus longues.*

Preuve (**Propriété 3**) La liste des positions est ordonnée de manière croissante (prop. 1). Lorsque l'algorithme 6 parcourt cette liste, il crée, pour chaque position p tel que $p = l[j]$, les arches $s[p..l[k]]$ où $l[k]$ varie de manière croissante. Les arches commençant à la position p sont donc rangées dans P de la plus courte à la plus longue. □

Propriété 4 *Pour toutes arches a et b telles que $d_a = d_b$, a'' est avant b'' dans $P[d_a]$ si et seulement si $g_a < g_b$. Autrement dit, parmi les arches qui finissent à la même position, les plus longues sont rangées avant les plus courtes.*

Preuve (**Propriété 4**) La liste des positions est ordonnée de manière croissante (prop. 1). Lorsque l'algorithme 6 parcourt cette liste, il crée les arches $s[l[j]..p]$ dans l'ordre croissant des $l[j]$. Les arches finissant à une position p sont donc rangées dans P de la plus longue à la plus courte. □

Pour chaque ligne i de P, nous notons les positions stockées de manière ordonnée dans cette ligne $i_1, i_2, \ldots, i_{k_i}$, où $k_i \leq A(s)$.

Propriété 5 *Pour tout $i \in [1..n]$, il existe l, $0 \leq l \leq k_i$, tel que les positions $\{i_1, \ldots, i_l\}$ correspondent à des extrémités droites d'arches, et les positions $\{i_{l+1}, \ldots, i_{k_i}\}$ à des extrémités gauches. Si $l = 0$, les positions $\{i_1, \ldots, i_{k_i}\}$ correspondent toutes à des extrémités gauches ; si $l = k_i$, les positions $\{i_1, \ldots, i_{k_i}\}$ correspondent toutes à des extrémités droites.*

Autrement dit, les extrémités droites sont rangées avant les extrémités gauches dans les lignes de P.

Preuve (**Propriété 5**) Considérons une position i, la ligne $P[i]$ contient les extrémités d'arches de motifs $s[i]$ commençant ou terminant à cette position. Ces arches vont être rangées dans P par l'algorithme 6 lors de l'examen de $L[j].listepos$, où $L[j].variant = s[i]$ est le variant extrémité de ces arches. $L[j].listepos$ comprend au moins deux positions dont i, ces positions sont examinées dans l'ordre de la liste, c'est-à-dire dans l'ordre croissant. La double boucle **pour** de l'algorithme 6 range dans P les arches $s[p..i]$, avec $p < i$ avant les arches $s[i..q]$, avec $i < q$. Donc les extrémités droites sont rangées avant les extrémités gauche dans P, ce qui démontre la propriété. □

La troisième et dernière étape permet de créer le codage α de la séquence sous la forme d'une liste.

Algorithme 7: 3e étape - *Création du codage α.*

```
pour i = 1 à n faire
    l ← P[i] ;
    si l ≠ ∅ alors
        pour j = 1 à longueur(l) faire
            α ← l[j] ;
```

L'algorithme 7 parcourt le tableau P et crée le codage α. Le codage α associé à l'exemple de la figure 4.8 est donné à la figure 4.10, sur cette figure on voit également les intervalles correspondant à chaque arche. Ces intervalles n'ont plus d'extrémités communes et leur relation de chevauchement respecte la relation d'incompatibilité entre arches.

L'ordre des arches dans P donne directement l'ordre du codage α. Ceci est dû à notre manière de lire la liste des positions et de remplir le tableau P :

- lorsque plusieurs arches commencent à une même position, elles sont incompatibles. Comme les arches sont rangées dans le tableau de la plus courte à la plus longue (prop. 3), les intervalles correspondant seront « décalés » de manière à se chevaucher. C'est le cas par exemple des arches a_1, a_2 et a_3, ou a_4 et a_5 ;
- de même, pour les arches finissant à la même position, elles aussi incompatibles, elles sont rangées de la plus longue à la plus courte (prop. 4), donc les intervalles correspondant seront également décalés de manière à se chevaucher C'est le cas des arches a_2 et a_4 à la position 7, ou des arches a_3, a_5 et a_6 à la dernière position ;
- pour les positions, comme la position 4 ou la 7, les arches qui finissent sont compatibles avec les arches qui commencent, comme toutes les « fins » sont rangées avant les

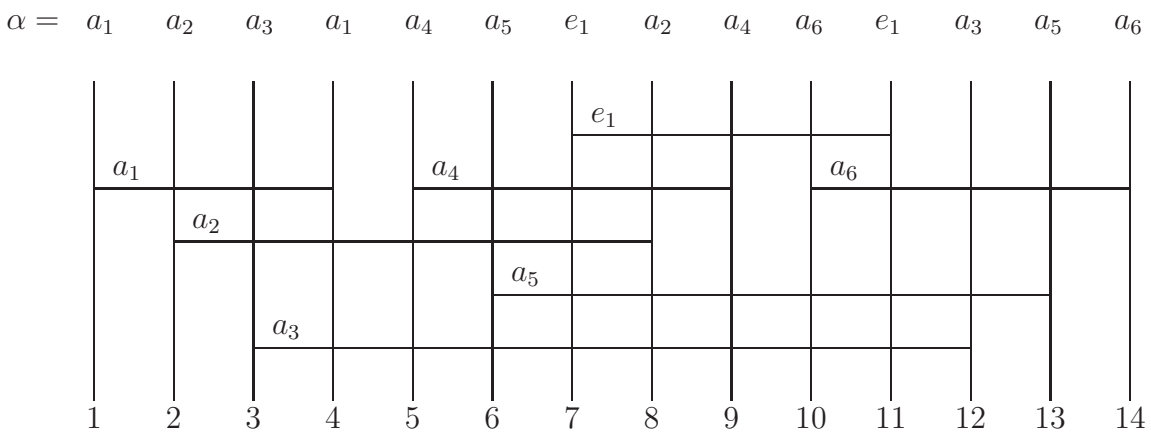


Fig. 4.10 – *Décalage des extrémités pour la construction du codage α.*

« débuts » (prop. 5), les intervalles correspondants vont donc être décalés de manière à ne pas se chevaucher.

Nous montrons maintenant formellement que le codage α de s que nous construisons est correct.

Théorème 4 Correction du codage α *Soient a et b deux intervalles différents, alors a et b se chevauchent dans le codage α obtenu après l'application des algorithmes 5, 6 et 7 sur la séquence s, si et seulement si a et b sont associés à des arches incompatibles de s.*

Preuve (**Correction du codage** α) Pour prouver l'équivalence, nous prouvons les deux implications.

1. Supposons que a chevauche b dans α. Nous supposons sans perte de généralité, que a commence avant b. Nous avons donc, $a' < b' < a'' < b''$ dans α et dans P. Il y a plusieurs possibilités :
   - a' et b' sont sur la même ligne i de P, alors a'' est sur une ligne $j > i$ (prop. 2), et b'' est sur une ligne $k > j$ (prop. 2 et $a'' < b''$). Dans ce cas, $i = g_a = g_b$, les arches a et b commencent à la même position et sont donc incompatibles ;
   - b' et a'' sont sur la même ligne i de P, alors a' est sur une ligne $j < i$ et b'' sur une ligne $k > i$ (prop. 2). Cette configuration est impossible car b' est une extrémité gauche et ne peut pas être rangée avant une extrémité droite (a'') sur la même ligne de P, comme montré par la propriété 5 ;
   - a'' et b'' sont sur la même ligne i de P, alors b' est sur une ligne $j < i$ et a' sur une ligne $k < j$ (prop. 2 et $a' < b'$). Dans ce cas, $i = d_a = d_b$, les arches a et b finissent à la même position et sont donc incompatibles ;

– chaque position est sur une ligne différente de P, cela implique que $g_a < g_b < d_a < d_b$, les intervalles se chevauchent, les arches associées sont donc incompatibles.

Ce sont les quatre seules possibilités de placement de a', a'', b' et b'' dans P, en effet a' et a'' ne peuvent pas se trouver sur la même ligne par définition des arches, de même que b' et b''. Chacune de ces possibilités montre que les arches associées à a et b sont incompatibles, ce qui prouve la première implication.

2. Supposons que les arches a et b soient incompatibles dans s. Nous supposons sans perte de généralité que a est plus courte que b, c'est-à-dire $d_a - g_a < d_b - g_b$. Nous avons trois possibilités :
   – les arches a et b se chevauchent, on a alors $g_a < g_b < d_a < d_b$, par conséquent a', b', a'', et b'' se retrouvent dans cet ordre sur des lignes différentes de P et les intervalles associés se chevauchent dans α ;
   – les arches a et b commencent à la même position, on a alors $g_a = g_b < d_a < d_b$. a est plus courte que b, donc $P[g_a]$ reçoit a' avant b' (prop. 3), par conséquent on a $a' < b' < a'' < b''$ dans P et dans le codage α ;
   – les arches a et b finissent à la même position, on a alors $g_b < g_a < d_a = d_b$. b est plus longue que a, donc $P[d_a]$ reçoit b'' avant a'' (prop. 4), par conséquent on a $b' < a' < b'' < a''$ dans P et dans le codage α.

   Pour chacune de ces possibilités, les intervalles associés aux arches a et b se chevauchent dans α, cela prouve la deuxième implication.

Nous avons montré les deux sens de l'équivalence, cela termine la preuve. □

4.3.3.2 Prétraitement sans décalage

Nous présentons dans l'algorithme 8, page 104, une méthode pour calculer un ensemble maximal d'arches deux à deux compatibles pour chacun des intervalles possibles (même ceux qui ne correspondent pas à des arches). Cet algorithme, nommé *eim* pour « ensembles indépendants maximaux », est une adaptation de celui de Apostolico et ses collaborateurs et s'applique sans décaler les intervalles correspondant aux arches. Nous appelons *stable max d'un facteur* le plus grand ensemble d'arches deux à deux compatibles dont les extrémités se situent sur ce facteur.

L'algorithme 8 prend en entrée la séquence s de longueur n sur laquelle chaque position est associée aux débuts et fins d'arches qui s'y trouvent. L'algorithme 8 calcule une matrice nommée S dont chaque entrée (i, j) est un stable max de $s[i..j]$. Seule la moitié supérieure droite de cette matrice est utilisée ; cela correspond aux entrées (i, j) telles que $j \geq i$. La diagonale de la matrice contient $\varnothing$; ce qui correspond aux stables max des intervalles $s[i..i]$ pour tout i.

Le principe de l'algorithme 8 est de calculer les stables max pour des facteurs de plus en plus grands. Nous remplissons donc la matrice S diagonale par diagonale. La variable t de l'algorithme 8 sert à identifier la diagonale que l'on est en train de remplir : t est le numéro de la colonne de la première case de la diagonale que l'on remplit. La fonction max

utilisée dans cet algorithme retourne l'ensemble de cardinal maximal parmi les ensembles passés en paramètres.

Algorithme 8: *Algorithme des eim.*

Données : Une séquence s de longueur n.

Résultat : Une matrice S contenant les stables max de tous les $s[i..j]$.

```
1 pour i = 1 à n faire
2     pour j = 1 à n faire
3         S(i, j) ← ∅ ;
4 pour t = 2 à n faire
5     pour i = 1 à n − t + 1 faire
6         j ← i + t − 1 ;
7         S(i, j) ← max { S(i + 1, j)
                        { max_{k∈[i+1,j−1]} (S(i, k) ∪ S(k, j))
                        { S(i + 1, j − 1) ∪ {a} si a = s[i..j] est une arche
```

Théorème 5 (Correction de l'algorithme 8) *À la suite de l'application de l'algorithme 8 sur une séquence s de longueur n, l'entrée (i, j) de la matrice S contient un stable max de $s[i..j]$ pour tout $1 \leq i \leq j \leq n$.*

Preuve (**Correction de l'algorithme 8**) Nous allons montrer par induction que l'algorithme 8 est correct.

Supposons que la matrice S soit remplie de manière optimale pour tous les facteurs de s de longueur inférieure à $(j - i + 1)$ et montrons que l'algorithme 8 remplit correctement les entrées $S(i, j)$ pour les facteurs de longueur $(j - i + 1)$. Montrons par l'absurde qu'il ne peut pas exister pour $s[i..j]$ un stable max $\mathcal{S} = \{a_1, a_2, \ldots, a_k\}$ tel que $k > m$ où m est le cardinal du résultat du max de la ligne **7**, c'est-à-dire $m = |S(i, j)|$. Les arches $\{a_1, a_2, \ldots, a_k\}$ sont ordonnées par ordre croissant de leurs positions gauches.

En considérant l'arche a_1, trois cas sont possibles :

1. Soit $g_{a_1} \neq i$: alors $\mathcal{S}$ est également un stable de $s[i + 1..j]$ donc par hypothèse d'induction $|S(i + 1, j)| \geq |\mathcal{S}| = k$ et par conséquent $m \geq k$, ce qui contredit $k > m$;
2. Soit $g_{a_1} = i$ et $d_{a_1} = k$ avec $k \in [i + 1, j - 1]$: on peut alors scinder $\mathcal{S}$ en deux ensembles $\mathcal{S}_1 = \{a_1, \ldots, a_l\}$ avec $g_{a_h} < k, \forall h \in [1..l]$ et $\mathcal{S}_2 = \{a_{l+1}, \ldots, a_k\}$ où $\mathcal{S}_1$ est un stable de $s[i..k]$ et $\mathcal{S}_2$ un stable de $s[k..j]$. Les arches de $\mathcal{S}_1$ sont les arches internes à a_1 plus a_1 elle-même. Nous avons par hypothèse d'induction $|S(i, k) \cup S(k, j)| \geq k$ d'où $m \geq k$, ce qui contredit $k > m$;

   Remarque : Étant donné que des stables de $s[i..k]$ et $s[k..j]$ ne peuvent pas avoir d'arches communes $|S(i, k) \cup S(k, j)| = |S(i, k)| + |S(k, j)|$.

3. Soit $g_{a_1} = i$ et $d_{a_1} = j$: dans ce cas $\mathcal{S}' = \{a_2, \ldots, a_k\}$ est un stable de $s[i+1..j-1]$ et par hypothèse d'induction $|S(i+1, j-1)| + 1 \geq |\mathcal{S}'| + 1 = k$ d'où $m \geq k$, ce qui contredit $k > m$.

Nous venons de montrer que si nous connaissons un stable max pour tous les facteurs de taille inférieure à $(j - i + 1)$ l'algorithme 8 calcule correctement un stable max pour les facteurs de longueur $(j - i + 1)$. Pour terminer la preuve nous devons montrer que nous calculons correctement les stables max des facteurs de longueur 1 : le stable max de ces facteurs est $\varnothing$ puisqu'ils ne contiennent aucune arche ; les valeurs $S(i, i)$ pour tout i sont bien remplies avec $\varnothing$ lors de l'initialisation (lignes **1-3**). □

La complexité en temps de l'algorithme 8 est en $O(n^3)$ où n est la longueur de s. Cette complexité provient des deux boucles **pour** (lignes **4** et **5**) qui remplissent $\frac{n(n-1)}{2} = O(n^2)$ cases de S et du calcul de $\max_{k \in [i+1, j-1]} (S(i, k) \cup S(k, j))$ (ligne **7**) qui est en $O(n)$.

En appliquant cet algorithme sur la carte s, nous stockons dans une matrice de taille n^2 les stables max pour chaque sous-chaîne de s. Nous avons besoin de la matrice similaire pour les générations d'arche dans r. Cette procédure constitue le prétraitement.

* *

*

Nous avons montré deux manières différentes d'obtenir les ensembles d'arches deux à deux compatibles pour toutes les arches d'une séquence s. La complexité en temps dans le pire des cas pour la méthode utilisant l'algorithme de Apostolico est en $O(|V|^2) = O(n^4)$ alors que la seconde méthode est en $O(n^3)$, où n est la longueur de s. On remarque que la complexité de la première méthode est dépendante du nombre d'arches sur la séquence, en effet $|V| = A(s)$. Cette méthode est donc intéressante lorsque le nombre d'arches est faible. Cependant, les séquences biologiques sur lesquelles nous avons appliqué notre algorithme admettent un grand nombre d'arches, aussi nous avons adopté la seconde méthode comme prétraitement.

Ensemble des arches à considérer. Soit $1 < i < n$ et $1 < j < m$. Nous calculons l'entrée $\mathcal{A}(i, j)$. L'ensemble des positions de départ d'une arche de s terminant à la position i est $\mathcal{C}_i = \{l \mid 0 < l < i \text{ et } s[l] = s[i]\}$. Cet ensemble est indépendant de j et ainsi, les arches devant être évaluées sont les mêmes pour toutes les entrées de la ligne i de la matrice. Cela signifie qu'il y a au plus $(i - 1)$ arches possibles à considérer pour calculer les entrées de la ligne i de $\mathcal{A}$. Donc, il y a $O(n^2)$ compressions d'arche et symétriquement $O(m^2)$ générations d'arches pour toute la matrice $\mathcal{A}$.

Comme nous le montrons ci-dessus, l'ensemble d'arches dont les compressions, resp. les générations, sont considérées dans la récurrence de la matrice de programmation dynamique dépend seulement de s, resp. de r. Donc, en utilisant l'algorithme 8 nous pouvons pré-calculer pour s et r un ensemble maximal d'arches compatibles pour chacune de leurs arches. Par conséquent, le prétraitement nous permet de calculer $\mathcal{A}(i, j)$ en $O(m + n)$. Nous stockons les résultats dans une table avec n^2 entrées pour s et une table avec m^2 entrées pour r.

4.3.4 Complexité de l'algorithme

Nous nous intéressons maintenant à la complexité globale de l'algorithme.

Théorème 6 Complexité de l'algorithme *Notre algorithme nécessite $O(p^3)$ temps et $O(p^3)$ espace, où $p = \max(n, m)$.*

Preuve (**Complexité de l'algorithme**)

Matrices des ensembles de stables max : Pour la carte s, nous calculons une matrice S dont l'entrée sur la $i^{\text{ème}}$ ligne et la $j^{\text{ème}}$ colonne contient un ensemble maximal d'arches compatibles de la sous-chaîne $s[i..j]$. Nous avons besoin d'une matrice similaire pour l'autre carte r. Un élément de la matrice stocke un ensemble maximal d'arches compatibles qui contient au plus n arches à cause de la définition de la relation de compatibilité. Comme il y a au plus $O(n^2)$ arches dans s et $O(m^2)$ arches dans r, ces matrices pour les cartes s et r ont une complexité en espace de $O(n^3 + m^3)$. Appliquer l'algorithme 8 pour calculer la matrice S sur la séquence s de longueur n requiert $O(n^3)$ en temps.

Calcul de programmation dynamique : La matrice $\mathcal{A}$ occupe $O(n \times m)$ unités de mémoire. Chaque entrée dépend des trois entrées voisines et d'au plus toutes les entrées précédentes sur la même ligne et sur la même colonne. Le calcul de chaque dépendance peut être effectué en temps $O(1)$. En calculant $\mathcal{A}(i, j)$, nous avons à considérer au plus $(i-1)$ compressions d'arche et $(j-1)$ générations d'arche. Donc, cela prend pour toute la matrice $\sum_{i=1}^{n} \left(\sum_{j=1}^{m} [(i-1) + (j-1)] \right) = O(mn^2 + nm^2)$.

Ainsi, si $p = \max(n, m)$, la complexité totale est de $O(p^3)$ en temps et $O(p^3)$ en espace. □

4.4 Autres alignements optimaux

La récurrence principale (fig. 4.3, page 88) calcule la distance entre deux cartes en trouvant un alignement optimal. Dans la plupart des cas, plusieurs alignements optimaux existent et il peut être intéressant d'en retrouver plus d'un lorsque l'on s'intéresse à l'évolution des séquences par exemple. Dans la figure 4.11, nous montrons trois exemples typiques dans lesquels il existe des alignements optimaux alternatifs.

Dans l'exemple (1), la contraction d'un variant a (en le a adjacent) qui suit une série de trois appariements exacts peut être effectuée de manière équivalente à toutes les positions. La contraction et l'appariement exact commutent. Dans l'exemple (2), la M_C de f commute avec la mutation à la troisième position. Dans ces deux exemples l'ensemble d'opérations reste le même (trois appariements exacts et une contraction pour le premier, trois mutations et une M_C dans le second), tandis que dans l'exemple (3), l'ensemble change. En effet, l'appariement exact entre les deux d est transformé en une *contraction distante*, et la M_C de c est transformée en une mutation. Une contraction distante signifie que le variant parent n'est pas à la position adjacente. Les deux alignements optimaux correspondent à deux histoires évolutives différentes. La contraction de d doit être effectuée après la M_C

$$\begin{array}{cccc} a & a & a & a \\ | & | & | & / \\ a & a & a & - \end{array} \qquad \begin{array}{cccc} e & c & g & f \\ ] &] &] &) \\ a & b & d & - \end{array} \qquad \begin{array}{cccc} a & c & f & d \\ ] &) &) & | \\ b & - & - & d \end{array}$$

$$\begin{array}{cccc} a & a & a & a \\ | & / & | & | \\ a & - & a & a \end{array} \qquad \begin{array}{cccc} e & c & g & f \\ ] &] &) &] \\ a & b & - & d \end{array} \qquad \begin{array}{cccc} a & c & f & d \\ ] &] &) & / \\ b & d & - & - \end{array}$$

(1) (2) (3)

FIG. 4.11 – *Exemples d'alignements optimaux alternatifs.*

de f et la mutation de c en d. C'est un cas de non-commutativité comme dans les arches. Le cas de l'exemple (3) est une sorte d'arche dans laquelle un des variants extrémaux a disparu de la séquence mais est encore visible dans l'autre séquence. Nous appelons cela *arche fantôme*.

Les histoires incluant des arches fantômes ont un attrait biologique. Le reste de cette section traite de leur calcul : nous donnons les lignes à ajouter à la récurrence de la programmation dynamique pour les prendre en compte et la procédure pour calculer leur coût, enfin, nous montrons comment adapter le prétraitement et discutons de la complexité de l'algorithme résultant.

4.4.1 Arches fantômes

Les arches fantômes sont intéressantes car elles mettent en évidence les amplifications et contractions distantes, et nous permettent de découvrir d'autres histoires évolutives des séquences. Regardons un exemple :

$$s = abcd \longrightarrow abc\mathbf{cc}d \longrightarrow abc\mathbf{e}cd \longrightarrow abecd = r$$

Le variant c, amplifié deux fois dans s, a disparu de r, mais il reste une trace de son amplification. Un alignement possible est :

$$\begin{array}{cccccc} s: & a & b & c & - & d \\ & | & | & [& \backslash & | \\ r: & a & b & e & c & d \end{array}$$

Cet alignement montre que le variant c de r provient d'une amplification. Dans cet alignement, on note que ec est une arche fantôme de graine c. Un autre alignement a le même coût optimal mais perd l'information sur l'origine de c :

$$\begin{array}{cccccc} s: & a & b & - & c & d \\ & | & | & (& | & | \\ r: & a & b & e & c & d \end{array}$$

Les arches fantômes nous permettent de trouver les traces d'anciennes amplifications. Notez que les arches fantômes peuvent être vues seulement lorsque l'on aligne deux séquences, elles ne peuvent pas être détectées dans une séquence seule. Pour différencier les deux classes d'arches, nous qualifions les arches non fantômes de *visibles*. Comparées aux arches visibles, la compression d'une arche fantôme nécessite une mutation supplémentaire.

Remarque : Par rapport aux arches visibles, qui lorsqu'elle sont générées ou compressées permettent une économie d'une mutation, les arches fantômes sont équivalentes en coût à un alignement calculé de manière strictement incrémentale. De ce fait, elles n'apparaissent pas dans l'algorithme 4 de la section 4.3 qui cherche juste à calculer un alignement optimal.

Les arches fantômes sont divisées en deux sous-classes selon l'extrémité à laquelle le variant a été muté, nous les avons nommées type 1 et type 2. Dans les arches fantômes de type 1, le variant de l'extrémité droite correspond à la graine alors que le variant de l'extrémité gauche a été muté. Dans les arches fantômes de type 2, c'est le variant de gauche qui correspond à la graine alors que le variant de droite à été muté. Dans l'exemple ci-dessus, l'arche fantôme *ec* est de type 1 ; dans l'exemple (3) de la figure 4.11, l'arche fantôme cfd est également de type 1.

4.4.2 Récurrence

Pour prendre en compte les arches fantômes en calculant l'alignement, nous ajoutons les cas suivants dans la récurrence donnée page 88 :

$\mathcal{A}(l,j) + K(s[i].s[l+1..i])$	Compression d'arche fantôme (type 1) si $i > 2, j > 0$, et $s[i] = r[j], \forall l \in [1, i-2]$ tel que $s[l] \neq s[i]$
$\mathcal{A}(l,j) + K(s[l..i-1].s[i]) + M$	Compression d'arche fantôme (type 2) si $i > 2, j > 0$, et $s[l] = r[j], \forall l \in [1, i-2]$ tel que $s[l] \neq s[i]$
$\mathcal{A}(i,l') + G(r[j].r[l'+1..j])$	Génération d'arche fantôme (type 1) si $j > 2, i > 0$, et $s[i] = r[j], \forall l' \in [1, j-2]$ tel que $r[l'] \neq r[j]$
$\mathcal{A}(i,l') + G(r[l'..j-1].r[l']) + M$	Génération d'arche fantôme (type 2) si $j > 2, i > 0$, et $s[i] = r[l'], \forall l' \in [1, j-2]$ tel que $r[l'] \neq r[j]$

Une arche fantôme est de longueur 2 au minimum, d'où les conditions $i > 2$ pour les compressions et $j > 2$ pour les générations. Une arche fantôme est alignée avec un variant de l'autre carte, d'où $j > 0$ pour les compressions et $i > 0$ pour les générations. Une arche fantôme de longueur 2 peut être compressée par une M_C ou simplement par une contraction. Symétriquement, les générations d'arche fantôme de longueur 2 sont réductibles à une seule opération élémentaire. Ces situations sont directement prises en compte dans la récurrence initiale (fig. 4.3, page 88). De telles compressions et générations ne sont donc pas considérées

dans ces lignes additionnelles et la position de fin d'une arche fantôme doit être dans $[1, i-2]$ pour les compressions et dans $[1, j-2]$ pour les générations.

Les opérations K et G donnent le coût de compression ou de génération de l'arche visible associée à l'arche fantôme. C'est pourquoi l'extrémité de l'arche fantôme mutée (extrémité gauche pour le type 1, droite pour le type 2) est remplacée par l'extrémité conservée. Pour les arches de type 1, la mutation du variant à l'extrémité gauche est déjà prise en compte dans la case $\mathcal{A}(l, j)$ pour les compressions ou dans $\mathcal{A}(i, l')$ pour les générations. Ce n'est pas le cas pour les arches de type 2, c'est pourquoi les générations et compressions d'arche de type 2 requièrent le terme $+M$.

4.4.3 Calcul des coûts de compression d'arche fantôme

Nous considérons les arches fantômes allant de la position l à i, avec $1 \leq l < i \leq n$. Les deux types d'arches fantômes sont symétriques ; aussi ici et dans le lemme 4, nous considérons seulement les arches fantômes de type 1. Pour le type 1, la séquence de l'arche considérée n'est pas $s[l..i]$, mais $s[i].s[l+1..i]$. En effet, le premier variant est changé en $s[i]$, qui est aussi le variant final. Ce changement crée autant d'arches qu'il y a d'autres occurrences de $s[i]$ dans $s[l+1..i]$. Nous montrons que connaissant un ensemble maximal d'arches correspondant à $s[l+1..i]$ nous pouvons déduire un ensemble maximal correspondant à $s[i].s[l+1..i]$. La sous-chaîne $s[l+1..i]$ n'est pas forcément une arche, nous donnons à la section 4.4.4 un algorithme permettant de calculer un ensemble maximal d'arches compatibles sur des sous-chaînes quelconques.

Lemme 4 Calcul d'un ensemble maximal d'arches fantômes (pour le type 1). *Soit $\mathcal{Y}, \mathcal{Z}$ les ensembles maximaux d'arches compatibles de $s[l+1..i]$ et $s[i].s[l+1..i]$ respectivement ($\mathcal{Y}$ est déjà calculé, $\mathcal{Z}$ est l'ensemble que nous cherchons). Soit $p > 1$ un entier et $k_1 = l < \ldots < k_p = i$ les positions ordonnées du variant $s[i]$ dans $s[i].s[l+1..i]$. Pour chaque h, h' dans $[1, p]$ tel que $h < h'$, nous notons l'arche allant de k_h à $k_{h'}$ par $H_{k_h,k_{h'}}$. Nous avons deux cas possibles illustrés à la figure 4.12 :*

1. *Soit il existe un sous-ensemble maximal $\mathcal{C}$ de positions ordonnées $\{c_1 \ldots c_q\} \subseteq \{k_2 \ldots k_p\}$ tel que*
   - *$1 \leq q < p$;*
   - *$c_q = k_p = i$;*
   - *et $H_{c_j,c_{j+1}} \in \mathcal{Y}, \forall j \in [1 \ldots q-1]$.*

   Alors $\mathcal{Z} = \mathcal{Y} \cup \{H_{k_1,c_1}\}$;
2. *Soit pour tout h tel que $1 < h < p$, $H_{k_h,k_p} \notin \mathcal{Y}$ et alors $\mathcal{Z} = \mathcal{Y} \cup \{H_{k_1,k_p}\}$.*

Notons que les deux cas du lemme sont exclusifs, s'il existe une arche $H_{k_h,k_p} \in \mathcal{Y}$ avec $1 < h < p$, on est dans le cas 1. S'il n'en existe pas, on est dans le cas 2. Avant de prouver le lemme 4, nous avons besoin de la propriété suivante.

Propriété 6 *Parmi les arches qui sont dans $s[i].s[l+1..i]$ et pas dans $s[l+1..i]$, c'est-à-dire parmi les arches de type H_{k_1,k_h} avec $1 < h \leq p$, au plus une peut être ajoutée à $\mathcal{Y}$ pour constituer un ensemble d'arches compatibles sur $s[i].s[l+1..i]$.*

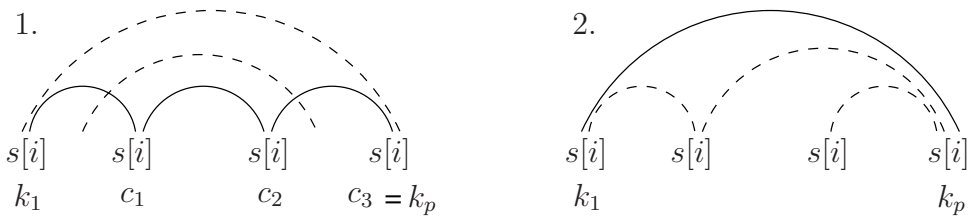


Fig. 4.12 – *Illustration des deux cas du lemme 4. Les arches en trait plein appartiennent aux ensembles maximaux tandis que les arches en pointillé non.*

Preuve (**Propriété 6**) En effet, les arches qui sont dans $s[i].s[l+1..i]$ et pas dans $s[l+1..i]$ commencent toutes à la même position, $k_1 = l$, et sont donc deux à deux incompatibles. C'est pour cela qu'au plus une d'entre elles peut être ajoutée à $\mathcal{Y}$ pour former un ensemble d'arches compatibles. □

Preuve (**Lemme 4**) Dans le premier cas, les arches $H_{c_j,c_{j+1}}$ telles que définies dans le lemme sont dans $\mathcal{Y}$. On est alors sûr que H_{k_1,c_1} est compatible avec toutes les arches de $\mathcal{Y}$. En effet, supposons qu'il existe une arche $a \in \mathcal{Y}$ qui est incompatible avec H_{k_1,c_1}, alors :

1. Soit a commence à la même position que H_{k_1,c_1} ;
2. Soit a chevauche H_{k_1,c_1} ;
3. Soit a finit à la même position que H_{k_1,c_1}.

L'hypothèse 1 est fausse par construction, en effet $a \in \mathcal{Y}$ et ne peut donc pas commencer à la position $k_1 = l$, position de début de H_{k_1,c_1}.

Si a chevauche H_{k_1,c_1}, alors $(g_a < c_1)$ et $(d_a > c_1)$[2], donc a chevauche également une des arches $H_{c_j,c_{j+1}}$ ou finit à la position $k_p = c_q$ et est incompatible avec H_{c_{q-1},c_q}, ce qui contredit l'hypothèse $a \in \mathcal{Y}$. Donc l'hypothèse 2 est également fausse.

Si a finit à la même position que H_{k_1,c_1}, alors son variant extrémité est $s[i]$ or par hypothèse il n'y a pas d'autres positions k_h avec $k_1 < h < c_1$ telle que $H_{k_h,c_1} \in \mathcal{Y}$ sinon $\mathcal{C}$ n'est pas maximal. Par conséquent, il ne peut pas exister d'arche $a \in \mathcal{Y}$ finissant en c_1. L'hypothèse 3 est donc fausse.

Nous venons de montrer qu'il n'existe pas d'arche $a \in \mathcal{Y}$ incompatible avec H_{k_1,c_1}, grâce à la propriété 6 et par la maximalité de $\mathcal{Y}$, on a $|\mathcal{Y}| \leq |\mathcal{Z}| \leq |\mathcal{Y}| + 1$ donc $\mathcal{Z} = \mathcal{Y} \cup \{H_{k_1,c_1}\}$ est un ensemble maximal d'arches compatibles sur $s[i].s[l+1..i]$.

Dans le second cas du lemme, aucune des arches H_{k_h,k_p} pour $h \in]1,p[$ n'est dans $\mathcal{Y}$ et ces arches sont les seules arches incompatibles avec H_{k_1,k_p}. Nous pouvons alors établir, avec les mêmes arguments que pour le cas 1, $\mathcal{Z} = \mathcal{Y} \cup \{H_{k_1,k_p}\}$ est un ensemble maximal d'arches compatibles sur $s[i].s[l+1..i]$. □

[2]Nous rappelons que la notation g_a signifie position gauche de a et d_a position droite de a.

4.4.4 Prétraitement

De la même manière que dans le cas général, nous avons besoin d'un prétraitement permettant de calculer le coût des compressions ou générations d'arche de manière rapide lors de la programmation dynamique.

Comme nous l'avons vu, l'algorithme 8, page 104, calcule un stable max pour tous les intervalles de la séquence, même ceux qui ne correspondent pas à des arches visibles. C'est exactement ce dont nous avons besoin dans notre problème pour calculer les coûts de compressions ou générations des arches fantômes, c'est-à-dire appliquer le lemme 4.

La complexité en temps de l'algorithme 8 est en $O(n^3)$ où n est la longueur de la séquence. En appliquant cet algorithme sur la carte s, nous stockons dans une matrice de taille n^2 les stables max pour chaque sous-chaîne de s. Nous avons besoin de la matrice similaire pour les générations d'arche dans r. Cette procédure constitue le prétraitement.

4.4.5 Complexité

Le prétraitement étendu sur la carte s de longueur n se fait en $O(n^3)$. Il doit également être exécuté sur r de longueur m. La complexité de ce prétraitement est donc en $O(p^3)$, où $p = \max(n, m)$. Dans la matrice de programmation dynamique, le calcul des différentes dépendances peut être effectué en $O(1)$ sauf pour les arches fantômes. Appliquer le lemme 4 pour l'arche fantôme $s[i].s[l{+}1..i]$ nécessite de tester $(i{-}l)$ fois l'appartenance à un ensemble d'arches. En ordonnant cet ensemble d'arches selon leurs extrémités dans la séquence, tester l'appartenance prend $O((i{-}l)\log(i{-}l))$ en temps. Notez qu'une arche fantôme et une arche visible finissant à la même position ne peuvent pas avoir la même position de départ puisque l'arche fantôme n'a pas le même variant à ses extrémités. Il y a au plus $(i-1)$ arches à considérer pour calculer la $i^{\text{ème}}$ ligne de $\mathcal{A}$. Donc cela prend $\sum_{i=1}^{n} i^2 \log i = O(n^3 \log n)$ pour toutes les lignes, et $O(m^3 \log m)$ pour toutes les colonnes. Ce qui donne en tout $O(nm + n^3 \log n + m^3 \log m) = O(max(n, m)^3 \log(max(n, m)) = O(p^3 \log p)$.

La complexité totale est donc en $O(p^3 \log p)$. Prendre en considération des arches fantômes augmente sensiblement la complexité théorique de notre algorithme.

Chapitre 5

Le logiciel MS_Align

Sommaire

5.1 Présentation 113
5.1.1 MS_Align2 114
5.1.2 MS_AlignN 116
5.2 Implémentation 120
5.2.1 Le programme 120
5.2.2 L'interface web 121

Dans ce chapitre, nous présentons le logiciel MS_Align résultant de l'implémentation de notre algorithme d'alignement de cartes de minisatellite, détaillé au chapitre 4. Ce logiciel se décline en deux versions, MS_Align2 et MS_AlignN. La première permet d'aligner deux cartes de minisatellite ; elle donne la représentation de leur alignement et la distance entre les deux cartes. La seconde calcule tous les alignements deux à deux de N cartes de minisatellite et renvoie la matrice de distances associée.

Nous présentons tout d'abord les deux versions du logiciel, ensuite nous évoquons leur implémentation et leur mise à disposition *via* le web.

5.1 Présentation

Les deux versions du logiciel sont accessibles depuis le serveur bioinformatique du Lirmm à l'adresse *http://degas.lirmm.fr/*, en suivant le lien Evolution of Genomic Repeated Sequences. Chacun des deux programmes possède une aide en ligne. Une description de la méthode qu'ils utilisent est également disponible sur le serveur bioinformatique à l'adresse *http://degas.lirmm.fr/REPSEQ/Minisat/index.html*. L'interface graphique de ces programmes comprend un écran de saisie et un écran de sortie.

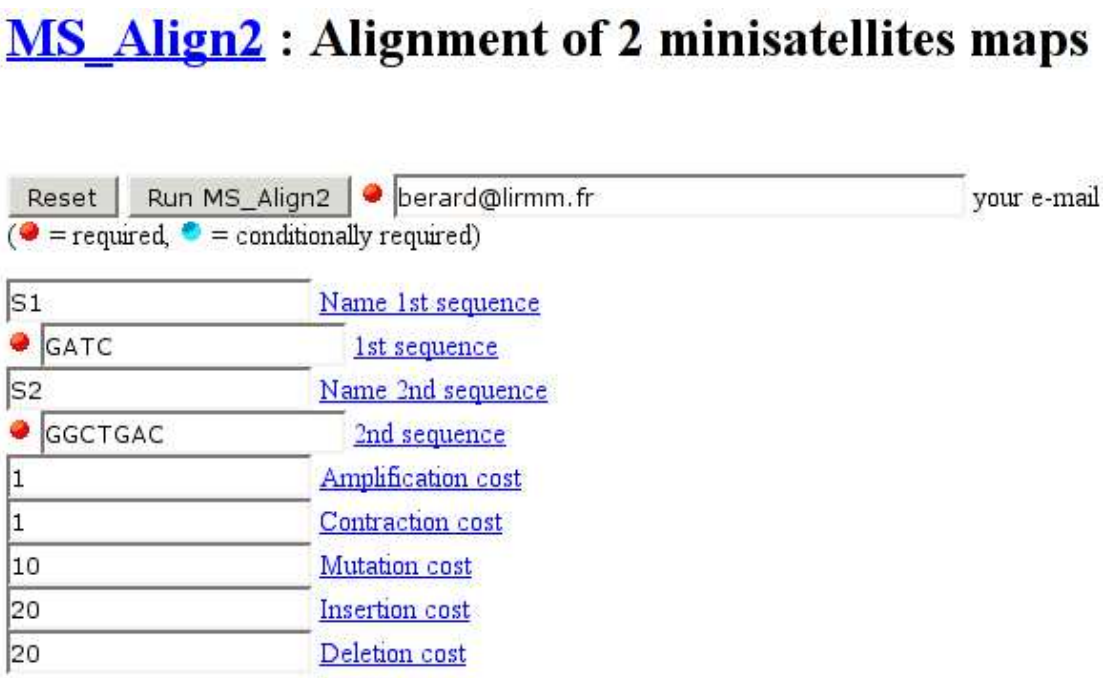


Fig. 5.1 – *Écran d'accueil de* MS_Align2.

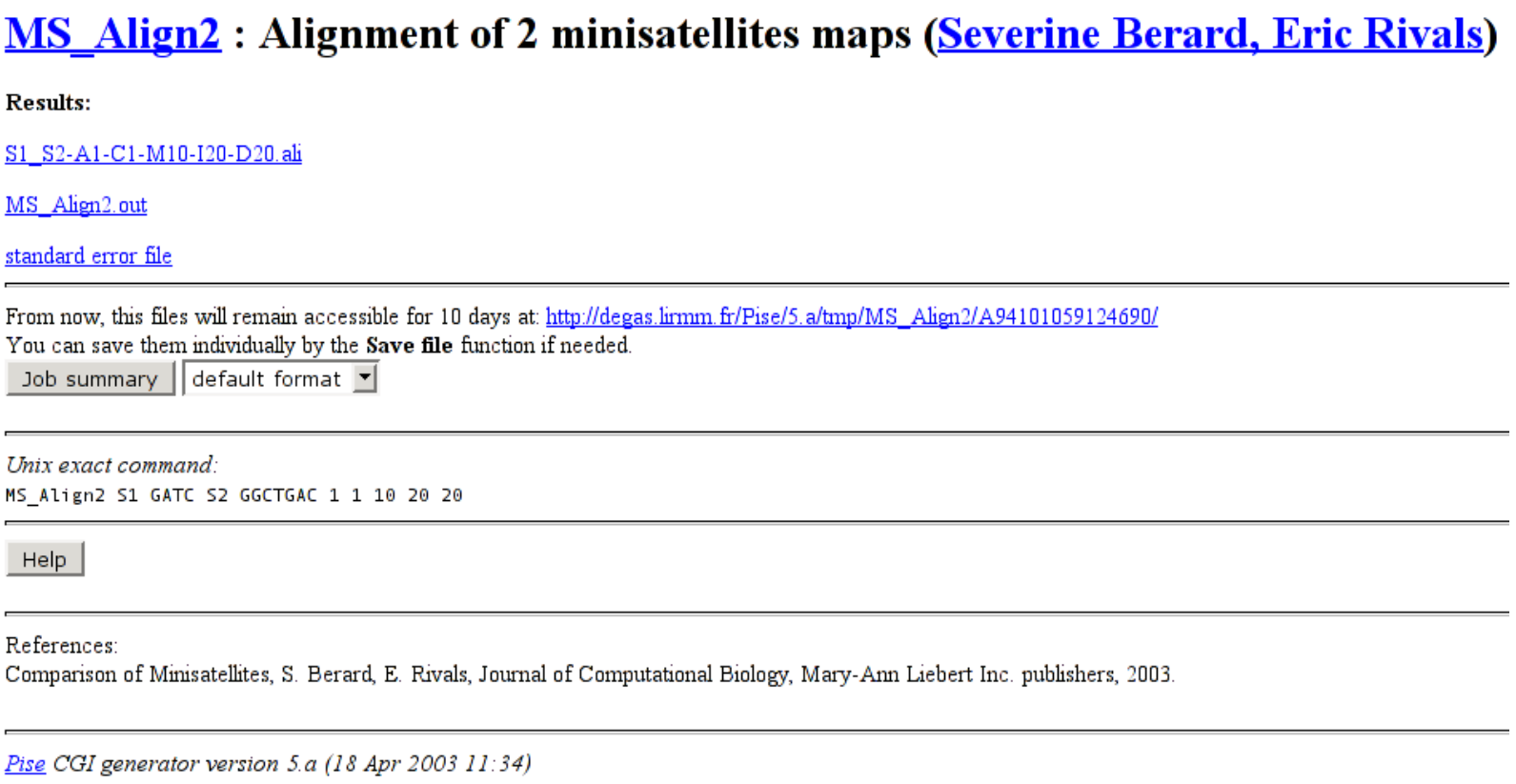
5.1.1 MS_Align2

MS_Align2 prend en entrée deux cartes de minisatellite et les coûts des opérations élémentaires de l'alignement. Il calcule la distance entre ces deux cartes ainsi qu'un alignement correspondant à cette distance. La figure 5.1 présente l'écran d'accueil de MS_Align2. Les cases `nom` des 1re et 2e séquences, ainsi que celles des coûts sont déjà remplies avec leurs valeurs par défaut, respectivement S1, S2, $A = 1$, $C = 1$, $M = 10$, $D = 20$ et $I = 20$. Nous avons ajouté ici le code de deux séquences en prenant le même exemple qu'au chapitre 2 page 47 lors de la présentation de la méthode d'alignement classique. On a donc S1 = $GATC$ et S2 = $GGCTGAC$.

Lorsque l'on lance le programme MS_Align2 en cliquant sur l'icône `Run MS_Align2`, on obtient l'écran de résultat montré à la figure 5.2. Sur cet écran on peut voir trois fichiers résultats :

- `S1_S2-A1-C1-M10-I20-D20.ali`, dont le nom est composé des noms des deux séquences, suivis des coûts de l'alignement, ce fichier contient le résultat du programme, c'est-à-dire l'alignement entre les deux séquences ;
- `MS_Align2.out` contient les affichages de MS_Align2 sur la sortie standard ;
- et `standard error file` contient les affichages de MS_Align2 sur la sortie erreur, s'il y en a.

Dans notre cas, seul le fichier résultat `S1_S2-A1-C1-M10-I20-D20.ali` nous intéresse. Le contenu de ce fichier est donné à la figure 5.3. On y trouve :



MS_Align2 : Alignment of 2 minisatellites maps (Severine Berard, Eric Rivals)

Results:

S1_S2-A1-C1-M10-I20-D20.ali

MS_Align2.out

standard error file

From now, this files will remain accessible for 10 days at: http://degas.lirmm.fr/Pise/5.a/tmp/MS_Align2/A94101059124690/
You can save them individually by the **Save file** function if needed.

Job summary | default format

Unix exact command:

```
MS_Align2 S1 GATC S2 GGCTGAC 1 1 10 20 20
```

Help

References:
Comparison of Minisatellites, S. Berard, E. Rivals, Journal of Computational Biology, Mary-Ann Liebert Inc. publishers, 2003.

Pise CGI generator version 5.a (18 Apr 2003 11:34)

FIG. 5.2 – *Écran de résultat de* MS_ALIGN2.

```
************************************************************
******************** Alignment between S1 and S2 ********************
************************************************************
Costs : Amp = 1    Con = 1    Mut = 10    Ins = 20    Del = 20

Alignment score : 33
Number of mutational events : 6

S1  : G-AT--C
      |\[|((|
S2  : GGCTGAC
************************************************************
```

Fig. 5.3 – *Alignement contenu dans le fichier* `S1_S2-A1-C1-M10-I20-D20.ali`.

- le rappel des coûts utilisés pour calculer l'alignement ;
- le score de cet alignement, ici le score est 33, il correspond à la somme des coûts d'une amplification (de coût 1), d'une mutation (de coût 10), et de deux A_M (chacune de coût 11) ;
- le nombre d'événements mutationnels de l'alignement ; 6 événements dans notre exemple, en effet une A_M compte pour deux ;
- et en dessous, la représentation de l'alignement, cet alignement se lit de la manière suivante : pour transformer $S1$ en $S2$, il faut amplifier le G de $S1$, muter le A en C, amplifier+muter le T en G et amplifier+muter le G obtenu, en A.

On peut se reporter à la figure 4.2, page 86, pour la lecture de l'alignement et la définition des symboles '|', '\', '[' et '('.

5.1.2 MS_AlignN

MS_AlignN calcule les distances deux à deux d'un jeu de cartes de minisatellite, il ne donne pas les alignements correspondant à ces distances. La figure 5.4 présente l'écran d'accueil de MS_AlignN. MS_AlignN prend en entrée les coûts pour le calcul de l'alignement et un ensemble de séquences données dans un format de type FASTA, c'est-à-dire que chaque séquence doit être encodée sur deux lignes, la première commence par le symbole '>' et donne le nom et/ou la description de la séquence, la seconde ligne donne le code de la séquence[1]. Un exemple de fichier d'entrée correspondant à ce format est donné à la figure 5.6, page 119 en haut. Cet ensemble de séquences peut être donné soit sous la forme d'un fichier, soit directement entré dans le cadre réservé à cet effet. Les coûts par défaut sont les mêmes que pour MS_Align2, $A = C = 1$, $M = 10$ et $D = I = 20$.

Lorsque l'on lance le programme MS_AlignN en cliquant sur l'icône `Run MS_AlignN`, on obtient l'écran de résultat montré à la figure 5.5. Sur cet écran, on peut voir trois fichiers

[1]Dans le « vrai » format FASTA, il est possible de coder la séquence sur plusieurs lignes.

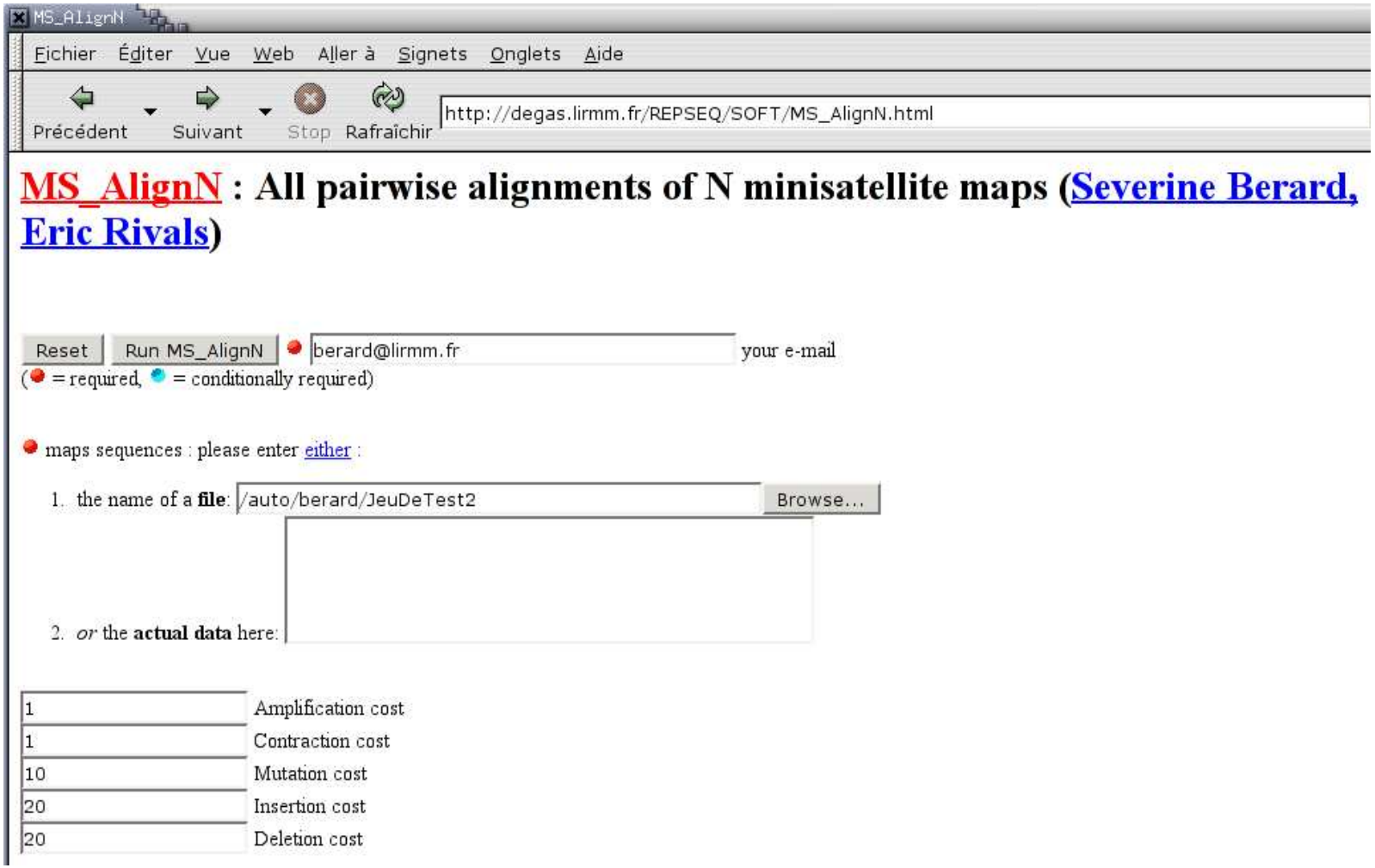


FIG. 5.4 – *Écran d'accueil de* MS_ALIGNN.

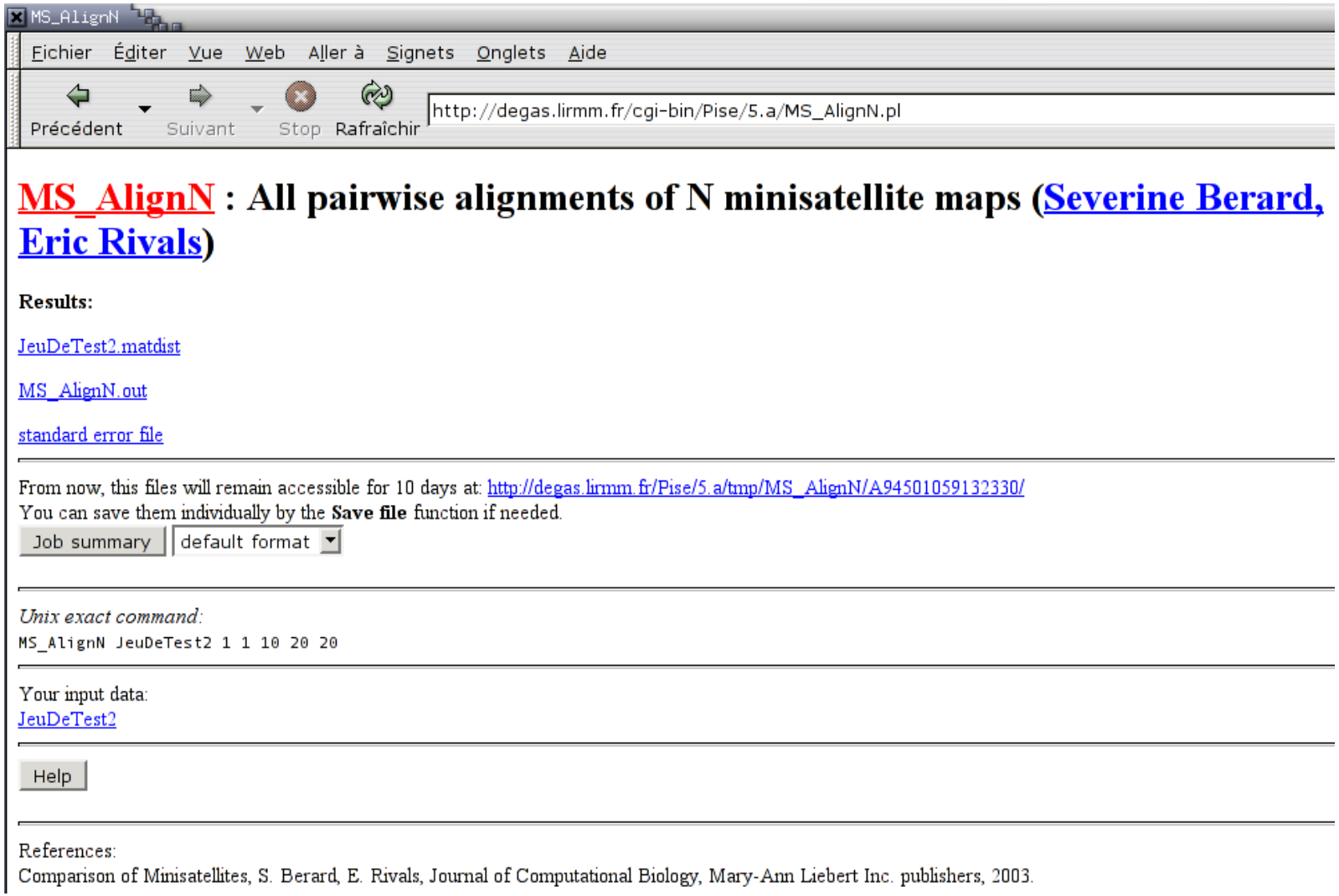


Fig. 5.5 – *Écran de résultat de* MS_AlignN.

```
>m3[Swedish]{1}
bbbbbbbbbbbbbbbbnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnggggggggggggggggggggggg
>m419[Indian]{1}
nnnbbbbbbbbbbbbbbbbnnbbnnnnnnnnnnnnnnnnnnnnnnnnnnnnggggggggggggggggggggggggggggggggg
>m493[Indian]{1}
nnbbbbbbbbbbbbbbbbnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnggggggggggggggggggggggggggggggg
>m119[Australian]{2}
bbbbbbbbbbnnnbbnnnnnnnnnnnnnnnnnnnnnnnnnnnnnggggggggggggggggggggggggggggg
>m109[Indian]{3}
bbbbbbbbbbbbbbbbbbbbbnnnnnnnnnnnnnnnnnnn
```

résultats[2] :

- `JeuDeTest2.matdist`, dont le nom est composé du nom du fichier d'entrée (ou maps.data si l'on a entré les séquences à la main), suivi de l'extension `.matdist`, ce fichier contient le résultat du programme, c'est-à-dire la matrice de distances entre les séquences données en entrée,
- `MS_AlignN.out` contient les affichages de MS_AlignN sur la sortie standard,
- et `standard error file` contient les affichages de MS_AlignN sur la sortie erreur ; les erreurs peuvent provenir d'un mauvais format de fichier par exemple.

Les contenus des fichiers `JeuDeTest2` et `JeuDeTest2.matdist` sont donnés à la figure 5.6. Au bas de cette figure nous avons ajouté l'alignement entre les deux premières séquences de `JeuDeTest2.matdist` obtenu avec MS_Align2. Toutes les séquences du fichier `JeuDeTest2` sont issues du jeu de données biologiques que nous a fourni M. Jobling. Le fichier résultat contenant la matrice de distances, `JeuDeTest2.matdist`, est au format Phylip [`Phylip`], c'est-à-dire que la première ligne indique le nombre total de séquences, ensuite les lignes vont par paires, la première ligne de la paire donne le nom de la séquence, la deuxième, les distances entre cette séquence et toutes les autres, dans l'ordre du fichier. Le fichier `JeuDeTest2.matdist` peut donc être directement passé en paramètre à un programme de reconstruction phylogénétique comme BioNJ [Gascuel, 1997] ou Phylip [`Phylip`].

5.2 Implémentation

Dans cette section nous décrivons succinctement l'implémentation de notre algorithme sous forme d'un programme informatique et ensuite sa « mise en ligne » *via* le logiciel Pise [Letondal, 2001].

5.2.1 Le programme

L'algorithme d'alignement de cartes de minisatellite est implémenté en langage C++. Les deux programmes MS_Align2 et MS_AlignN partagent une grande partie du code, MS_Align 2 possède une partie supplémentaire qui permet de « dessiner » l'alignement et MS_AlignN est capable de gérer plusieurs alignements pour construire la matrice de distances. MS_Align2 et MS_AlignN comportent respectivement environ 2500 et 2800 lignes, commentaires et mise en page compris[3]. Les deux logiciels sont compilables sous différentes plates-formes dont Linux et Sun. Nous prévoyons un portage sous Window et Mac.

Au niveau des temps de calcul, MS_Align2 rend un résultat instantanément. Le temps de calcul de MS_AlignN dépend du nombre de séquences passées en entrée. Le tableau ci-dessous présente les temps de calcul relevés pour MS_AlignN par rapport au nombre

[2]Si le calcul prend plus de quelques minutes, un mèl est envoyé à l'utilisateur lorsque les résultats sont prêts.

[3]Il faut donc en ôter à peu près un quart pour obtenir un code brut.

de séquences en entrée. Les tests ont été effectués sur un Pentium IV, avec un processeur de 1,8 GHz et 1 Go de RAM, ces résultats sont des « temps utilisateurs » avec un pourcentage d'occupation de la CPU allant de 91 % à 98 %.

Nombre de séquences	Temps
10	6 s
20	27 s
40	2 m 37 s
60	4 m 30 s
80	8 m 01 s
100	14 m 03 s
200	50 m 11 s
432	3 h 54 m
609	7 h 43 m

La longueur moyenne des séquences utilisées pour ces tests est de 80 variants. Les chiffres 432 et 609 sont caractéristiques du jeu de données de cartes du minisatellite MSY1 fourni par M. Jobling. En effet, 609 est le nombre total de séquences dans ce jeu de données, et 432 correspond aux nombres de séquences dont l'haplogroupe est déterminé (cf. chapitre 7).

Le logiciel MS_AlignN a également été utilisé sur un jeu de 648 cartes du minisatellite MS205 par É. Fontanillas (Fontanillas, 2002).

Malgré une complexité en temps de l'algorithme sous-jacent en $O(n^3)$ (voir section 4.3.4, page 106), on remarque que le temps d'exécution de MS_Align2 est quasi nul, et que le temps d'exécution de MS_AlignN reste raisonnable même pour des jeux de données importants (parmi les plus grands disponibles actuellement).

5.2.2 L'interface web

Nous avons utilisé le logiciel Pise de Catherine Letondal [Letondal, 2001] pour générer une interface graphique pour nos programmes. Une description de Pise se trouve à l'adresse *http://www-alt.pasteur.fr/~letondal/Pise/*. Il suffit de donner les caractéristiques du programme en langage XML au logiciel Pise pour qu'il génère l'interface graphique et gère l'affichage des résultats et l'envoi de mèls.

Pour MS_AlignN, nous avons constaté un « *bug* » dans Pise : lorsque le fichier résultat n'est pas produit au bout d'une minute, Pise n'envoie pas de mèl pour prévenir l'utilisateur lorsque le fichier est généré. La documentation de Pise ne nous a pas permis d'éviter ce *bug*. Étant donné que nous connaissons le nom du fichier résultat nous pouvons l'obtenir en entrant son nom dans la barre d'adresse du navigateur après la fin des calculs, mais ce n'est pas une solution acceptable. Nous prévoyons de développer une interface web alternative.

Chapitre 6

Extensions du modèle SSE

Sommaire

6.1 Qu'est ce qui change ? 125
6.1.1 Les amplifications 125
6.1.2 Les arches 125
6.1.3 Le nombre d'arches 127
6.1.3.1 Nombre d'arches exactes, ordre et arité quelconques . . 127
6.1.3.2 Nombre d'arches exactes, ordre quelconque et arité 1 . 130
6.1.3.3 Nombre d'arches approchées, ordre quelconque et arité 1 130
6.2 Alignement avec amplification d'ordre 1 et d'arité $\rho - 1$ . . . 130
6.3 Nouvelles problématiques 133
6.3.1 Coût des générations/compressions d'arche 133
6.3.2 Arches approchées 134
6.3.2.1 Limite de la modélisation en arches 136
6.3.2.2 Relation avec le problème d'histoire des duplications . . 137
6.3.3 Conclusion : extensions envisagées 138
6.4 Génération optimale de séquence avec les arches EOS 139
6.4.1 Arches EOS, arches exactes d'ordre supérieur 139
6.4.2 Relation de compatibilité entre arches EOS 140
6.4.2.1 Exemples 140
6.4.2.2 Définitions 141
6.4.3 La relation chronologiquement compatible 144
6.4.4 Ensemble maximal d'arches EOS compatibles 146
6.5 Rapprochement avec des problèmes déjà étudiés 149
6.5.1 Problème de stable max 150
6.5.1.1 Problème de 2-intervalles 150
6.5.1.2 Problème de trapézoïdes circulaires 151
6.5.2 Problème de graphe orienté 151

6.5.2.1 Exemple de graphe de compatibilité 152

6.5.2.2 Les pistes 153

Dans ce chapitre, nous nous intéressons à la résolution du problème de l'alignement de cartes de minisatellite en essayant de mieux prendre en considération leur mécanisme d'évolution. Dans ce but, nous examinons les extensions possibles du modèle sse (voir chapitre 4, section 4.1). Dans ces extensions, nous conservons le même ensemble d'opérations, mais nous essayons de relâcher les contraintes sur les amplifications et les contractions. Nous envisageons par exemple les triplications d'un variant ou l'amplification de deux variants à la fois. Considérer ces nouvelles amplifications étend la notion d'arches que nous avons définie au chapitre 4, section 4.2, et pose de nouveaux problèmes.

Nous essayons de résoudre le problème de l'alignement de cartes sous un modèle étendu d'évolution de séquences de la même manière que dans le chapitre 4, c'est-à-dire par programmation dynamique. D'autres méthodes sont envisageables. Comme nous l'avons vu dans le chapitre 4, c'est le calcul des coûts de génération/compression d'arche qui sont au cœur du problème. Pour ce calcul nous utilisions la recherche d'un stable max dans un graphe de chevauchement construit à partir des arches et de la relation de compatibilité. Dans ce chapitre, c'est ce calcul que nous essayons de résoudre sous un nouveau modèle d'évolution. Nous avons tout d'abord fait un travail de modélisation des « nouvelles » arches. Cela nous a amené à déterminer un modèle d'évolution nommé esse sous lequel nous avons défini plus formellement la relation de compatibilité entre ces nouvelles arches. Nous avons montré que trouver un ensemble maximal d'arches compatibles sous le modèle esse avec cette relation de compatibilité nécessite la recherche d'une chronologie de poids maximal. Cependant, nous n'avons pas trouvé de solution exacte pour le problème de recherche de chronologie de poids maximal. Les pistes que nous avons explorées peuvent donner lieu à des heuristiques.

Ce chapitre est organisé de la manière suivante : dans la section 6.1, nous proposons différentes voies pour étendre le modèle sse. Dans la section 6.2, nous décrivons un article de Behzadi et Steyaert qui propose un algorithme d'alignement de cartes de minisatellite sous un modèle plus général, où les amplifications sont étendues aux p-plications, avec $p \geq 2$. Nous présentons les problématiques qu'implique l'extension du modèle évolutif à la section 6.3. Ensuite, à la section 6.4, nous choisissons une des extensions proposées et nous étudions le problème de trouver le coût de génération optimal d'une séquence dans le but de valuer les opérations de génération/compression d'arche. Pour cela, nous avons besoin de calculer un ensemble maximal d'arches sur cette séquence sous le nouveau modèle d'évolution, cela implique la recherche d'une chronologie de poids maximal. Enfin, la dernière section est une section prospective consacrée aux problèmes déjà étudiés desquels nous avons essayé de nous rapprocher.

6.1 Qu'est ce qui change ?

Les extensions du modèle SSE que nous avons envisagées portent uniquement sur les opérations d'amplification et de contraction. L'ensemble des opérations reste

inchangé : mutation (M), insertion (I), délétion (D), amplification (A), contraction (C), amplification+mutation (A_M) et mutation+contraction (M_C). Les extensions considérées conservent la symétrie. Nous supposons l'hypothèse H1 sur les coûts des opérations, c'est-à-dire que $I > A + M$ et $D > M + C$ (cf. page 82). Ainsi un variant est « inséré » par une amplification+mutation et « délété » par une contraction-mutation. Nous conservons également l'hypothèse $A, C < M, D, I$ (cf. page 81).

Amplification et contraction sont des opérations symétriques, aussi, dans les sections suivantes, nous considérons seulement l'opération d'amplification.

Dans cette section, nous présentons ce qui diffère du modèle SSE dans les extensions que nous envisageons : les amplifications, les arches et le nombre d'arches.

6.1.1 Les amplifications

Dans cette section, nous donnons les définitions concernant les caractéristiques de l'opération d'amplification.

Définition 25 (Ordre) *Nous appelons* **ordre** *d'une amplification le nombre de variant(s) copié(s).*

Définition 26 (Arité) *Nous appelons* **arité** *d'une amplification le nombre de copie(s) qu'elle produit.*

Nous montrons ci-dessous des exemples d'amplifications en précisant leur ordre et leur arité :

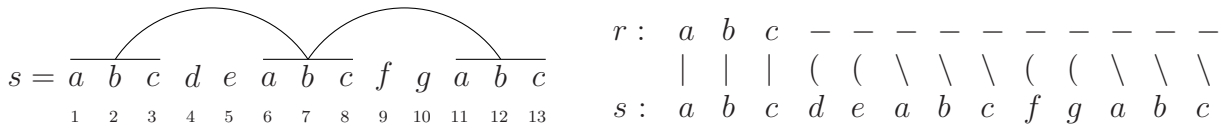
Ordre	Arité	Amplification
1	3	$a \longrightarrow a\,a\,a\,a$
3	1	$abc \longrightarrow abc\,abc$
2	2	$ab \longrightarrow ab\,ab\,ab$

Tout au long du chapitre 4 nous avons considéré des amplifications d'ordre 1 et d'arité 1. Les extensions que nous étudions considèrent des amplifications et contractions d'arité et/ou d'ordre supérieurs à 1.

6.1.2 Les arches

Le principe des arches, comme vu au chapitre 4, dans la section 4.2, est toujours avantageux. Ce sont des facteurs de la séquence délimités par les traces d'une amplification.

Les arches associées à des amplifications d'ordre et d'arité supérieurs à 1 ont une forme différente des arches vues au chapitre 4. Cette différence de forme provient du fait que les motifs de ces arches peuvent être de taille supérieure à 1 et en nombre supérieur à 1 également suivant l'ordre et l'arité de l'amplification qui a produit l'arche. Une arche créée par une amplification d'ordre o et d'arité a est appelée arche d'ordre o et d'arité a. Une illustration de ces nouvelles arches est donnée à la figure 6.1(a). Dans cette figure on montre, sur une séquence s, une arche d'ordre 3 et d'arité 2 dont le motif est abc. On peut aligner une séquence $r = abc$ avec s comme montré à la figure 6.1(b).



(a) (b)

Fig. 6.1 – *La figure (a) présente un exemple de « nouvelle arche » sur une séquence s et la figure (b) montre un alignement se servant de cette arche.*

L'alignement de la figure 6.1(b) peut correspondre au listing suivant :

$r = abc$
Amplification d'ordre 3 et d'arité 2 de abc
$abc\,abc\,abc$
Amplification+Mutation du premier c
$abc\,d\,abc\,abc$
Amplification+Mutation du d
$abc\,de\,abc\,abc$
Amplification+Mutation du second c
$abc\,de\,abc\,f\,abc$
Amplification+Mutation du f
$abc\,de\,abc\,fg\,abc = s$

Définition 27 (Arche d'ordre et d'arité supérieurs à 1) *Une arche d'ordre et d'arité supérieurs à 1 sur une séquence s est une suite ordonnée de facteurs de s $(f_1, f_2, \ldots, f_k)$, tels que $\forall i, j \in [1 \ldots k], i < j \Rightarrow d_{f_i} < g_{f_j}$. Autrement dit, ces facteurs ne se chevauchent pas.*

Définition 28 (Pied d'une arche) *Nous appelons* **pieds d'une arche** *les facteurs composant cette arche. Un pied d'une arche est soit le motif copié soit une copie.*

Dans l'exemple de la figure 6.1(a), l'arche montrée a 3 pieds : $s[1..3]$, $s[6..8]$ et $s[11..13]$. On remarque qu'une arche créée par une amplification d'arité a possède $a + 1$ pieds.

Les amplifications produisent des copies identiques au motif amplifié. Cependant il se peut que des événements mutationnels se produisent sur les pieds de l'arche et ceux-ci ne sont alors plus égaux. Nous distinguons deux types d'arches : les arches exactes et les arches approchées.

Définition 29 (Arche exacte) *Une arche exacte est une arche dont tous les pieds sont identiques, c'est-à-dire qu'ils ont la même longueur et le même motif.*

L'ordre d'une arche exacte est déterminé par la longueur de ses pieds.

Définition 30 (Arche approchée) *Une arche approchée est une arche dont les pieds ne sont pas tous identiques.*

Les générations/compressions d'arche exacte et approchée peuvent permettre de faire des économies lors d'un alignement. Comme on peut le voir sur l'alignement de la figure 6.1(b), l'arche d'ordre 3 et d'arité 2 permet de générer 6 variants par amplifications, bien que ceux-ci ne soient pas adjacents à des variants identiques. Sans arche, ces 6 variants auraient dû être générés par A_M et le coût de l'alignement aurait donc augmenté de $6 \times M$.

De la même manière que dans le chapitre 4, les arches peuvent être complexes, c'est-à-dire contenir d'autres arches. Obtenir le meilleur coût de génération/compression d'une arche complexe nécessite de trouver un ensemble d'arches internes compatibles. La relation de compatibilité entre arches doit donc être redéfinie pour s'adapter aux arches d'ordre et d'arité supérieurs à 1.

6.1.3 Le nombre d'arches

Dans cette section, nous recensons le nombre d'arches qui peuvent se trouver sur une séquence de longueur n. Nous appelons N_o^a le nombre d'arches d'ordre o et d'arité a. Nous notons la partie entière inférieure d'un décimal d par $\lfloor d \rfloor$ et la combinaison de k éléments parmi n par :

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}$$

Nous commençons par dénombrer les arches d'ordre et d'arité quelconques à la section 6.1.3.1. Nous nous restreignons ensuite à l'arité 1. Pour cette arité, nous comptons le nombre d'arches exactes (section 6.1.3.2), puis le nombre d'arches approchées (section 6.1.3.3).

6.1.3.1 Nombre d'arches exactes, ordre et arité quelconques

Nous comptons ici les arches exactes de tout ordre et de toute arité dans le pire des cas, c'est-à-dire sur une séquence s contenant n variants identiques. Les pieds de ces arches ont la même longueur. La table 6.1 récapitule le dénombrement des arches par ordre et par arité. Dans ce tableau, nous dénombrons les arches en sommant sur les différentes positions possibles pour leurs pieds.

Ordre 1		
	Arité 1	$N_1^1 = \sum_{i=1}^{n-1}(n-i) = \frac{1}{2}\ n(n-1)$
	Arité 2	$N_1^2 = \sum_{i=1}^{n-2}\sum_{j=i+1}^{n-1}(n-j) = \frac{1}{6}\ n(n-1)(n-2)$
	Arité 3	$N_1^3 = \sum_{i=1}^{n-3}\sum_{j=i+1}^{n-2}\sum_{k=j+1}^{n-1}(n-k) = \frac{1}{24}\ n(n-1)(n-2)(n-3)$
	Arité a	$N_1^a = \left(\begin{array}{c} n \\ a+1 \end{array}\right) = \frac{n!}{(a+1)!(n-a-1)!}$
	Arité $n-1$	$N_1^{n-1} = 1$
Ordre 2		
	Arité 1	$N_2^1 = \sum_{i=1}^{n-3}(n-i-2) = \frac{1}{2}\ (n-2)(n-3)$
	Arité 2	$N_2^2 = \sum_{i=1}^{n-5}\sum_{j=i+2}^{n-3}(n-j-2) = \frac{1}{6}\ (n-3)(n-4)(n-5)$
	Arité a	$N_2^a = \left(\begin{array}{c} n-(a+1) \\ a+1 \end{array}\right) = \frac{(n-a-1)!}{(a+1)!(n-2a-2)!}$
	Arité $\lfloor\frac{n-2}{2}\rfloor$	$N_2^{\lfloor\frac{n-2}{2}\rfloor} = 1$
Ordre 3		
	Arité 1	$N_3^1 = \sum_{i=1}^{n-5}(n-i-4) = \frac{1}{2}\ (n-4)(n-5)$
	Arité 2	$N_3^2 = \sum_{i=1}^{n-8}\sum_{j=i+3}^{n-5}(n-j-4) = \frac{1}{6}\ (n-6)(n-7)(n-8)$
	Arité a	$N_3^a = \left(\begin{array}{c} n-2(a+1) \\ a+1 \end{array}\right) = \frac{(n-2a-2))!}{(a+1)!(n-3a-3)!}$
	Arité $\lfloor\frac{n-3}{3}\rfloor$	$N_3^{\lfloor\frac{n-3}{3}\rfloor} = 1$
Ordre o		
	Arité 1	$N_o^1 = \sum_{i=1}^{n-(2o-1)}(n-i-(2o-2)) = \frac{1}{2}(n-2o+2)(n-2o+1)$
	Arité a	$N_o^a = \left(\begin{array}{c} n-(o-1)(a+1) \\ a+1 \end{array}\right)$
	Arité $\lfloor\frac{n-o}{o}\rfloor$	$N_o^{\lfloor\frac{n-o}{o}\rfloor} = 1$

Tab. 6.1 – *Tableau de dénombrement des arches d'ordre et d'arité supérieurs à* 1.

$$s = \underset{1}{x}\ x\ x\ x\ x\ x\ \underbrace{x\ x}_{i}\ x\ x\ x\ \underbrace{x\ x}_{j}\ x\ x\ \underbrace{x\ x}_{k}\ x\ x\ x\ x\ x\ x\ \underset{n}{x}$$

FIG. 6.2 – *Illustration du dénombrement de N_2^2. i, j et k sont respectivement les premières positions des premier, deuxième et dernier pieds d'une arche d'ordre 2 et d'arité 2.*

Calcul de $\mathbf{N_2^2}$ Prenons par exemple le calcul de N_2^2, c'est-à-dire le nombre d'arches d'ordre 2 et d'arité 2. La figure 6.2 montre un schéma correspondant à ce cas. Nous appelons i la position de début du premier pied d'une arche. Cette position i peut se situer entre 1 et $n-5$, il faut en effet au moins 5 positions derrière i pour placer le deuxième motif du premier pied et les deux autres pieds de longueur 2. La position de début du second pied, appelée j, peut se situer entre $i+2$ (il ne faut pas qu'il y ait de chevauchement entre les pieds d'une arche), et $n-3$ (pour pouvoir placer la fin du second pied et le dernier pied). Il reste ensuite $n-j-2$ positions pour placer le début du dernier pied, d'où la formule :

$$N_2^2 = \sum_{i=1}^{n-5} \sum_{j=i+2}^{n-3} n-j-2$$

Calcul de $\mathbf{N_o^a}$ Compter le nombre d'arches d'ordre o et d'arité a revient à compter le nombre de placements différents des débuts de pieds de cette arche parmi les $n-(o-1)(a+1)$ positions possibles. Il n'y que $n-(o-1)(a+1)$ positions possibles car deux pieds d'une même arche ne se chevauchent pas. Lorsque l'on place le début d'un pied d'une arche d'ordre o à la position i, les $(o-1)$ positions suivantes ne peuvent pas recevoir le début d'un autre pied. Lorsque l'on doit placer les $(a+1)$ pieds d'une arche d'ordre o et d'arité a, il y a $(o-1)(a+1)$ positions « interdites ». Le nombre d'arches différentes est donc la combinaison de $(a+1)$ débuts de pieds parmi $n-(o-1)(a+1)$ positions possibles, c'est-à-dire :

$$N_o^a = \begin{pmatrix} n-(o-1)(a+1) \\ a+1 \end{pmatrix}$$

Calcul du nombre total d'arches Sur une séquence de longueur n, l'ordre maximal d'une arche est $\lfloor \frac{n}{2} \rfloor$. L'arité maximale dépend de l'ordre de l'arche, pour une arche d'ordre o et d'arité a, il faut pouvoir placer les $(a+1)$ pieds de taille o sur la séquence de longueur n, d'où :

$$(a+1) \times o \leq n \Longleftrightarrow a \leq \frac{n-o}{o}$$

L'arité étant un nombre entier, l'arité maximale est $\lfloor \frac{n-o}{o} \rfloor$. Le nombre total d'arches, d'ordre et d'arité quelconques, sur une séquence s de longueur n dans le pire des cas est donc donné par la formule suivante :

$$A(s) = \sum_{o=1}^{\lfloor \frac{n}{2} \rfloor} \sum_{a=1}^{\lfloor \frac{n-o}{o} \rfloor} \left(\begin{array}{c} n-(o-1)(a+1) \\ a+1 \end{array} \right)$$

Étant donné que $\sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) = O(2^n)$, on a $A(s) = O(2^n)$. On remarque que même en se limitant aux arches d'ordre 1 et d'arité quelconque, on a déjà $O(2^n)$ arches.

6.1.3.2 Nombre d'arches exactes, ordre quelconque et arité 1

Ici, nous nous limitons à l'arité 1 et nous comptons le nombre d'arches d'ordre quelconque sur une séquence de longueur n dans le pire des cas. D'après la table 6.1, ce nombre est donné par :

$$\sum_{o=1}^{\lfloor \frac{n}{2} \rfloor} N_o^1 = \sum_{o=1}^{\lfloor \frac{n}{2} \rfloor} \frac{1}{2}(n-2o+2)(n-2o+1)$$

Ce qui donne :

$$\text{si } n \text{ est pair}, \quad \sum_{o=1}^{\frac{n}{2}} N_o^1 = \frac{1}{24}(2n^3+3n^2-2n) = O(n^3) \ ;$$

$$\text{si } n \text{ est impair}, \quad \sum_{o=1}^{\frac{n-1}{2}} N_o^1 = \frac{1}{24}(2n^3+3n^2-2n-3) = O(n^3).$$

6.1.3.3 Nombre d'arches approchées, ordre quelconque et arité 1

Une arche approchée d'arité 1 peut être vue comme une paire de facteurs qui ne se chevauchent pas. Le nombre d'arches approchées d'arité 1, sur une séquence de longueur n, est égal au nombre de combinaisons des 4 extrémités des facteurs parmi les $n+2$ positions possibles. Il y a $n+2$ positions car les facteurs peuvent être de longueur 1 et donc les extrémités se situent à la même position. Le nombre d'arches approchées d'arité 1, sur une séquence de longueur n, est par conséquent donné par le terme suivant :

$$\left(\begin{array}{c} n+2 \\ 4 \end{array} \right) = O(n^4)$$

6.2 Alignement avec amplification d'ordre 1 et d'arité $\rho-1$

Dans cette section, nous présentons l'article [Behzadi et Steyaert, 2003]. Behzadi et Steyaert proposent un algorithme d'alignement de cartes de minisatellite sous le modèle

SSE où les amplifications peuvent être d'arité supérieure à 1. La complexité dans ce modèle plus général est $(n^3|\Sigma|\rho)$ en temps et $O(n^2|\Sigma|\rho)$ en espace, où Σ est l'alphabet de motifs considérés et ρ l'arité maximale des amplifications et contractions plus 1.

Ils considèrent l'alignement entre deux séquences s et r comme une série de transformations pour passer de s à r. Dans cette transformation, chaque symbole de s va générer une sous-chaîne de r, cette sous-chaîne pouvant être vide. Les symboles de s qui génèrent des sous-chaînes non vides de r sont nommés *symboles génératifs* et les autres symboles sont nommés *symboles disparaissants*. Ils appellent *génération* une transformation d'un symbole x en une chaîne non vide et *génération non décroissante*, une génération qui utilise seulement des mutations, insertions et amplifications. REMARQUE : Les symboles mutés font à la fois partie des symboles génératifs et des symboles disparaissants.

La solution de Behzadi et Steyaert repose sur deux lemmes principaux :

Lemme 5 (Lemme de génération) *La génération optimale d'une chaîne non vide à partir d'un symbole x peut être obtenue par une génération non décroissante.*

Lemme 6 (Indépendance des contractions) *Dans une transformation optimale d'une chaîne s à une chaîne r, toutes les opérations de contraction peuvent être faites avant les opérations de génération.*

Une représentation de leur principe d'alignement est donné à la figure 6.3. Leur transformation se fait en deux phases, d'abord ils réduisent les sous-chaînes disparaissantes et ensuite ils effectuent les générations. Leur algorithme se divise en deux parties :

- un prétraitement qui calcule les coûts des générations des sous-chaînes de r, stockés dans la matrice R, et les coûts des réductions des sous-chaînes de s, stockés dans la matrice S. L'entrée $R(i, j, x)$ donne le coût minimal de la génération de la sous-chaîne $r[i..j]$ à partir du symbole x. L'entrée $S[i, j]$ donne le coût minimal de réduction de la sous-chaîne $s[i..j]$ en $s[i]$;
- une seconde phase qui détermine la distance de transformation optimale. Le calcul se fait dans une matrice à deux dimensions, nommée TD, de la manière suivante, le résultat se trouvant dans la case en bas à droite de cette matrice :

  Initialisation

  $$TD(0,0) = 0 \quad \text{et} \quad TD(0,j) = \infty \quad \forall j > 0,$$

  Récurrence

  $$\forall i > 0 \quad TD(i,j) = \min \begin{cases} TD(i-1,l) + R(l+1, j, s[i]) & \forall 0 \leq l < j \\ TD(k,j) + S(k,i) & \forall 0 < k < i \end{cases}.$$

Les deux phases de leur algorithme utilisent la programmation dynamique, à la différence de notre algorithme sous le modèle SSE (cf. chap. 4), qui pour la phase de prétraitement, résout un problème de graphe.
Bien que leur méthode améliore la complexité, leur article n'éclaircit pas certains points. Par exemple, ils utilisent une matrice R à trois dimensions pour calculer les générations des

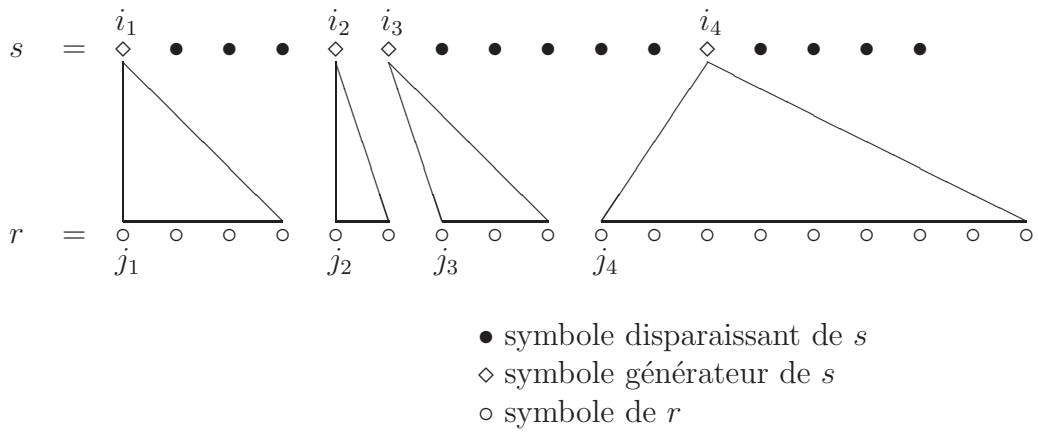


Fig. 6.3 – *Alignement de cartes de minisatellite [Behzadi et Steyaert, 2003].*

sous-chaînes de r à partir d'un symbole quelconque. Cependant, la matrice S qui calcule les contractions de sous-chaînes de s n'a que deux dimensions, la réduction des sous-chaînes $s[i..j]$ se faisant toujours dans le symbole $s[i]$. Ainsi, dans la preuve de la validité de leur récurrence principale ([Behzadi et Steyaert, 2003, p. 40]), ils oublient le cas où $s[i]$ est un symbole disparaissant qui se réduit en un symbole de r.

De plus, dans leur explication, $s[1]$ est toujours un symbole génératif : « Il y a deux séquences de positions $1 = i_1 < i_2 < \ldots < i_l = n + 1$ et $1 = j_1 < j_2 < \ldots < j_l = m + 1$ telles que i_1, i_2, …, i_{l-1} sont les positions génératives dans la chaîne s : $s[i_k]$ génère $r[j_k..(j_{k+1} - 1)]$ pour tout $1 \leq k < l$. Avant les générations, les sous-chaînes $s[i_k..(i_{k+1} - 1)]$ sont réduites à $s[i_k]$ pour tout $k < l$. » [Behzadi et Steyaert, 2003, p. 37-8], ceci est illustré à la figure 6.3. Or un alignement de coût optimal n'a pas toujours $s[1]$ comme symbole génératif.

En appliquant la méthode décrite dans cet article, nous avons aligné $s = ab$ avec $r = b$ où $\Sigma = \{a, b\}$, $\rho = 2$, c'est-à-dire seulement des amplifications d'arité 1, le coût des mutations est 10, celui des insertions et délétions est 20, et les amplifications et contractions coûtent 1. Nous trouvons l'alignement suivant :

$$\begin{pmatrix} a & b \\ - & b \end{pmatrix}$$

Cet alignement coûte 20 (délétion du a), alors que

$$\begin{pmatrix} a & b \\ b & - \end{pmatrix}$$

coûte seulement 11 (mutation $a \to b$ de coût 10, il y a donc maintenant deux b dans la séquence, on peut donc contracter l'un d'eux en l'autre pour un coût de 1). Ce problème vient *a priori* du fait que la méthode ne calcule pas le coût de réduction de ab dans b.

La méthode de Behzadi et Steyaert qui utilise seulement de la programmation dynamique est très intéressante. Elle aborde le problème sous l'angle des grammaires formelles. Il est surprenant que la non-commutativité des opérations, et donc l'ordre d'application des événements mutationnels, qui sont la principale source de difficultés, n'apparaissent pas dans leur article.

6.3 Nouvelles problématiques

Pour étendre le modèle SSE, nous avons tout d'abord envisagé d'autoriser les amplifications et contractions de tout ordre et de toute arité. Mais cela génère un nombre exponentiel d'arches à considérer comme vu à la section 6.1.3. Cette extension n'est donc pas appropriée pour résoudre le problème de l'alignement de la même manière qu'au chapitre 4.

Nous nous limitons donc à une extension du modèle SSE qui autorise les amplifications et contractions de tout ordre mais d'arité 1 seulement. Nous appelons ce type d'amplification, les *amplifications d'ordre supérieur* et les arches qu'elles créent, *les arches d'ordre supérieur*. Cette restriction ne nous gène pas pour la modélisation des événements évolutifs des minisatellites, car c'est l'arité 1 qui est la plus probable pour ce type d'événements (Jobling et al., 1998) (voir aussi chapitre 7, page 161).

Quel est le nouveau problème à résoudre ? Nous souhaitons trouver un alignement optimal entre deux cartes de minisatellite s et r, de longueurs respectives n et m, sous le modèle SSE étendu aux amplifications et contractions d'ordre supérieur.

Pour résoudre ce problème, nous avons besoin de considérer les arches d'ordre supérieur. Cela nécessite de valuer les compressions et générations d'arche, et pour ce faire de fixer un coût pour les amplifications et contractions d'ordre supérieur. Plusieurs questions se posent : quel coût donner aux amplifications et aux générations/compressions d'arche ? mais aussi peut-on vraiment prendre en compte les arches approchées ?

6.3.1 Coût des générations/compressions d'arche

Le coût d'une génération/compression d'arche d'ordre supérieur dépend d'une part du coût de l'amplification d'ordre supérieur mais aussi de la manière de créer sa partie interne.

Nous notons c la fonction de coût valuant les amplifications d'ordre o. Plusieurs possibilités se présentent, où A est le coût d'une amplification d'ordre 1 :

- $c(o) = o \times A$, le plus simple ;
- $c(o) = \frac{o \times A}{\log o}$, permet de privilégier les amplifications d'ordre élevé ;
- $c(o) = \sqrt{o}A$, *idem* ;
- ou $c(o) = o^{\alpha}A$, $\alpha \in \mathbb{R}^{+*}$, le plus général.

Cette liste n'est pas exhaustive. Nous souhaitons choisir le coût le plus vraisemblable pour notre application biologique.

La partie interne d'une arche peut être calculée récursivement en se servant de l'ensemble d'arches internes compatibles de poids maximal, où le poids d'une arche est l'éco-

nomie produite par son utilisation. Cette notion de poids est facile à saisir si l'on choisit le coût le plus simple, c'est-à-dire $c(o) = o \times A$. Dans ce cas, l'économie apportée par une arche simple d'ordre o est $o \times M$. Plus généralement, l'économie amenée par une arche exacte d'ordre o est $o \times (A + M) - c(o)$.

Donner un poids à une arche approchée est plus difficile. En effet, ce poids dépend à la fois du coût de l'amplification mais aussi de la manière de « transformer » un pied en l'autre. L'arche étant approchée, les deux pieds ne sont pas identiques. Plusieurs problèmes se posent :

- quel pied choisir comme « géniteur » ? Par exemple, une arche approchée dont les deux pieds sont *abc* et *abbc* ne produira pas la même économie si l'on amplifie *abc* ou *abbc* : dans le 1er cas, générer le second pied de l'arche coûte $c(3) + A$; dans le 2nd cas, $c(4) + C$;
- est-on sûr que l'amplification ayant produit l'arche approchée est de motif l'un des deux pieds ? Par exemple, une amplification de *abcd* suivie de deux mutations peut donner :

  $$abcd \longrightarrow abcd\,abcd \longrightarrow aecd\,abfd.$$

  Comment considérer cette arche, et surtout comment l'aligner avec la séquence d'en face ? Faut-il aligner le motif *abcd*, *aecd* ou *abfd* ?
- si les pieds sont longs, trouver l'amplification d'ordre supérieur puis la série de transformations menant au pied généré peut nécessiter un alignement entre les deux pieds. Ceci pose donc un problème de récursivité.

Donner un poids aux arches approchées n'est donc pas simple.

Dans la suite de ce chapitre, nous considérons que les amplifications d'ordre o coûtent $o \times A$. Nous pouvons alors définir le poids d'une arche exacte d'ordre o et d'arité 1.

Définition 31 (Poids d'une arche exacte d'ordre o et d'arité 1) *Le poids d'une arche exacte d'ordre o et d'arité 1, est $o \times M$.*

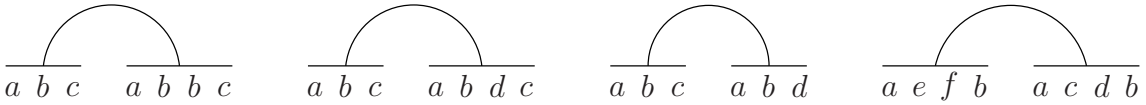
6.3.2 Arches approchées

Nous avons vu sur l'alignement de la figure 6.1(b), page 126, que les arches exactes permettaient des économies. Les arches approchées peuvent également être utilisées dans ce but. Une amplification d'ordre supérieur suivie d'une délétion par exemple, peut être avantageuse. La figure 6.4 montre quelques exemple d'arches approchées intéressantes.

Nous considérons ici les alignements du pied gauche de l'arche avec l'arche entière, et nous supposons que les 2 pieds gauches sont appariés. Pour le cas (c) par exemple, cela donne :

$$\begin{array}{cccccc} a & b & c & - & - & - \\ | & | & | & ? & ? & ? \\ a & b & c & a & b & d \end{array}$$

Nous cherchons à calculer le coût de cet alignement en remplaçant les '?'. Examinons les exemples présentés à la figure 6.4 :

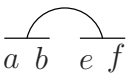


(a) (b) (c) (d)

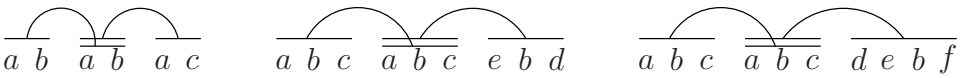
FIG. 6.4 – *Exemples d'arches approchées.*

- l'arche (a) permet de générer les 4 variants du pied droit par amplifications – 3 par l'amplification d'ordre supérieur de motif *abc* et le 2e *b* par une amplification d'ordre 1 – et produit donc une économie de $4M$ sur un alignement sans arche, et de $2M$ si on considère l'arche de motif *c* et l'arche de motif *b* incluse dans l'arche *c* ;
- l'arche (b) permet de faire une seule A_M, celle du *d*, cela entraîne une économie de $3M$ sur un alignement sans arche, et de $2M$ si l'on considère l'arche de motif *c* ;
- l'arche (c) amène une économie de $2M$. Notons que l'on ne peut pas générer une arche exacte de motif *ab* à moins de ne pas apparier les deux variants *c* ensemble (cf. schéma ci-dessus) ;
- l'arche (d) permet quant à elle de générer le *a* et le *b* du pied droit par amplification au lieu du *b* seulement.

REMARQUE : Si l'on considère un coût d'amplification d'ordre *o* égal à $o^\alpha A$, où $\alpha < 1$, alors l'arche suivante devient également intéressante :



Les arches de la figure 6.5 peuvent elles aussi amener des économies. On considère encore les alignements du pied gauche de la première arche avec la séquence entière, dans lesquels les deux pieds gauches sont appariés. Ces exemples montrent des arches qui seules ne sont pas utiles pour engendrer des économies, mais peuvent le devenir si elles sont combinées à d'autres.



(a) (b) (c)

FIG. 6.5 – *Exemples d'arches approchées (2).*

Prenons l'arche $(ab - ac)$ de la figure 6.5(a). Elle économise seulement la mutation du variant *a*, tout comme le ferait l'arche $(a - a)$, cependant, si l'arche $(ab - ab)$ appartient à l'alignement, on ne peut plus faire l'arche $(a - a)$ car le 1er *a* de cette arche est alors suivi

d'un b, mais on peut toujours faire l'arche $(ab - ac)$:

$$\begin{array}{cccccc} a & b & - & - & - & - \\ | & | & ? & ? & ? & ? \\ a & b & a & b & a & c \end{array} \qquad \begin{array}{cccccc} a & b & - & - & - & - \\ | & | & \backslash & \backslash & \backslash & (\\ a & b & a & b & a & c \end{array}$$

Le principe est le même pour les arches $(abc - ebd)$ du cas (b) et $(abc - debf)$ du cas (c).

Nous venons de présenter des exemples d'arches approchées et leur utilité dans un alignement. Dans les deux sections suivantes nous allons voir que tenir compte des arches approchées dans le problème de l'alignement, et donc dans le problème de la recherche d'un ensemble maximal d'arches, est difficile.

6.3.2.1 Limite de la modélisation en arches

Nous avons défini les arches approchées comme des arches dont les pieds ne sont pas identiques. Elles correspondent à des amplifications d'ordre supérieur suivies d'autres événements mutationnels. Est-ce que cette succession d'événements est toujours modélisable sous la forme d'arches ?

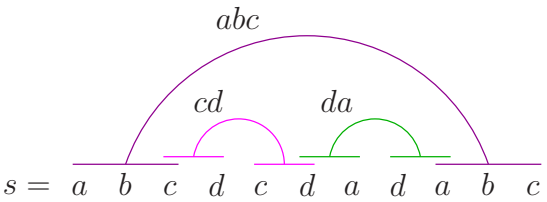


Fig. 6.6 – *Un exemple embêtant.*

La figure 6.6 montre une séquence s sur laquelle nous avons représenté 3 arches, abc, cd et da. Dans un alignement entre $r = abc$ et s, la génération de ces trois arches permettrait de générer 7 variants par amplification seulement, sur les 8 variants à générer pour passer de r à s. Le coût d'un tel alignement serait donc $8A + M$ (le d de la quatrième position étant généré par une A_M).

Il n'est pas possible que deux amplifications différentes génèrent le même variant. Or, quels que soient le sens et l'ordre d'application de ces trois arches, il y a toujours un variant généré deux fois. Par exemple, si l'on essaye d'effectuer les trois amplifications de la gauche vers la droite dans l'ordre abc, puis cd et enfin da, on constate que l'amplification de da génère un a déjà produit par l'amplification de abc. Nous pourrions donc conclure que le coût d'un tel alignement est strictement supérieur à $8A + M$. Pourtant le listing suivant donne un tel coût :

$r = abc$		
	Amplification d'ordre 3 et d'arité 1 de abc	**3A**
$abc\,abc$		
	Amplification+Mutation du premier c	$\mathbf{A + M}$
$abc\,d\,abc$		
	Amplification d'ordre 2 et d'arité 1 de cd	**2A**
$abcd\,cd\,abc$		
	Amplification d'ordre 2 et d'arité 1 de da	**2A**
$abcd\,cda\,dabc = s$		
		$\mathbf{8A + M}$

Comment une modélisation avec des arches peut représenter un tel listing ? En fait le pied généré par l'arche *abc* est coupé en deux par le pied généré par l'arche *da*. On peut imaginer une représentation des arches telle que leurs pieds puissent être en plusieurs morceaux. La figure 6.7 montre ce que cela donne sur l'exemple précédent.

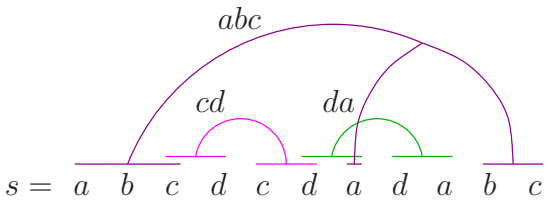


FIG. 6.7 – *Les arches multipied, une solution ?*

À la fois le pied droit et le pied gauche peuvent être séparés en plusieurs parties. Une arche définie comme un couple de facteurs n'est donc plus utilisable dans ce cas. Ce que nous avons défini comme arche approchée ne correspond pas à tous les types d'arches approchées provenant d'une amplification d'ordre supérieur suivie d'événements mutationnels. Nous pouvons modéliser les arches approchées sous forme de deux facteurs si leurs pieds n'ont pas subi d'« insertions » de variants provenant d'une autre amplification.

Donc notre modélisation actuelle des arches ne permet pas de prendre en compte des arches de type similaire à l'arche *abc* dans la figure 6.7.

6.3.2.2 Relation avec le problème d'histoire des duplications

Nous montrons ici que considérer toutes les arches d'ordre supérieur (donc aussi des arches de type « multipied » comme illustré à la figure 6.7) dans le problème d'alignement est fortement similaire au problème de trouver l'histoire optimale des duplications dans une séquence (cf. section 3.2.1, page 72).

Pour cela considérons l'alignement d'une carte s de longueur n et d'une carte $r = \varnothing$. Ce problème revient à trouver un ensemble d'arches compatibles de poids maximal sur s. Une

fois cet ensemble trouvé, pour aligner s avec $\varnothing$ il suffit de muter+contracter les variants qui n'appartiennent pas à des pieds d'arches, puis de compresser les arches de l'ensemble, de la plus interne à la plus externe. Une fois la dernière arche compressée, on délète son variant graine. Notons qu'il n'y a qu'une seule opération de délétion dans cet alignement – ceci est dû à l'hypothèse H1. Dans l'alignement, les mutations-contractions peuvent être vues comme des arches approchées de longueur 2.

Le problème de l'histoire des duplications est de trouver la suite de contractions de poids minimal qui réduit une séquence de k motifs de taille m en un seul motif. Ces contractions sont associées à une fonction de coût.

Trouver l'histoire des duplications d'une carte de minisatellite où chaque variant est représenté par un motif de taille 1 – son symbole – et où seules les contractions binaires sont autorisées, permet de trouver l'alignement de cette carte avec $\varnothing$ en considérant les arches d'ordre supérieur. En effet, chaque contraction de l'histoire peut être associée à une compression d'arche, et le motif restant à la suite des contractions peut être supprimé par délétion pour obtenir $\varnothing$.

Trouver l'alignement – ou le listing correspondant – entre une séquence de m motifs de taille 1 et $\varnothing$ en utilisant les arches d'ordre supérieur permet de trouver une histoire des duplications de la séquence. Il suffit d'ôter la seule opération de délétion présente dans cet alignement et de faire correspondre les compressions d'arches aux contractions du problème de l'histoire des duplications. Mais est-ce l'histoire optimale ?

6.3.3 Conclusion : extensions envisagées

Dans cette section, nous précisons l'extension du modèle sse que nous considérons dans tout le reste de ce chapitre.

Nous nous limitons aux arches exactes d'ordre supérieur. Nous donnons un coût $o \times A$ aux amplifications d'ordre o. Certains résultats sur les arches exactes peuvent s'appliquer sur les arches approchées si un poids est défini pour ce type d'arches.

Nous appelons les arches Exactes d'Ordre Supérieur les arches eos. Nous définissons un nouveau modèle d'évolution pour les séquences, nommé esse (pour *Extended Single Step Evolutionary model*). Ce nouveau modèle comprend les opérations unitaires suivantes :

- mutation de coût M ;
- insertion de coût I ;
- délétion de coût D ;
- amplification d'ordre supérieur de coût $o \times A$, où o est l'ordre de l'amplification ;
- contraction d'ordre supérieur de coût $o \times C$, où o est l'ordre de la contraction ;
- A_M, qui correspond à une amplification d'ordre 1 et d'arité 1 suivie d'une mutation, de coût $A + M$;
- M_C, qui correspond à une mutation suivie d'une contraction d'ordre 1 et d'arité 1, de coût $M + C$.

Notons qu'une A_M « insère » un seul caractère pour un coût de $A + M$, et qu'une amplification « insère » autant de variants que son ordre.

Dans le reste de ce chapitre, nous considérons seulement les arches EOS sauf précision contraire. Nous nous intéressons principalement au problème de trouver un listing de coût optimal pour générer une séquence s à partir de $\varnothing$ sous le modèle ESSE restreint aux arches EOS. En effet, trouver une méthode pour calculer un tel listing nous permettrait de calculer les coûts de générations/compressions d'arche. Nous pourrions alors utiliser les générations/compressions d'arche comme des opérations élémentaires dans un alignement ou un listing.

6.4 Génération optimale de séquence avec les arches EOS

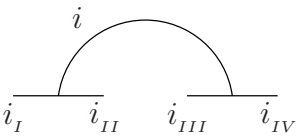
Le but de cette section est de trouver une méthode permettant de calculer un listing de coût optimal transformant $\varnothing$ en une séquence s sous le modèle ESSE restreint aux arches EOS. Sous ces hypothèses, un listing de coût optimal qui génère s à partir de $\varnothing$ contient seulement des insertions, amplifications et amplifications+mutations. De plus, lorsqu'une amplification génère une arche dans un listing, les pieds de cette arche ne peuvent pas subir d'autres événements mutationnels sinon on sort de la restriction aux arches EOS. Autrement dit, tous les événements d'amplification possibles dans les listings transformant $\varnothing$ en s correspondent à des arches EOS visibles sur la séquence s.

Cette section est organisée de la manière suivante : dans un premier temps nous définissons les arches EOS de manière formelle. Ensuite, dans la section 6.4.2, nous caractérisons la relation de compatibilité entre arches EOS. Dans la section 6.4.3, nous décrivons la relation de compatibilité que nous avons définie. Enfin, dans la section 6.4.4, nous expliquons comment construire un listing de coût optimal transformant $\varnothing$ en s en connaissant un ensemble maximal d'arches EOS compatibles sur s.

6.4.1 Arches EOS, arches exactes d'ordre supérieur

Définition 32 (Arche EOS) *Soit s une séquence de longueur n. Une arche EOS est un couple de facteurs $(s[i_I..i_{II}], s[i_{III}..i_{IV}])$ tels que :*
- $1 \leq i_I \leq i_{II} < i_{III} \leq i_{IV} \leq n$;
- $s[i_I, i_{II}] = s[i_{III}, i_{IV}]$.

Une arche EOS est représentée de la manière suivante :



On remarque que si $i_I = i_{II}$ et $i_{III} = i_{IV}$, on est dans le cas d'une arche « classique » comme vu au chapitre 4, c'est-à-dire une arche d'ordre 1 et d'arité 1. Des exemples d'arches EOS sont donnés à la figure 6.8.

Dans cet exemple, il y a 10 arches d'ordre 1 (6 de motif d, 3 de motif c et 1 de motif b), 3 arches d'ordre 2 de motif cd et 1 arche d'ordre 3 de motif bcd. On peut entrevoir la complexité engendrée par l'extension de la définition d'amplification.

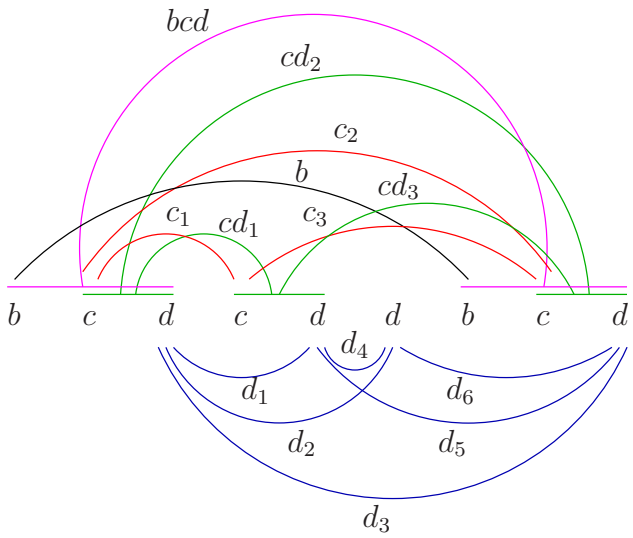


Fig. 6.8 – *Exemple d'arches* eos*, arches exactes d'ordre supérieur.*

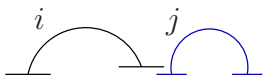
6.4.2 Relation de compatibilité entre arches eos

Dans un premier temps nous donnons des exemples de compatibilité entre arches eos. Nous définissons ensuite formellement la relation de compatibilité.

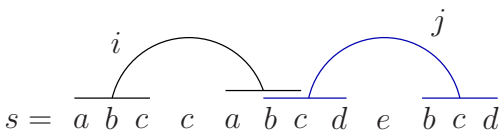
6.4.2.1 Exemples

Nous considérons que deux arches d'une séquence s sont compatibles si elles peuvent être générées dans le même listing produisant s à partir de $\varnothing$. Ainsi, les arches i et j montrées ci-dessous sont compatibles :

Les arches i et j dans la configuration ci-dessous sont également compatibles :



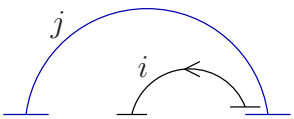
Dans cette dernière configuration, les parties qui se chevauchent sont identiques. Une illustration de cette configuration est montrée sur une séquence $s = abc\ c\ a\,bc\,d\ e\ bcd$, où l'arche i est l'arche d'ordre 3 et de motif abc et l'arche j est d'ordre 3 et de motif bcd :



La séquence s peut être alignée avec $r = \varnothing$ de la manière suivante :

$r = \varnothing$
Insertion de a
a
Amplification+Mutation de a
ab
Amplification+Mutation de b
abc
Génération de l'arche i de motif abc
$abc\ abc$
Amplification+Mutation du deuxième c
$abc\ abc\ d$
Amplification du premier c
$abc\,c\,abc\,d$
Génération de l'arche j de motif bcd
$abc\,c\,abc\,d\ bcd$
Amplification+Mutation du premier d
$abc\ c\ a\,bc\,d\ e\ bcd = s$

De la même manière, les arches i et j dans la configuration ci-dessous sont compatibles :



j contient i

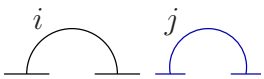
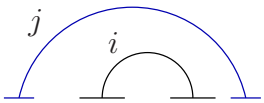
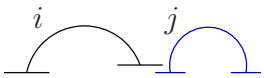
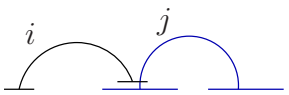
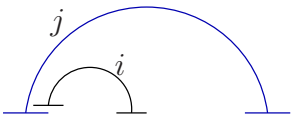
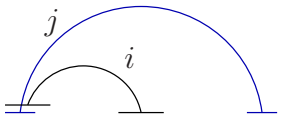
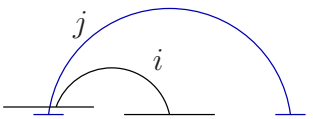
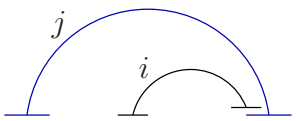
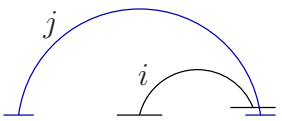
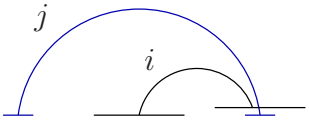
Cette configuration implique que l'arche interne i soit générée après l'arche j, mais aussi que l'amplification associée à l'arche i s'effectue de la droite vers la gauche. En effet, une amplification ne peut pas produire des variants déjà présents.

La notion de chronologie dans la génération/compression d'arche était déjà présente au chapitre 4. En effet, dans une arche complexe, les arches les plus internes devaient être compressées en premier (cf. p. 93), donc générées en dernier. La notion de sens est quant à elle spécifique aux arches EOS. Nous pouvons noter le sens d'une arche par une flèche comme illustré à la figure 6.9.

La table 6.2 représente un récapitulatif des différents cas où les arches i et j sont compatibles, à condition qu'elles soient exécutées dans le bon sens et avec la bonne chronologie. Pour chaque cas de compatibilité de la table 6.2, la 1re colonne du tableau donne les relations entre indices pour le schéma représenté dans la 2e colonne et pour le cas symétrique, c'est-à-dire si i est à la place de j et *vice versa*. La 3e colonne est réservée aux commentaires.

6.4.2.2 Définitions

Pour mieux cerner cette notion de sens et de chronologie, nous précisons la définition des arches EOS. Nous décomposons ces arches en trois facteurs.

a	$i_{IV} < j_I$ ou $j_{IV} < i_I$		i **précède** j
b	$j_{II} < i_I$ et $j_{III} > i_{IV}$ ou $i_{II} < j_I$ et $i_{III} > j_{IV}$		j **contient** i *j doit être générée avant i*
c	$i_{III} \leq j_I \leq i_{IV} \leq j_{II}$ ou $j_{III} \leq i_I \leq j_{IV} \leq i_{II}$		i **précède** j *ici les pieds peuvent être strictement superposés.*
c'	$i_{II} < j_I \leq i_{III} \leq i_{IV} \leq j_{II}$ ou $j_{II} < i_I \leq j_{III} \leq j_{IV} \leq i_{II}$		i **précède** j *i doit être générée avant j*
d	$j_I < i_I \leq j_{II} \leq i_{II}$ et $i_{IV} < j_{III}$ ou $i_I < j_I \leq i_{II} \leq j_{II}$ et $j_{IV} < i_{III}$		j **contient** i *j doit être générée avant i*
e	$j_I \leq i_I \leq j_{II} < i_{II}$ et $i_{IV} < j_{III}$ ou $i_I \leq j_I \leq i_{II} < j_{II}$ et $j_{IV} < i_{III}$		j **contient** i *j doit être générée avant i*
f	$i_I \leq j_I \leq j_{II} \leq i_{II}$ et $i_{IV} < j_{III}$ ou $j_I \leq i_I \leq i_{II} \leq j_{II}$ et $j_{IV} < i_{III}$		j **contient** i *j doit être générée avant i*
Ci-dessous, les symétriques des relations d, e et f (j **contient** i) :   			

Tab. 6.2 – *Les différents cas de compatibilité entre arches. Nous pouvons les diviser en deux groupes :* **précédence**, *l'une est à gauche de l'autre, et* **contenance**, *l'une contient l'autre.*

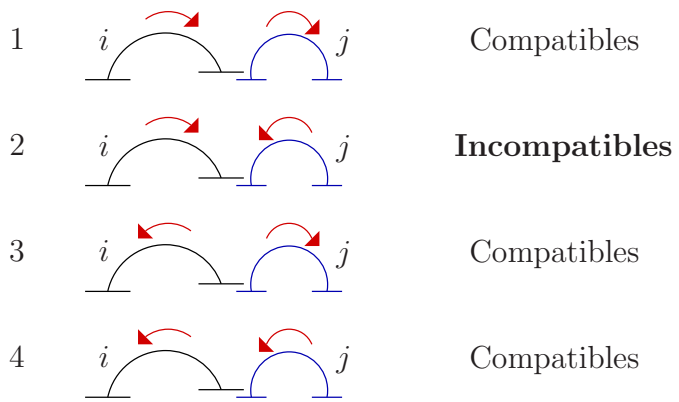


FIG. 6.9 – *Compatibilité entre arches selon leur sens, sous réserve d'une chronologie correcte : dans le cas* 1*, i doit être générée avant j, dans la cas* 4 *c'est le contraire, et dans les cas* 2 *et* 3 *la chronologie n'affecte pas la compatibilité.*

Définition 33 (Arche EOS) *Une arche* EOS *sur une séquence s de longueur n est un triplet de facteurs adjacents associé à une amplification d'ordre supérieur. Les trois facteurs d'une arche* EOS *sont nommés :*

- **pied source :** *c'est le motif copié par l'amplification associée à l'arche ;*
- **pied généré :** *c'est le résultat de l'amplification du pied source ;*
- **intérieur :** *il représente la partie de la séquence qui sera générée* après *l'amplification, il peut être vide. L'intérieur est situé entre le pied source et le pied généré.*

De la même manière que précédemment, le pied source est le facteur $s[i_I, i_{II}]$, le pied généré est le facteur $s[i_{III}, i_{IV}]$, avec $1 \leq i_I \leq i_{II} < i_{III} \leq i_{IV} \leq n$ et $s[i_I, i_{II}] = s[i_{III}, i_{IV}]$.

Une illustration d'une arche EOS et de ses facteurs est donnée à la figure 6.10. Ce découpage en trois facteurs induit directement le sens de l'arche – du pied source vers le pied généré. Donc chaque arche représentée par un arc correspond maintenant à deux triplets de facteurs, un pour chaque sens. Lorsque qu'une arche a est effectuée de la gauche vers la droite, nous la notons $\overrightarrow{a}$; si elle est effectuée dans l'autre sens, nous la notons $\overleftarrow{a}$.

On peut grouper ces trois facteurs en deux parties : la *partie source*, représentée en trait plein, qui est composée du pied source, et la *partie générée*, représentée en trait pointillé, qui est composée de l'intérieur et du pied généré, comme indiqué sur la figure 6.10. La partie source et la partie générée sont des facteurs de la séquence. Lors d'un alignement, seule la partie source de l'arche a est alignée avec la séquence d'en face ; la partie générée sera créée par l'évolution interne de l'arche a. Notez que ceci est vrai si et seulement si a n'est pas une arche interne à une arche déjà alignée. Si c'est le cas, alors aucune des parties de a n'est alignée avec la séquence d'en face.

Notation *Nous notons S_i la partie source de l'arche i et $G_i = I_i \cup P_i$ sa partie générée, qui est l'union de l'intérieur de i, I_i, et du pied généré, P_i.*

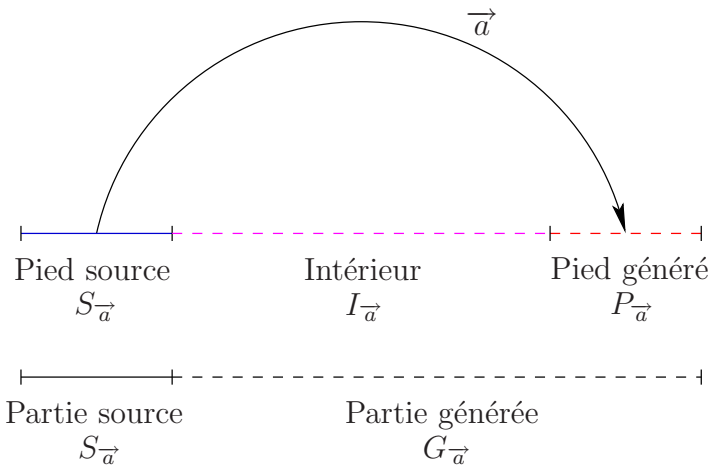


Fig. 6.10 – *Arche* eos *découpée en trois facteurs.*

Si l'on considère un listing dans lequel l'arche i est amplifiée, alors on ne peut pas amplifier par la suite une arche j telle que $G_j \cap S_i \neq \varnothing$ ou $G_j \cap P_i \neq \varnothing$. En effet, l'arche j ne peut pas générer des variants déjà présents dans la séquence (S_i) ou produits par i (P_i). L'intérieur de i par contre peut être généré par d'autres arches k telles que $G_k \cap (S_i \cup P_i) = \varnothing$ et $G_k \subseteq I_i$.

Définition 34 (Chronologiquement compatible) *Soient i et j deux arches* eos*. i est chronologiquement compatible avec j, noté $i\mathcal{K}j$, si $G_j \cap (S_i \cup P_i) = \varnothing$.*

Une arche i est chronologiquement compatible avec une arche j ($i\mathcal{K}j$) si j peut être générée après i dans un même listing.

6.4.3 La relation chronologiquement compatible

Dans cette section, nous détaillons les caractéristiques de la relation de compatibilité $\mathcal{K}$ que nous venons de définir.

Cette relation n'est :

- ni symétrique, c'est-à-dire $i\mathcal{K}j \not\Rightarrow j\mathcal{K}i$;
- ni antisymétrique, c'est-à-dire $i\mathcal{K}j \not\Rightarrow \text{non}(j\mathcal{K}i)$;
- ni transitive, c'est-à-dire $(i\mathcal{K}j \text{ et } j\mathcal{K}k) \not\Rightarrow i\mathcal{K}k$.

On peut diviser la relation chronologiquement compatible en deux relations que sont le *voisinage* et la *contenance* :

- x voisine y si et seulement si $x\mathcal{K}y$ et $G_y \cap I_x = \varnothing$;
- x contient y si et seulement si $x\mathcal{K}y$ et $G_y \subseteq I_x$.

Ces relations étendent les relations de précédence et de contenance vues au chapitre 4, page 94. Ces deux relations sont exclusives et représentent deux cas distincts de la relation $\mathcal{K}$. En effet, si $x\mathcal{K}y$, c'est-à-dire si $(G_y \cap (S_x \cup P_x) = \varnothing)$, alors soit $G_y \cap I_x = \varnothing$, soit

$G_y \subseteq I_x$. Il n'est pas possible que G_y chevauche I_x car cela impliquerait que $G_y \cap S_x \neq \varnothing$ ou $G_y \cap P_x \neq \varnothing$.

Lorsque x contient y, y ne contient pas x, x ne voisine pas y et y ne voisine pas x. La relation contient est antisymétrique. Lorsque x voisine y, x ne contient pas y, y ne contient pas x, mais on ne peut pas savoir si y voisine x ou pas. Donc la relation voisine n'est ni symétrique, ni antisymétrique, ceci est illustré à la figure 6.11.

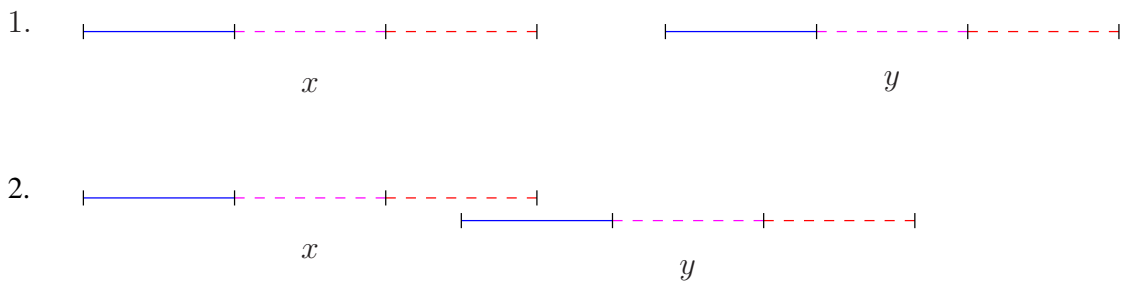


FIG. 6.11 – *Relation de voisinage entre arches. Dans le cas* 1, *x voisine y et y voisine x. Dans le cas* 2, *x voisine y mais y ne voisine pas x car $G_x \cap S_y \neq \varnothing$.*

Propriété 7 *La relation contient est transitive.*

Preuve (**Propriété 7**) En effet quelles que soient x, y et z, trois arches, si x contient y et y contient z, alors x contient z :

– x contient $y \Rightarrow G_y \cap (S_x \cup P_x) = \varnothing$ et $G_y \subseteq I_x$;

– y contient $z \Rightarrow G_z \cap (S_y \cup P_y) = \varnothing$ et $G_z \subseteq I_y$.

Comme $G_z \subseteq I_y \subset G_y$, si $G_z \cap P_x \neq \varnothing$, alors $G_y \cap P_x \neq \varnothing$ ce qui contredit l'hypothèse x contient y. De la même manière, si $G_z \cap S_x \neq \varnothing$, alors $G_y \cap S_x \neq \varnothing$ ce qui contredit également l'hypothèse x contient y. Donc $G_z \cap P_x = \varnothing$ et $G_z \cap S_x = \varnothing$. De plus on a $G_z \subseteq I_y \subset G_y \subseteq I_x$ donc $G_z \subset I_x$. D'où x contient z. □

Nous coupons en deux la relation de voisinage :

– *x est à gauche* de y si et seulement si x voisine y et $I_x < G_y$;

– *x est à droite* de y si et seulement si x voisine y et $G_y < I_x$.

La relation $<$ est une relation de précédence entre facteurs. Si l'on note g_w et d_w les positions gauche et droite d'un facteur w dans la séquence, alors $w < z$, où z est un facteur de la même séquence, signifie $d_w < g_z$.

Les relations « est à gauche » et « est à droite » sont contraires : lorsque x voisine y, si x n'est pas à gauche de y alors x est à droite de y et *vice versa*.

Propriété 8 *Les relations « est à gauche » et « est à droite » sont transitives.*

Preuve (**Propriété 8**) Prouvons le pour « est à gauche ». Quelles que soient x, y et z, trois arches, si x est à gauche de y et y est à gauche de z, alors x est à gauche de z :

– x est à gauche de $y \Rightarrow G_y \cap (S_x \cup P_x) = \varnothing$ et $I_x < G_y$;
– y est à gauche de $z \Rightarrow G_z \cap (S_y \cup P_y) = \varnothing$ et $I_y < G_z$.

On a $G_z > I_y$, or $I_y \subset G_y$ et $G_y > I_x$, donc $G_z > I_x$. Pour tout arche a, on a $S_a < I_a < P_a$ ou $P_a < I_a < S_a$ suivant qu'elle soit orientée dans un sens ou dans l'autre, c'est-à-dire suivant que l'on ait $\overrightarrow{a}$ ou $\overleftarrow{a}$.

Supposons $\overrightarrow{x}$, on a alors $S_x < I_x < P_x$ et comme $G_z > I_x$ on a directement $G_z \cap S_x = \varnothing$. Supposons que $G_z \cap P_x \neq \varnothing$, comme $S_x < I_x < P_x$ et $I_y < G_z$, alors :
– soit $I_y \cap P_x \neq \varnothing$ ce qui contredit l'hypothèse que x voisine y ;
– soit $I_y < P_x$, alors $I_x \not< G_y$, ce qui contredit l'hypothèse que x est à gauche de y.

Donc $G_z \cap P_x = \varnothing$ et x est à gauche de z.

Si on suppose $\overleftarrow{x}$, on a alors $P_x < I_x < S_x$ et comme $G_z > I_x$ on a directement $G_z \cap P_x = \varnothing$. On montre de la même manière que précédemment que $G_z \cap S_x = \varnothing$ et x est à gauche de z. □

Propriété 9 *Si x est à gauche de y et y contient z, alors x est à gauche de z.*

Preuve (**Propriété 9**) En effet :
– x est à gauche de $y \Rightarrow G_y \cap (S_x \cup P_x) = \varnothing$ et $I_x < G_y$;
– y contient $z \Rightarrow G_z \cap (S_y \cup P_y) = \varnothing$ et $G_z \subseteq I_y$.

On a $G_z \subseteq I_y \subset G_y$, par conséquent $G_z \subset G_y$. De plus :
– $G_y > I_x$ donc $G_z > I_x$;
– et $G_y \cap (S_x \cup P_x) = \varnothing$ donc $G_z \cap (S_x \cup P_x) = \varnothing$.

Par conséquent, x est à gauche de z. □

Propriété 10 *Si x est à droite de y et y contient z, alors x est à droite de z.*

La preuve de la propriété 10 est similaire à la preuve de la propriété 9.

6.4.4 Ensemble maximal d'arches EOS compatibles

Nous montrons dans cette section que trouver un ensemble maximal d'arches compatibles (au sens de $\mathcal{K}$) sur une séquence s nous permet de trouver un listing de coût minimal pour générer s à partir de $\varnothing$. Dans un premier temps nous définissons ce qu'est un ensemble maximal d'arches EOS compatibles et nous montrons ensuite comment trouver le listing.

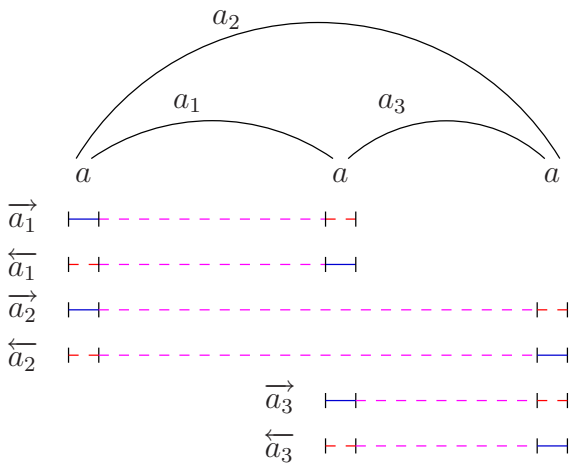
À quoi ressemble un ensemble maximal d'arches compatibles si l'on se donne $\mathcal{K}$ comme relation de compatibilité ? La compatibilité entre deux arches dépend de la chronologie de leur amplification. Chercher un ensemble maximal d'arches compatibles revient à chercher une suite ordonnée d'arches compatibles de poids maximal, où le poids d'une arche est l'économie apportée par son utilisation (cf. définition 31, page 134). Cette suite est ordonnée de manière à ce que toutes les arches des positions $[1 \dots k-1]$ soient chronologiquement compatibles avec l'arche à la position k, et ce pour toutes les positions k possibles. Nous appelons une telle suite une chronologie.

Définition 35 (Chronologie) *Une chronologie de s est une suite ordonnée d'arches* EOS *de s $(i_1, \ldots, i_k)$, telle que $\forall k, l \in [1..k]$ tels que $k < l$, i_k est chronologiquement compatible avec i_l.*

Une chronologie est donc une suite ordonnée d'arches associées à des amplifications qui peuvent s'effectuer dans un même listing. On remarque que si l'on dispose d'un listing, on peut en extraire une chronologie. Il suffit en effet de relever les événements d'amplification et leurs arches associées dans l'ordre induit par le listing.

Définition 36 (Poids d'une chronologie) *Le poids d'une chronologie est la somme des poids des arches qui la composent.*

EXEMPLE *Ancien cas limite dans la relation de compatibilité du chapitre 4 (p. 94)*



Les chronologies suivantes sont maximales : $(\overrightarrow{a_1}, \overrightarrow{a_3})$, $(\overleftarrow{a_1}, \overrightarrow{a_3})$, $(\overrightarrow{a_2}, \overrightarrow{a_1})$, $(\overrightarrow{a_2}, \overleftarrow{a_3})$, $(\overleftarrow{a_2}, \overleftarrow{a_3})$, $(\overleftarrow{a_2}, \overrightarrow{a_1})$, $(\overrightarrow{a_3}, \overleftarrow{a_1})$, $(\overleftarrow{a_3}, \overleftarrow{a_1})$. La compatibilité entre deux arches dépend de la chronologie de leur amplification. Par exemple, $\overrightarrow{a_1}$ est compatible avec $\overrightarrow{a_3}$ si elle est effectuée avant $\overrightarrow{a_3}$, mais incompatible si effectuée après.

Nous montrons maintenant que l'on peut trouver un listing de coût optimal qui génère s à partir de $\varnothing$, en utilisant une chronologie de poids maximal de s dans le but le maximiser le nombre d'économies apportées par les arches.

Notation *Soit i une arche* EOS*, nous notons $ordre(i)$, l'ordre de l'arche i. Le poids de i est donc $ordre(i) \times M$.*

Soit $K = (i_1, i_2, \ldots, i_k)$ une chronologie, nous notons $gain(K)$ la somme des ordres des arches contenues par K : $gain(K) = ordre(i_1) + ordre(i_2) + \ldots + ordre(i_k)$. Le poids de K est donc $gain(K) \times M$. On remarque que K est de poids maximal si et seulement si $gain(K)$ est maximal.

Nous décrivons maintenant comment générer une chaîne s à partir de $\varnothing$ en connaissant une chronologie de poids maximal sur s, notée $K = (i_1, i_2, \ldots, i_k)$. Nous avons besoin d'insérer un premier caractère dans $\varnothing$, nous insérons le premier caractère du pied source de i_1. Ensuite nous procédons de la manière suivante : pour chaque arche i_j, dans l'ordre induit par K, nous générons son pied source par A_M puis nous effectuons l'amplification d'ordre $ordre(i_j)$ associée à l'arche i_j. Lorsque nous avons passé toute les arches, nous générons les caractères manquants par A_M pour terminer s. Cette procédure est décrite dans l'algorithme 9.

Algorithme 9: *Génération de s à partir de $\varnothing$.*

Données : Une séquence s de longueur n et $K = (i_1, i_2, \ldots, i_k)$ une chronologie de poids maximal sur s.

Résultat : Un listing pour générer s à partir de $\varnothing$.

```
1 Insérer s[i_1_I] ;
2 pour j = 1 à k faire
3     si pied source de i_j non complet alors
4         Amplifier+Muter les caractères manquants ;
5     Effectuer l'amplification d'ordre ordre(i_j) associée à l'arche i_j ;
6 Amplifier+Muter les caractères manquants pour obtenir s ;
```

Le listing produit par l'algorithme 9 permet de générer s à partir de $\varnothing$. Nous appelons ce listing L_{alg} et nous notons son coût $c(L_{alg})$.

Propriété 11 *Le coût du listing produit par l'algorithme 9 à partir de la séquence s de longueur n et d'une chronologie de poids maximal sur s notée $K = (i_1, i_2, \ldots, i_k)$ est :*

$$c(L_{alg}) = I + (n-1) \times A + (n - 1 - gain(K)) \times M$$

Preuve (**Propriété 11**) Dans l'algorithme 9, les caractères de s sont produits seulement par insertion (ligne **1**), amplifications (ligne **5**) et amplifications+mutations (lignes **4** et **6**). L'algorithme n'effectue qu'une seule insertion, celle du premier caractère. Il effectue k amplifications qui produisent $ordre(i_1) + ordre(i_2) + \ldots + ordre(i_k) = gain(K)$ caractères. Les $(n - 1 - gain(K))$ caractères restants de s sont générés par des amplifications+mutations. Donc le coût du listing produit est de $I + gain(K) \times A + (n - 1 - gain(K)) \times (A + M)$, c'est-à-dire $I + (n-1) \times A + (n - 1 - gain(K)) \times M$. □

Notons que le coût du listing produit par l'algorithme 9 est minimal lorsque $gain(K)$ est maximal, donc lorsque K est une chronologie de poids maximal.

Théorème 7 (Coût d'un listing optimal générant s à partir de $\varnothing$) *Soit s une chaîne de longueur n, le coût d'un listing optimal générant s à partir de $\varnothing$ est de la forme :*

$$I + (n-1) \times A + b \times M.$$

Preuve (**Théorème 7**) Soit L_{opt} un listing optimal générant s à partir de $\varnothing$. Le listing L_{opt} contient au moins une insertion, celle d'un premier caractère dans $\varnothing$ et au plus une insertion, car il est optimal. En effet, toutes les insertions sauf la première peuvent être remplacées par une A_M de coût inférieur. Donc L_{opt} contient exactement une insertion. Tous les caractères de s sauf le premier sont générés par amplifications ou amplifications+mutations. L_{opt} ne contient ni délétion, ni contraction, ni M_C, donc son coût peut s'écrire sous la forme :

$$c(L_{opt}) = I + (n-1-b) \times A + b \times (A+M), b \in [0 \dots n-1].$$

d'où

$$c(L_{opt}) = I + (n-1) \times A + b \times M.$$

□

Théorème 8 (Optimalité de l'algorithme 9) *Le listing produit par l'algorithme 9 a un coût minimal.*

Preuve (**Théorème 8**) Pour prouver le théorème, nous montrons qu'un listing optimal L_{opt} ne peut pas avoir un coût inférieur à L_{alg}.

Dans L_{opt}, d'après le théorème 7, il y a $n-1-b$ variants de s produits par amplifications seulement. Par conséquent L_{opt} comporte $1 \leq k' \leq n-1-b$ événements d'amplification. Notons j_1, j_2, …, $j_{k'}$ ces amplifications dans l'ordre de L_{opt}. $K' = (j_1, j_2, \dots, j_{k'})$ est par définition une chronologie sur s et $gain(K') = n-1-b$ par construction, d'où $b = n-1-gain(K')$.

On a :

– $c(L_{opt}) < c(L_{alg})$ si et seulement si $b < n-1-gain(K)$;

– $b < n-1-gain(K) \Leftrightarrow n-1-gain(K') < n-1-gain(K) \Leftrightarrow gain(K) < gain(K')$.

Or par hypothèse, K est une chronologie de poids maximal, donc $gain(K) \not< gain(K')$, par conséquent $c(L_{opt}) \not< c(L_{alg})$. □

Nous venons de montrer que pour générer de manière optimale une séquence s à partir de $\varnothing$, nous avons besoin de trouver une chronologie de poids maximal sur s. Dans la section suivante, nous présentons les pistes que nous avons explorées pour essayer de résoudre ce problème.

6.5 Rapprochement avec des problèmes déjà étudiés

Notre but est de chercher une chronologie de poids maximal parmi les arches de s. Dans un premier temps, nous avons essayer d'adapter des problèmes de stable max à notre type de données (section 6.5.1). Nous avons ensuite cherché directement la chronologie maximale dans un graphe orienté selon la relation $\mathcal{K}$ (section 6.5.2), mais sans succès.

6.5.1 Problème de stable max

Considérons le graphe de la relation contraire à $\mathcal{K}$. Soient x et y deux sommets de ce graphe, alors il existe un arc de x vers y si et seulement si $G_y \cap (S_x \cup P_x) \neq \varnothing$. Il existe un arc de x vers y si et seulement si x n'est pas chronologiquement compatible avec y, c'est-à-dire si les arches qu'ils représentent « se croisent ». Deux arches « se croisent », si elles ne sont ni voisines, ni contenues l'une dans l'autre. Chercher une chronologie maximale peut utiliser la recherche d'un stable de poids maximal dans ce graphe.

6.5.1.1 Problème de 2-intervalles

Comme nous l'avons vu en présentant les travaux de Stéphane Vialette [Vialette, 2001] à la section 3.1.2, un 2-intervalle est un couple d'intervalles sur la droite réelle. Un 2-intervalle peut correspondre aux 2 pieds d'une arche eos.

Un graphe de 2-intervalles est le graphe d'intersection des 2-intervalles suivant un modèle de comparaison $\mathcal{R} \subseteq \{<, \sqsubset, \text{⨼}\}$. Ces relations sont décrites à la figure 3.3, page 68. Il existe une arête dans le graphe de 2-intervalles si les 2-intervalles correspondant aux sommets ne sont pas comparables par $\mathcal{R}$. Ainsi, trouver un stable max dans un tel graphe revient à trouver le nombre maximal de 2-intervalles comparables par $\mathcal{R}$.

Pour $\mathcal{R} = \{<, \sqsubset\}$, c'est-à-dire pour un ensemble contenant les relations de précédence et de contenance, Stéphane Vialette donne une solution en $O(n^2)$ pour trouver un stable max dans le graphe de 2-intervalles correspondant (voir section 3.1.2, page 67). Le modèle de compatibilité qu'il utilise n'autorise pas les intervalles composant deux 2-intervalles différents à se chevaucher, alors que notre relation de compatibilité oui. L'exemple de la figure 6.12 montre deux configurations dans lesquelles les 2-intervalles sont comparables dans notre modèle mais pas dans celui de Stéphane Vialette.

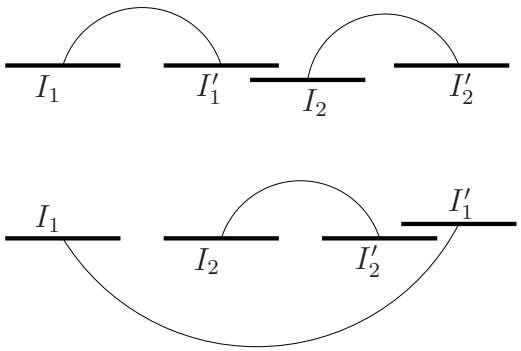


Fig. 6.12 – *Différence de la relation de compatibilité entre deux 2-intervalles.*

Cette différence est un obstacle de base pour adapter son modèle de comparaison au nôtre. Il nous est donc impossible d'utiliser son algorithme.

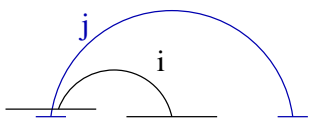
6.5.1.2 Problème de trapézoïdes circulaires

Les graphes de trapézoïdes circulaires sont une généralisation des graphes de chevauchement. Pour résoudre le problème du stable max dans un graphe d'intersection de trapézoïdes circulaires, un nouveau type de graphe est introduit par [Felsner et al., 1997] : les graphes de croisement (voir section 3.1.3, page 69).

Ces graphes sont définis sur des paires d'intervalles, ou doubles intervalles. Un double intervalle est une paire (I_1, I_2) d'intervalles sur la droite réelle, telle que I_2 est un sous-intervalle de I_1, c'est-à-dire $I_2 \subset I_1$. Deux doubles intervalles *se croisent* s'ils ne sont ni précédents, ni que l'un d'eux est contenu dans l'autre. $\mathcal{G} = (X, E)$ est un graphe de croisement si ses sommets peuvent être mis en correspondance un à un avec un ensemble de doubles intervalles tel qu'il existe une arête entre deux sommets de $\mathcal{G}$ si et seulement si leurs doubles intervalles correspondants se croisent.

Les arches EOS peuvent être représentées par des doubles intervalles : à une arche EOS i, nous pouvons associer le double intervalle $([i_I, i_{IV}], [i_{II}, i_{III}])$. En effet, par construction on a directement $[i_{II}, i_{III}] \subset [i_I, i_{IV}]$, et donc $([i_I, i_{IV}], [i_{II}, i_{III}])$ est un double intervalle naturellement associé à l'arche i.

Il existe un algorithme en $O(n^2)$ pour trouver un stable max dans un graphe de croisement $\mathcal{G} = (V, E)$ induit par une famille $\mathcal{F}$ de doubles intervalles (algorithme 2, page 71). Cet algorithme construit un ordre de contenance et un ordre de précédence. Nous n'avons pas réussi à adapter ces structures à nos données, car elles impliquent que ce qui ce situe à l'intérieur d'un double intervalle n'interagit pas avec ce qui ce trouve à l'extérieur, or dans notre cas c'est possible :



j contient i

Dans ce schéma, i est contenue dans j mais peut interagir avec des arches extérieures à j.

6.5.2 Problème de graphe orienté

La relation de compatibilité $\mathcal{K}$ entre arches EOS est orientée, nous nous intéressons donc aux graphes orientés.

Nous pouvons associer à chaque arche EOS d'une séquence s de longueur n un sommet dans un graphe orienté $\mathcal{D} = (V, A)$. Nous appelons ce graphe le *graphe de compatibilité* de s. L'ensemble des arcs A code la relation de compatibilité entre arches : il existe un arc d'un sommet x à un sommet y si l'arche associée à x est chronologiquement compatible à l'arche associée à y. Si l'on donne aux arches le même nom que celui des sommets auxquels elles sont associées, alors :

$$(x, y) \in A \iff x \mathcal{K} y$$

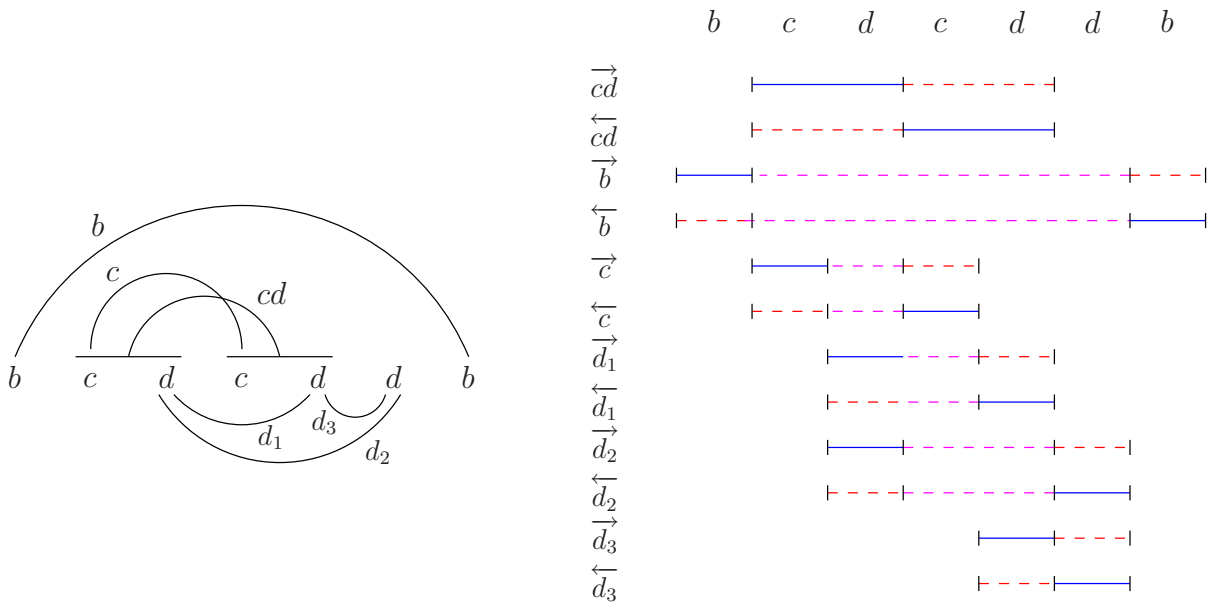


Fig. 6.13 – *Transformation d'arches en triplets de facteurs.*

Les sommets du graphe de compatibilité sont valués par le poids de l'arche qu'ils représentent.

Un chronologie correspond dans un graphe orienté à un chemin simple transitif. Un chemin simple est une suite ordonnée $(x_1, ..., x_n)$ de sommets reliés par des arcs telle que $x_i = x_j \Rightarrow i = j$. Un chemin $(x_1, x_2, \ldots, x_k)$ est transitif si $\forall i, j$ tels que $1 \leq i < j \leq k, (x_i, x_j) \in A$. La chronologie $K = (i_1, i_2, \ldots, i_k)$ correspond au chemin $(i_1, i_2, \ldots, i_k)$ dans $\mathcal{D}$, ce chemin est simple car il ne peut pas y avoir deux fois la même arche dans K et il est transitif par définition de la chronologie.

Trouver une chronologie de poids maximal est donc équivalent à trouver un chemin transitif simple de poids maximal dans $\mathcal{D}$.

6.5.2.1 Exemple de graphe de compatibilité

Construisons un graphe de compatibilité sur un exemple $s = bcdcddb$ (cette séquence est un peu moins complexe que celle de la figure 6.8, page 140). Dans un premier temps, nous transformons chaque arche de s en deux triplets de facteurs (un pour chaque sens). Ceci est montré à la figure 6.13. Toutes les arches sont de poids 1, sauf les arches de motif cd qui sont de poids 2. La figure 6.14 montre le graphe de compatibilité associé $\mathcal{D} = (V, A)$ et correspond à une chronologie de poids maximal.

Dans cet exemple, un chemin transitif de poids maximal du graphe de compatibilité est $(\overrightarrow{b}, \overrightarrow{cd}, \overrightarrow{d_3})$ de poids 4.

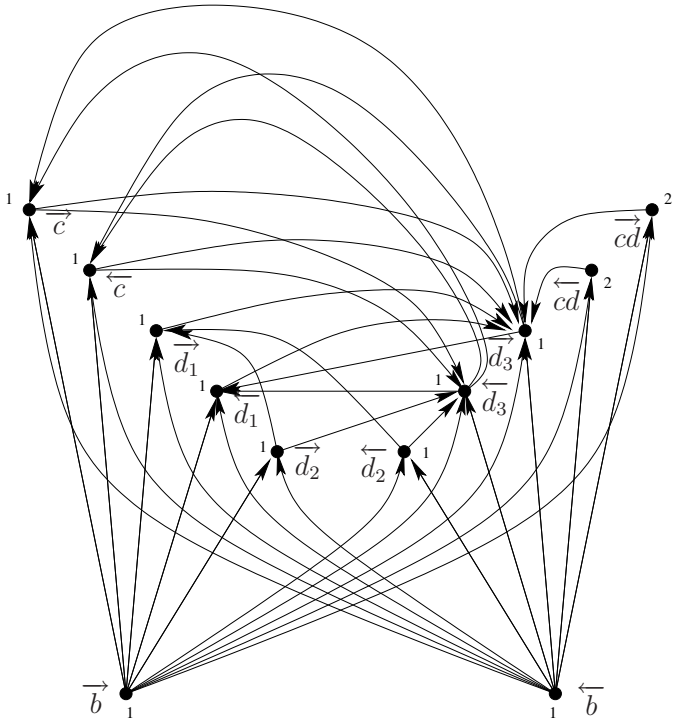


FIG. 6.14 – *Graphe de compatibilité associé à la séquence de la figure 6.13.*

6.5.2.2 Les pistes

Dans cette section, nous évoquons deux des pistes que nous avons explorées.

Sous-graphe transitif maximal Le but est d'extraire un sous graphe transitif maximal du graphe de compatibilité, puis de chercher le plus long chemin dans ce sous-graphe. Cependant le problème de décision TRANSITIVE SUBGRAPH associé à la recherche d'un sous graphe transitif maximal est NP-complet :

TRANSITIVE SUBGRAPH

Données : Un graphe orienté $\mathcal{D} = (V, A)$, un entier positif $k \leq |A|$.

Résultat : Existe-t-il un sous-ensemble $A' \subseteq A$, avec $|A'| \geq k$, tel que $\mathcal{D}' = (V, A')$ est transitif, c'est-à-dire pour toutes les paires $u, v \in V$, s'il existe un sommet $w \in V$ pour lequel $(u, w), (w, v) \in A'$, alors $(u, v) \in A'$?

Le problème TRANSITIVE SUBGRAPH a été montré NP-complet par M. Yannakakis, dans [Yannakakis, 1978], par transformation à partir du problème BIPARTITE SUBGRAPH sans triangle [Garey et Johnson, 1979]. Un graphe est biparti si ses sommets peuvent être divisés en deux ensembles X et Y, de sorte que toutes les arêtes du graphe relient un sommet dans X à un sommet dans Y. Le problème BIPARTITE SUBGRAPH est de trouver pour un graphe $\mathcal{G} = (V, E)$ un sous-ensemble d'arêtes E' tel que le graphe $\mathcal{G} = (V, E')$ est biparti.

Malgré les propriétés de notre graphe, nous n'avons pas trouvé de solution polynômiale au problème du sous-graphe transitif maximal.

Chercher le chemin simple transitif de poids maximal Le problème du plus long chemin entre deux sommets d'un graphe est identique au problème Longest Path et il est NP-complet pour les graphes quelconques [Garey et Johnson, 1979]. Cependant, si l'on considère le problème Longest Path sur les graphes orientés acycliques, nommés GOA[1], la complexité en temps est $O(|V| + |A|)$ [Cormen et al., 1994].

Mais la recherche d'un plus long chemin simple transitif est également NP-complet même pour les GOA. Ceci a été montré par M. Habib en faisant une réduction à partir du problème 3-SAT [Habib, 2003]. Le problème du plus long chemin simple transitif est appelé aussi sous-ordre total maximal.

* *
*

Ce chapitre apporte une modélisation des arches engendrées par l'extension des opérations d'amplification et de contraction du modèle sse. Nous avons dénombré ce nouveau type d'arches et soulevé des questions quant à leur coûts. Nous avons montré que, sous le modèle esse restreint aux arches eos, trouver le coût minimal de génération d'une chaîne s à partir de $\varnothing$ nécessite le calcul d'une chronologie maximale. Les rapprochements aux problèmes déjà étudiés pour calculer cette chronologie peuvent donner des pistes pour des algorithmes heuristiques, par exemple nous pourrions adapter un algorithme d'approximation de 3-SAT pour approximer un plus long chemin transitif dans le graphe de compatibilité.

[1]Les GOA sont aussi connus sous l'acronyme anglophone DAG pour *Directed Acyclic Graph*.

Deuxième partie

Génétique

Chapitre 7

État de l'art

Sommaire

7.1 Instabilité des minisatellites 157
7.2 Calcul de distances génétiques à partir de cartes de minisatellite 159
7.3 Étude du minisatellite MSY1 161

Nous avons décrit d'une manière générale dans le premier chapitre ce que sont les répétitions en tandem et, en particulier, les minisatellites (section 1.4). Nous décrivons ici les caractéristiques des processus de mutations impliqués dans le polymorphisme des minisatellites et notamment du minisatellite MSY1. Les travaux évoqués dans ce chapitre, (Jeffreys et al., 1997), (Bríon et al., 2002), (Jobling et al., 1998), (Bouzekri et al., 1998) et (Andreassen et al., 2002), utilisent la méthode MVR-PCR, mise au point par Jeffreys et ses collaborateurs (cf. page 30) (Jeffreys et al., 1991). Ce procédé fournit une *carte de minisatellite*, c'est-à-dire une séquence de symboles, où chaque symbole correspond à un variant de l'unité répétée. Ces cartes, également appelées codes MVR, permettent une étude plus approfondie des minisatellites.

Nous examinons dans cette section les hypothèses proposées pour expliquer l'instabilité des minisatellites, puis nous détaillons une méthode permettant de mesurer une distance entre cartes de minisatellite. Enfin, nous nous intéressons plus particulièrement au minisatellite MSY1, situé sur le chromosome humain Y, que nous étudions avec notre algorithme d'alignement au chapitre 8.

7.1 Instabilité des minisatellites

Comme nous l'avons vu dans le chapitre 1, certains minisatellites évoluent rapidement. Dans l'article (Jeffreys et al., 1997), Jeffreys et ses collaborateurs étudient les processus impliqués dans la mutation des minisatellites. Cet article est une compilation de nombreux travaux.

(Jeffreys et al., 1997)

Les auteurs veulent étudier l'instabilité des minisatellites et comprendre les processus de mutation impliqués. Pour cela, ils cherchent à détecter de nouveaux mutants (ou nouveaux allèles), principalement sur des minisatellites humains. L'objectif idéal étant d'avoir la carte du minisatellite avant et après l'événement de mutation. Les auteurs cherchent également à expliquer un phénomène observé : la polarité. En effet, les variations entre allèles sont plus fréquemment observées d'un côté de la répétition en tandem (en $5'$ ou en $3'$). Enfin, ils démontrent que les radiations ionisantes induisent un taux élevé de mutations dans les minisatellites.

Processus de mutation L'instabilité des minisatellites semble impliquer deux processus de mutation distincts suivant qu'ils se situent dans les cellules somatiques ou germinales. La méthode classique pour détecter les mutations germinales dans les minisatellites humains est l'analyse généalogique[1]. Détecter les mutations somatiques, en pratique, est plus difficile car l'instabilité somatique est 200 à 1000 fois plus faible que l'instabilité germinale. Les mutations maternelles sont moins bien comprises car les minisatellites mutent plus dans les lignées mâles, exception faite de MS1. Une des hypothèses expliquant l'instabilité des minisatellites est qu'ils marquent les sites chromosomaux impliqués activement dans la recherche d'homologie entre chromosomes homologues lors de l'appariement à la méiose et sont ainsi impliqués dans les événements de recombinaison.

Les auteurs donnent des hypothèses pour le modèle des mutations dans les spermatozoïdes (mutations germinales). L'instabilité des minisatellites dans les spermatozoïdes a été caractérisée pour quatre loci humains différents : MS205, MS32, B6.7 et CEB1. Des traits communs ont émergés de ces quatre sites :

- la fréquence des mutants est très supérieure à celle trouvée dans le sang (pour MS32 elle est 250 fois plus élevée) ;
- les mutations impliquent un gain ou une perte d'un petit nombre d'unités (dans 80 % des cas inférieures à 5 unités) ;
- la distribution en taille des mutants provenant d'un allèle donné semble être constante, indépendamment de la longueur de l'allèle (ceci s'est avéré faux pour le minisatellite CEB1 (Buard et al., 1998)) ;
- dans ces quatre loci, on trouve un biais vers le gain plutôt que la perte d'unités.

Ce biais en gain d'unité pose la question de la protection de tels sites contre les expansions infinies. Dans cet article, plusieurs hypothèses sont faites, comme l'occurrence de délétions compensatoires majeures. Cette question a été résolue ultérieurement par Buard et ses collaborateurs (Buard et al., 2002).

Polarité La polarité des mutations suggère que l'instabilité est d'une certaine manière régulée par de l'ADN flanquant, c'est-à-dire de l'ADN se trouvant proche du minisatellite. Cet ADN flanquant jouerait le rôle d'initiateur et pourrait servir pour diriger les cassures

[1] *Pedigree analysis* en anglais.

ou entailles dans le début du minisatellite et pour initier la mutation. Cette hypothèse implique que l'instabilité germinale n'est pas une propriété intrinsèque du minisatellite mais est conférée par des régulateurs flanquants. Ce modèle induit plusieurs hypothèses que les auteurs ont testées :

1. Le taux de mutation est constant quelle que soit la longueur du minisatellite : les auteurs pensent que cette hypothèse est correcte pour MS32 et MS205 (cependant elles s'est avérée fausse, en particulier pour CEB1 et B6.7 (Buard et al., 1998)) ;
2. Des substitutions de nucléotides dans l'ADN flanquant affectant l'initiateur influencent le taux de mutation : les auteurs trouvent exactement ce phénomène pour MS32, ils le constatent également chez MS205 et g3 ;
3. Différents loci sujets à la conversion polarisée ont des motifs d'ADN flanquant en commun : les auteurs ont cherché à le vérifier pour MS31, MS205, CEB1 et MS32, mais n'ont rien trouvé.

Radiations Les minisatellites ne montrent pas seulement une haute fréquence de mutations spontanées dans les cellules germinales mais apparaissent aussi comme très sensibles aux mutations induites par des radiations ionisantes. Ceci est constaté à la fois dans des souris irradiées expérimentalement et chez les populations humaines exposées suite à la catastrophe de Chernobyl. Les auteurs ont également fait des études sur les enfants des survivants d'Hiroshima et de Nagasaki et ont constaté que les conséquences biologiques sont différentes entre l'exposition chronique à un environnement pollué et une dose de radiation due à une bombe atomique. Les mécanismes des mutations induites par des radiations restent énigmatiques.

7.2 Calcul de distances génétiques à partir de cartes de minisatellite

Nous avons décrit dans le chapitre 4 une méthode pour calculer une distance entre cartes de minisatellite. Parallèlement à nos travaux, une autre équipe de chercheurs a également mis au point une technique de comparaison de cartes de minisatellite.

(Bríon et al., 2002)

Dans cet article, les auteurs proposent une méthode pour mesurer les variations de structures des minisatellites et pour calculer les distances génétiques en utilisant les cartes obtenues par MVR-PCR. Leur méthode est basée sur les similitudes statistiques des motifs. Ils comparent des populations étroitement liées. Par exemple, leur méthode a été appliquée pour analyser les variations du minisatellite MSY1 dans cinq jeux de données de populations européennes et nord africaines.

Dans leur méthode, chaque carte de minisatellite est considérée comme un vecteur de dimension n, $x = (x_1, x_2, \ldots, x_n)$ tel que chaque x_i est le $i^{\text{ème}}$ symbole de la carte. Ils

x_1	x_2	x_3	...	x_n			
y_1	y_2	y_3	...	y_n	y_{n+1}	...	y_m

Les deux séquences sont calées à gauche

x_1	x_2	x_3	x_4	x_5	...	x_n			
		y_1	y_2	y_3	...	y_{n-2}	...	y_m	

Décalage de y lorsque $h = 2$

FIG. 7.1 – *Exemple de décalages utilisés dans le calcul de la distance de Bríon.*

développent d'abord un estimateur de distance entre x de longueur n et y de longueur m. Ils définissent $\rho_h(x, y)$ comme le nombre d'appariements exacts entre x et y lorsque y a été décalée de h positions après l'avoir calée avec x sur la gauche (cf. figure 7.1), où h peut être positif ou négatif suivant que l'on décale vers la gauche ou la droite, $h \in [-m+1, \ldots, -1, 0, 1, \ldots, n-1]$. $\rho_h(x, y)$ est donnée par la formule :

$$\rho_h(x,y) = \sum_{k=\max(1,h+1)}^{\min(n,m+h)} f(k) \qquad \text{où } f(k) = \begin{cases} 1 \text{ si } x_k = y_{k-h} \\ 0 \text{ sinon} \end{cases}$$

Ils définissent ensuite une similarité entre les deux cartes comme la somme des $\rho_h(x, y)$ pour $h \in [-m+1, n-1]$. Les auteurs sont conscients du fait que cette similarité ne prend pas en compte l'ordre dans lequel apparaissent les répétitions. Pour essayer de pallier cela, ils donnent progressivement moins de poids aux appariements exacts (collectés dans $\rho_h(x, y)$) associés aux grands décalages. En pratique, ils considèrent des fonctions décroissantes $\varphi(k)$ du paramètre de décalage k positif, $k = |h|$. La mesure de similarité est alors redéfinie en multipliant les quantités $\rho_h(x, y)$ par les poids $\frac{1}{2}(\varphi(|h|) + \varphi(||n-m| - h|))$. $|h|$ est la valeur absolue du décalage nécessaire pour caler les deux séquences à partir de la gauche, tandis que $||n-m| - h|$[2] est la valeur absolue du décalage nécessaire pour les caler à partir de la droite. Les auteurs choisissent pour φ une fonction exponentielle $\varphi(k) = \phi^k, \forall \phi,\ 0 < \phi < 1$. La mesure de similarité est donc donnée par la formule :

$$\rho(x,y) = \frac{1}{2} \sum_{h=-m+1}^{n-1} \rho_h(x,y)(\varphi(|h|) + \varphi(||n-m| - h|))$$

Les auteurs cherchent à définir une distance entre x et y à partir de la mesure de similarité. Ils la définissent de la manière suivante : $d(x, y) = \rho(x, x) + \rho(y, y) - 2\rho(x, y)$. Un inconvénient de cette définition est qu'elle tend à donner des plus grandes valeurs à des séquences de grandes tailles. Pour pallier ce problème, Bríon et ses collaborateurs divisent l'expression globale par la plus grande valeur de cette quantité (quand x et y sont complètement différents, c'est-à-dire quand $\rho(x, y) = 0$). Cela donne la définition finale de leur distance entre deux séquences :

$$d(x,y) = \frac{\rho(x,x) + \rho(y,y) - 2\rho(x,y)}{\rho(x,x) + \rho(y,y)}$$

[2]L'expression originale de l'article est $|n - m - h|$, mais elle est correcte seulement si $n < m$.

Il est clair que $d(x, y) = 1$ si x et y sont complètement différents et $d(x, y) = 0$ si x et y sont identiques. Mais $d(x, y) = 0$ peut arriver également lorsque les séquences ne sont pas identiques. C'est une des raisons pour laquelle $d(x, y)$ n'est pas une distance au sens mathématique (cf. chapitre 2, section 2.2.3).

Une fois qu'ils ont établi leur « pseudo-distance », ils définissent une mesure de variabilité pour une population donnée. Ils considèrent une population de séquences de MSY1, telle que la probabilité que la séquence x apparaisse est p_x, c'est-à-dire que la densité de probabilité de la population P est donnée par $\{p_x \mid x \in P\}$. Ils définissent alors la variabilité à l'intérieur d'une population, puis entre deux populations en se basant sur la distance d. Ils définissent donc des distances intra- et inter-population. Ils appliquent ensuite leur méthode à la comparaison de cartes du minisatellite MSY1. Leur jeu de données est composé de cinq groupes de populations : Galiciens, Britanniques, Valenciens, Basques et Maghrébins. L'arbre qu'ils reconstruisent à partir de leur matrice de distance et de la méthode Neighbor-Joining sépare la population nord africaine des quatre populations européennes.

Les auteurs définissent dans cet article une mesure de similarité statistique. Elle calcule la somme pondérée du nombre de variants partagés quand les deux cartes sont calées à différentes positions relatives. Leur mesure et la distance qui en découle dépendent fortement de la fonction de poids utilisée. La distance ne remplit pas les conditions d'une distance métrique ; c'est un inconvénient majeur pour la reconstruction phylogénique. Elle ne prend pas en compte précisément les variations complètes puisqu'elle ne calcule pas un alignement. De plus, leur distance est difficile à interpréter car elle n'a pas de liaison avec des événements biologiques.

7.3 Étude du minisatellite MSY1

Les articles de cette section décrivent le minisatellite MSY1 que nous avons utilisé pour nos expérimentations. Nous donnons l'entrée de MSY1 dans la banque de données GenBank dans l'annexe B, page 193. Nous rappelons que les minisatellites sont des séquences répétées en tandem dont le motif de base varie entre 10 et 50 paires de bases[3], habituellement riches en GC, trouvés dans la plupart des génomes des eucaryotes supérieurs. Il y a des variations dans le nombre de leurs unités, ainsi que dans leur séquence ; ces variations peuvent être étudiées de manière pratique avec la méthode MVR-PCR.

(Jobling et al., 1998)

Les auteurs décrivent dans cet article l'isolation et la caractérisation du premier minisatellite haploïde découvert sur le chromosome Y humain, MSY1. Ils présentent la génération des cartes pour les chromosomes Y par MVR-PCR et une étude générale sur la diversité de MSY1. Ce qui nous intéresse plus particulièrement est la structure du minisatellite MSY1.

[3]Intervalle donné par les auteurs des articles de cette section.

Le chromosome Y est hérité du père et échappe sur la plupart de sa longueur à la recombinaison (cf. section 1.5.5, page 27). Les polymorphismes de l'ADN du chromosome Y contiennent donc de l'information sur l'histoire des lignées paternelles. Cette information, combinée à des études utilisant l'ADN mitochondrial hérité de la mère et des marqueurs hérités des deux parents, contribue à la compréhension de l'histoire et de l'évolution humaine. Plusieurs types de marqueurs sont disponibles sur le chromosome Y : des substitutions de bases et quelques indels, qui peuvent être vus comme des événements uniques, mais aussi des microsatellites qui mutent plus rapidement que ces autres marqueurs. Cette diversité de vitesse d'évolution peut être exploitée pour répondre à des questions sur différentes échelles de temps.

C'est l'analyse des marqueurs polymorphes qui permet d'étudier les différences génétiques entre les individus ou les populations. Il est possible d'associer à un individu un ensemble de polymorphismes sur le même chromosome que l'on nomme haplotype. Les polymorphismes évoluant lentement sur le chromosome Y, tels les SNPs, définissent de tels haplotypes. Les haplotypes les plus voisins sont regroupés en *haplogroupes*. Les microsatellites peuvent être utilisés pour évaluer la diversité à l'intérieur de ces haplogroupes ou entre des populations très proches.

En plus de son potentiel comme marqueur évolutif, le chromosome Y est intéressant à cause des conséquences de son haploïdie sur ses processus de mutation. La partie non recombinante du chromosome Y échappe aux événements dans lesquels l'information de la séquence est échangée entre les allèles de différentes lignées et par conséquent, le chromosome Y fournit un support pour l'étude des processus de mutation intra-allélique.

Les répétitions de MSY1 sont riches en AT ($75-80$ %). Cinq variants de l'unité répétée, numérotés de 1 à 5, ont été identifiés. Les codes des cinq variants sont montrés à la figure 7.2, partie haute. Les répétitions de 1 à 4 diffèrent à deux positions : une transition C/T à la position 3 et une transversion C/G à la position 13. Le type 5 diffère du type 2 par une transversion à la position 21.

Les auteurs ont étudiés pour cet article 465 chromosomes Y provenant de populations réparties à travers le monde, représentant 20 des 23 haplogroupes définis à l'époque[4]. Les allèles ont pu être cartographiés dans leur totalité. Les auteurs ont donc pu examiner la diversité des longueurs et des structures internes. Le résultat les a surpris : les 465 allèles ont tous entre 48 et 114 répétitions et 83 % d'entre eux, entre 58 et 77. À l'intérieur des haplogroupes, la distribution des longueurs est encore plus resserrée : les 11 chromosomes de l'haplogroupe 3, par exemple, ont entre 89 et 93 répétitions. Cette distribution de faible variance est en contraste avec celle observée chez des minisatellites autosomaux : par exemple MS32 a entre 12 et 800 répétitions ou même plus. Ils pensent que les insertions ou délétions d'un grand nombre de répétitions doivent être rares et que les mutations les plus communes doivent probablement impliquer un petit nombre de répétitions, voire une seule. Cela a été confirmé dans (Andreassen et al., 2002) (cf. dernier paragraphe).

Malgré cette contrainte de longueur, ils observent un fort degré de diversité dans la structure interne : parmi les 465 chromosomes, on compte 386 allèles différents, impliquant

[4]Il y en a à l'heure actuelle 153 (The Y Chromosome Consortium, 2002).

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
○	Type 1 :	*C*	*A*	*C*	*A*	*A*	*T*	*A*	*T*	*A*	*C*	*A*	*T*	*G*	*A*	*T*	*G*	*T*	*A*	*T*	*A*	*T*	*T*	*A*	*T*	*A*
✪	Type 2 :	.	.	*T*	.	.	.	.	.	.	.	.	.	*G*	.	.	.	.	.	.	.	.	.	.	.	.
●	Type 3 :	.	.	*C*	.	.	.	.	.	.	.	.	.	*C*	.	.	.	.	.	.	.	.	.	.	.	.
●	Type 4 :	.	.	*T*	.	.	.	.	.	.	.	.	.	*C*	.	.	.	.	.	.	.	.	.	.	.	.
	Type 5 :	.	.	*T*	.	.	.	.	.	.	.	.	.	*G*	.	.	.	.	.	.	.	*A*	.	.	.	.

Code	Pop.	Hg	Carte MSY1
*m*19	Anglais	2	
*m*1	Anglais	1	
*m*110	Indien	3	
*m*47	Pygmée	6	
*m*82	San	7	
*m*71	Kényan	8	
*m*121	Finnois	16	
*m*707	Maya	18	
*m*65	Japonais	20	
*m*6	Anglais	21	
*m*125	Berbère	21	
*m*715	Bantu	21	

5′ 3′

Type 1 : ○ Type 2 : ✪ Type 3 : ● Type 4 : ● Nul : ⊙

FIG. 7.2 – *En haut, le code des cinq variants séquencés du minisatellite MSY1. En bas, un tableau présentant quelques exemples de cartes de MSY1. La* 1^{re} *colonne donne le code de l'individu, la* 2^e*, sa population d'origine, la* 3^e*, son haplogroupe et la* 4^e *la carte de son minisatellite MSY1. Le symbole* ⊙ *indique que le variant ne fait pas partie des cinq types connus (Jobling et al., 1998).*

un taux de mutation très élevé pour MSY1.

Les types de répétitions ont tendance à se partitionner en bloc pouvant aller jusqu'à 64 motifs identiques et adjacents ; ceci est à opposer aux minisatellites autosomaux dont les répétitions sont hautement entremêlées. Cette différence d'organisation reflète des différences dans les processus de mutation. Des exemples de cartes de minisatellite sont donnés à la figure 7.2, partie basse. Les différents types de répétitions ne sont pas distribués au hasard à travers les allèles. À trois exceptions près, les allèles ont un type 4 à leur extrémité $3'$ et jamais à l'extrémité $5'$. Les types 1 ne sont jamais vus à l'extrémité $3'$ ni adjacents à des types 4. Les auteurs supposent que cette organisation est due à la génération séquentielle des répétitions à l'extrémité $5'$ des allèles ancêtres. Cela suggère, comme pour les minisatellites autosomaux, que le processus de mutation est polarisé. Les répétitions de type nul, notées $\odot$, signifient que la répétition ne correspond pas à l'un des cinq types précédemment décrits.

Des cartes identiques apparaissent usuellement dans les mêmes populations ou haplogroupes. Quelques exceptions pourraient être des cas d'**homoplasie**. L'homoplasie est le fait d'avoir des séquences présentant des états identiques mais ayant subi différentes évolutions. Les populations contenant un nombre significatif de cartes partagées sont des cas manifestes d'isolation culturelle ou géographique, les exemples cités dans l'article sont les habitants de l'île de Cook (Polynésie), les Basques et en particulier les Suruis (indiens du Brésil).

Les auteurs s'intéressent plus particulièrement aux structures modulaires. Par exemple la structure $(1, 3, 4)$ est un bloc de répétitions de type 1 en $5'$, suivi par un bloc central de type 3 et un bloc de type 4 en $3'$. Ils supposent que la génération d'un nouveau bloc de répétition est moins probable que l'expansion ou la contraction d'un bloc existant, ceci est corroboré par des observations sur MSY1. Ils observent 45 structures modulaires différentes (incluant les répétitions nulles comme un type), dont $(1, 3, 4)$ pour 169 chromosomes et $(3, 1, 3, 4)$ pour 122 chromosomes. Seulement 19 structures modulaires n'apparaissent qu'une fois.

Les minisatellites sont habituellement riches en GC, MSY1 est quant à lui riche en AT ; d'autres minisatellites autosomaux partagent cette caractéristique, COL2A1, ApoB et FRA16B. Ils ont des traits communs avec MSY1 et peuvent certainement partager des processus de mutation.

Les auteurs pensent que les marqueurs multi-alléliques comme MSY1 offrent une meilleure opportunité pour étudier la diversité du chromosome Y que les marqueurs « événement unique » qui définissent les haplogroupes.

Des recherches antérieures ont montré que le chromosome Y présentait moins de diversité que les autres parties du génome. Les substitutions sur le Y sont rares, mais sont très utiles pour définir les haplogroupes et pour reconstruire des arbres. Bien que fiables, ces arbres ont une faible résolution. Des marqueurs plus variables, comme les micro- et minisatellites, révèlent la diversité à l'intérieur des haplogroupes et peuvent être utilisés pour étudier les phénomènes de micro-évolution. MSY1 est le seul « système » du chromosome Y où les caractéristiques d'un grand nombre de mutations peuvent être étudiées en

détail. Cela fournit un outil unique et puissant pour l'étude des mutations dans un système haploïde et pour la datation des lignées paternelles.

(Bouzekri et al., 1998)

Cet article se situe dans la continuité du précédent. Il a été publié à sa suite dans le même numéro de la revue *Human Molecular Genetics*. Ils traitent tous deux du minisatellite MSY1.

Lors de l'analyse de certains allèles de MSY1 avec la méthode MVR-PCR à trois états (les plus courants : types 1, 3 et 4), Bouzekri et ses collaborateurs ont remarqué l'existence de nouveaux types de répétition. En séquençant complètement un de ces allèles, ils ont découvert que ces nouveaux types de répétitions incluent, par rapport aux types 1, 3 et 4, une substitution de base supplémentaire $T \rightarrow C$ à la position 6. Ils ont appelés ces nouveaux variants, 1a, 3a et 4a, leur séquence est donnée à la figure 7.3, partie haute. Ils ont donc mis au point une nouvelle MVR-PCR permettant de détecter ces nouveaux variants, et ce sont aperçus que les nouveaux variants sont confinés aux allèles de l'haplogroupe 8, regroupant des populations africaines. La diversité de cet haplogroupe est faible, la longueur de ses allèles varie entre 54 et 67 répétitions.

Dans l'haplogroupe 8, la transition $T \rightarrow C$ est propagée à travers tout le minisatellite, à l'exception d'une ou deux répétitions en $3'$, mais sans éliminer les types 1, 3 et 4 déjà existants. Des exemples de cartes sont donnés à la figure 7.3, partie basse. On y retrouve l'individu Kényan $m71$ dont la carte n'avait pas pu être déterminée lors de la première analyse MVR-PCR (cf. figure 7.2, partie basse). Les chromosomes ancêtres de cet haplogroupe ne présentent pas cette substitution. Les auteurs supposent que ce phénomène provient d'une substitution initiale $T \rightarrow C$ dans une des répétitions du minisatellite, suivie d'un mécanisme de réparation biaisé qui l'aurait propagée. Ils mettent donc en évidence un nouveau processus d'homogénéisation.

(Andreassen et al., 2002)

Dans cet article, les auteurs étudient directement les types de mutations observées sur MSY1 et établissent une relation entre l'âge du père et le taux de mutation observé dans le minisatellite MSY1 transmis à leurs fils.

Pour cela, ils ont étudié 1071 paires d'allèles provenant d'un père et de son fils. Leur but était de détecter des mutations dans la longueur de l'allèle de MSY1. Parmi ces paires, 600 ont également été analysées par MVR-PCR pour détecter des mutations dans la séquence de MSY1 qui ne produisent pas de changement de longueur. Les auteurs appellent ce type de mutations : « *boundary switches* ».

Ils observent une fréquence de mutations de $2,5$ % et $1,3$ % respectivement pour les polymorphismes de longueur et les *boundary switches*. Soit un taux de mutation total pour le code MVR de $3,8$ %. Ils observent de plus que le taux de polymorphismes de longueur

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
○	Type 1 :	*C*	*A*	**C**	*A*	*A*	*T*	*A*	*T*	*A*	*C*	*A*	*T*	**G**	*A*	*T*	*G*	*T*	*A*	*T*	*A*	*T*	*T*	*A*	*T*	*A*
○	Type 1a :	.	.	.	.	.	*C*	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
●	Type 3 :	*C*	*A*	**C**	*A*	*A*	*T*	*A*	*T*	*A*	*C*	*A*	*T*	**C**	*A*	*T*	*G*	*T*	*A*	*T*	*A*	*T*	*T*	*A*	*T*	*A*
●	Type 3a :	.	.	.	.	.	*C*	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
●	Type 4 :	*C*	*A*	**T**	*A*	*A*	*T*	*A*	*T*	*A*	*C*	*A*	*T*	**C**	*A*	*T*	*G*	*T*	*A*	*T*	*A*	*T*	*T*	*A*	*T*	*A*
●	Type 4a :	.	.	.	.	.	*C*	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Code	Pop.	Hg	Carte MSY1
*m*19	Anglais	2	
*m*66	Kényan	8	
*m*118	Nigérien	8	
*m*71	Kényan	8	
*m*73	Kényan	8	
*m*75	Kényan	8	
			5′ 3′

Type 1 : ○ Type 1a : ○ Type 3 : ● Type 3a : ● Type 4 : ● Type 4a : ● Nul : ⊙

FIG. 7.3 – *En haut, le code des variants de MSY1 spécifiques à l'haplogroupe* 8*, 1a, 3a et 4a. En bas, six cartes de MSY1, dont cinq appartiennent à l'haplogroupe* 8 *(Bouzekri et al., 1998).*

croît avec l'âge du père. Cependant cette relation n'est pas vérifiée pour les *boundary switches*.

En s'intéressant à l'endroit de la mutation dans la carte, ils montrent que le taux des polymorphismes de longueur n'est pas particulièrement élevé au début ($5'$) ou à la fin ($3'$) de la répétition. Cela contredit l'hypothèse de polarité avancée par Jobling et ses collaborateurs dans l'article que nous avons présenté.

Andreassen et ses collaborateurs constatent que plus de 75 % des polymorphismes de longueur sont unitaires, c'est-à-dire qu'ils impliquent le gain ou la perte d'une seule unité répétée. À partir de ce constat et du fait qu'ils observent que l'âge du père influence le taux de mutation, les auteurs suggèrent que le glissement à la réplication[5] est le principal mécanisme des polymorphismes de longueur. Cependant, dans un cas particulier ils constatent des changements structurels complexes qui pourraient indiquer que certains des mutants peuvent se produire à la suite d'échanges entre les chromatides sœurs.

Les auteurs remarquent que le taux de mutation des *boundary switches* est bien plus élevé que ce à quoi on pourrait s'attendre si on fait l'hypothèse que ces mutations proviennent de deux polymorphismes de longueur unitaire indépendants. Ils pensent que le fait que les *boundary switches* ne suivent pas la même distribution d'âge que les polymorphismes de longueur appuie également cette hypothèse. Les auteurs suggèrent que ces *boundary switches* sont produites par un autre mécanisme de mutation que celui des polymorphismes de longueur unitaire.

* *
*

Nous avons vu dans cette section les processus impliqués dans la mutation des minisatellites. Nous nous sommes particulièrement intéressés au minisatellite MSY1 qui est le sujet principal de l'expérimentation de notre algorithme d'alignement de cartes de minisatellite décrit au chapitre 4. Nous avons présenté les caractéristiques de MSY1 ainsi que les hypothèses concernant son mode de mutation.

[5] *Replication slippage.*

Chapitre 8

Application au minisatellite MSY1

Sommaire

8.1 Introduction 169
8.2 Le jeu de données et la distance utilisée 171
8.3 Résultats 172
8.3.1 Prédiction d'haplogroupes avec les cartes MSY1 172
8.3.2 Arbre des relations entre haplogroupes du chromosome Y 173
8.3.3 Relations évolutives à l'intérieur des haplogroupes 174
8.3.4 Évolution interne dans deux grandes populations 180
8.4 Conclusion 183

Dans ce chapitre, nous détaillons les applications à des données biologiques de l'algorithme présenté au chapitre 4.

8.1 Introduction

Certains minisatellites montrent une variabilité dans leur nombre de copies et par conséquent dans la longueur de leur allèle. La variabilité des minisatellites, en terme de nombre d'unités répétées et d'entremêlement des motifs, peut être étudiée de manière informatique grâce aux cartes de minisatellite fournies par la méthode MVR-PCR (Jeffreys et al., 1991). La comparaison des structures internes des allèles est la clé de la compréhension des mécanismes d'expansion des minisatellites. Des réarrangements complexes, impliquant des transferts de groupes de répétitions entre allèles, aussi bien que des duplications intra-alléliques, ont été découverts par des alignements « fait main » entre des codes MVR de minisatellites avant et après mutations (Buard et Vergnaud, 1994), (Jeffreys et al., 1994), (May et al., 1996), (Tamaki et al., 1999) , (Stead et Jeffreys, 2000).

Les alignements entre cartes de minisatellite ont également été utilisés pour déduire les relations évolutives entre les allèles, dans le but d'étudier l'histoire récente des populations humaines (Armour et al., 1993), (Alonso et Armour, 1998), (Hurles et al., 1998),

(Jobling et al., 1998) et (Stead et Jeffreys, 2002). Les deux minisatellites autosomaux qui ont été utilisés jusqu'à présent pour ces études de populations, MS32 et MS205, montrent tous deux des réarrangements intra-alléliques prédominants et :

- soit une variabilité très polarisée, impliquant que les relations entre allèles peuvent se déduire à partir de l'analyse de l'extrémité la plus stable du minisatellite seulement (Armour et al., 1993), (Alonso et Armour, 1998) ;
- soit un faible taux de mutation et par conséquent, une diversité relativement restreinte (Stead et Jeffreys, 2002).

Le pouvoir d'investigation potentiel des minisatellites pour les études sur l'évolution ou pour l'identification reste jusqu'ici limité à cause du manque d'une méthode automatique objective pour comparer les allèles. D'autre part, le signal évolutif devrait mieux être détecté entre les allèles qui n'ont pas subi de réarrangements intensifs lors de recombinaisons. La plupart du chromosome Y échappe à la recombinaison, les mutations sur cette portion du Y représentent un simple enregistrement de son évolution (Jobling et Tyler-Smith, 1995). Le chromosome Y contient toute la gamme des systèmes polymorphes observés dans le génome humain avec des taux de variation allant de 5×10^{-7} pour les substitutions de nucléotides à $3,8\,\%$ pour le minisatellite hypervariable MSY1 (DYF155S1) (Andreassen et al., 2002). Le meilleur moyen d'utiliser les informations contenues dans les différents éléments polymorphes de Y est d'adopter une analyse hiérarchique des marqueurs, du plus stable au plus variable (Hurles et al., 1998), (Hurles et al., 2002). Dans cette approche hiérarchique, plus le système est variable, plus l'histoire inférée est récente.

Pour les raisons évoquées plus haut, nous avons choisi pour nos expériences le minisatellite MSY1, dont nous avons détaillé les caractéristiques au chapitre précédent. Nous voulons savoir, dans un premier temps, si nous pouvons retrouver un signal phylogénétique à partir des cartes de MSY1 en utilisant notre méthode d'alignement. Si c'est le cas, il nous faut vérifier que notre méthode reste stable si nous faisons varier ses paramètres. Nous avons vu que le minisatellite MSY1 évolue rapidement, sommes-nous capables de détecter un signal micro-évolutif permettant, par exemple, l'étude des relations intra-haplogroupes ou des mouvements récents de populations ? Enfin, MSY1 est-il capable de retrouver à la fois les relations inter- et intra-haplogroupes ?

Le reste du chapitre est organisé de la manière suivante : section 8.2, nous décrivons notre jeu de données et expliquons la méthodologie que nous avons adoptée. Ensuite, dans la section 8.3, nous donnons les résultats que nous avons obtenus. Premièrement, section 8.3.1, nous montrons que l'haplogroupe d'un individu peut être prédit de manière relativement sûre à partir de sa seule carte MSY1. À partir des distances entre allèles MSY1, nous décrivons, section 8.3.2, des distances inter-haplogroupes qui nous permettent de reconstruire un arbre évolutif pour ces haplogroupes. Cette phylogénie ne montre pas seulement les mêmes relations anciennes que celle des arbres basés sur des SNPs, mais fournit en plus une structure plus résolue à des niveaux plus bas de l'arbre. L'arbre peut être encore plus raffiné en utilisant les distances MSY1 pour regarder les relations entre allèles à l'intérieur des haplogroupes, section 8.3.3. Dans les haplogroupes ne contenant pas des populations européennes, les arbres regroupent les allèles par populations et les populations

par proximité linguistique ou géographique. Nous avons également construit des arbres phylogénétiques pour quelques populations, section 8.3.4, et constaté que les individus de même haplogroupe se groupent ensemble. Finalement, section 8.4, nous concluons en discutant les résultats obtenus.

8.2 Le jeu de données et la distance utilisée

Mark Jobling nous a fourni 690 cartes de MSY1, parmi lesquelles 609 sont complètement déterminées. Pour chaque individu, sa population d'origine est donnée. Sur les chromosomes Y, Jobling et Tyler-Smith (Jobling et Tyler-Smith, 2000) ont défini 27 haplogroupes en utilisant des SNPs et d'autres marqueurs stables, et ils ont reconstruit l'arbre le plus parcimonieux pour ces haplogroupes. À travers ce chapitre, nous nous référons seulement à ces haplogroupes et utilisons la même numérotation qu'eux. Pour 432 individus parmi les 609, nous connaissons l'haplogroupe du chromosome Y auquel ils appartiennent. Le tableau en annexe A, page 189, montre le nombre de cartes par haplogroupes et par populations.

Les variants de MSY1 diffèrent les uns des autres par au plus trois substitutions et sont organisés suivant une structure modulaire simple. La longueur contrainte des allèles et les structures modulaires observées plaident en faveur d'amplifications et de contractions unaires (Jobling et al., 1998). Le taux de mutation a été récemment estimé et a montré que la plupart des mutations résultant dans le gain ou la perte d'un variant n'affectent pas la structure modulaire (Andreassen et al., 2002). Le modèle mutationnel unaire que nous avons choisi (modèle SSE, chapitre 4) semble bien adapté à MSY1.

Nous avons mis au point un programme informatique pour aligner les cartes MVR [Bérard et Rivals, 2002], [Bérard et Rivals, 2003], la méthode est décrite au chapitre 4 et son implémentation, MS_AlignN, au chapitre 5. Cet algorithme calcule une distance entre paire d'allèles par une méthode de programmation dynamique sous un modèle évolutif approprié, le modèle SSE. Il trouve de manière sûre un alignement qui maximise la similarité des deux cartes et la distance peut être interprétée comme le coût minimal des événements nécessaires pour transformer une carte en l'autre.

Nous avons utilisé MS_AlignN pour comparer 609 allèles complets de MSY1 provenant d'hommes de 76 populations et 18 haplogroupes différents. Les distances obtenues entre les paires d'allèles ont été utilisées pour construire des arbres phylogénétiques.

Nous notons le coût de l'amplification par A, de la contraction par C, de la mutation par M, de la délétion par D et de l'insertion par I. Étant donné que pour MSY1, les amplifications et contractions sont plus probables que les autres événements mutationnels, nous leur attribuons un coût plus faible que ceux des autres événements, donc $A, C < M, D, I$. Les coûts que nous utilisons sont en général $A = C = 1, M = 10, D = I = 20$. A et C ont le même coût ainsi que D et I à cause de la symétrie du score. Il est important de noter que la valeur exacte de D et I n'a pas d'importance si elle est supérieure à $A + M$, ce qui est vrai pour notre cas. Seul le ratio M/A est important et nous avons essayé plusieurs valeurs de ce rapport de manière à le faire varier. Tous les arbres montrés dans ce chapitre, sauf celui de Jobling et Tyler-Smith, figure 8.2 (a), ont été construits à partir de distances

calculées avec les paramètres $A = C = 1, M = 10$, et $D = I = 20$.

Les arbres phylogénétiques ont été construits avec la méthode de [Gascuel, 1997] appelée BioNJ. Elle prend en entrée la matrice de distances entre les cartes et donne en sortie un arbre phylogénétique non enraciné, dont les longueurs des branches représentent les distances. Les arbres des figures 8.1 et 8.2 (b) sont visualisés avec le programme Phylip [`Phylip`], les autres arbres, figures 8.3, 8.4, 8.5, 8.6, 8.7 et 8.8, sont visualisés avec le programme NJ-Plot [Perrière et Gouy, 1996].

8.3 Résultats

Nous présentons dans cette section les principaux résultats de nos expériences.

8.3.1 Prédiction d'haplogroupes avec les cartes MSY1

Dans cette section, nous cherchons à retrouver l'haplogroupe d'un allèle en regardant seulement sa carte MVR.

Soit i un individu, s sa carte MSY1 et k un nombre entier. La méthode de classification dite des « k plus proches voisins » recherche les k plus proches voisins de s, notés $V(s)$, selon les distances deux à deux [Gordon, 1999]. Elle prédit l'haplogroupe de l'individu en considérant l'haplogroupe le plus représenté dans $V(s)$. La représentation de l'haplogroupe H noté $R(H)$ est donnée par :

$$R(H) = \sum_{t \in (H \cap V(s))} \frac{1}{1 + d(s,t)}$$

Dans $R(H)$, le nombre de voisins est pondéré par l'inverse de leur distance avec s, c'est-à-dire leur proximité. L'individu i est prédit dans l'haplogroupe H tel que $R(H)$ est maximum. En cas d'égalité, i est prédit dans l'haplogroupe, parmi ceux dont $R(H)$ est maximum, qui a le plus de représentants dans $V(s)$.

En utilisant la matrice de distances deux à deux entre les individus et la méthode des k plus proches voisins, nous pouvons prédire correctement l'haplogroupe dans 80 % des cas quand k varie entre 3 et 5 voisins. Lorsque k est dans les intervalles $[3, 11]$ et $[1, 25]$ voisins, les prédictions sont correctes dans 78 % et 75 % des cas respectivement. Si l'on ne regarde pas seulement l'haplogroupe le plus proche, mais les deux ou trois plus proches, alors le pourcentage de prédictions correctes monte respectivement jusqu'à 90 % et 93 % pour $k = 9$. L'exactitude des prédictions reste au même niveau quand on fait varier les paramètres de la méthode d'alignement entre $(M/A = 4)$ et $(M/A = 10)$. Cela montre la robustesse de la prédiction. Il existe des cas d'homoplasie entre cartes de différents haplogroupes, c'est-à-dire le fait que les mêmes allèles peuvent apparaître dans des haplogroupes différents. Le pourcentage de prédiction correcte d'haplogroupes pour la valeur optimale de k indique que l'homoplasie est une exception plutôt qu'une règle.

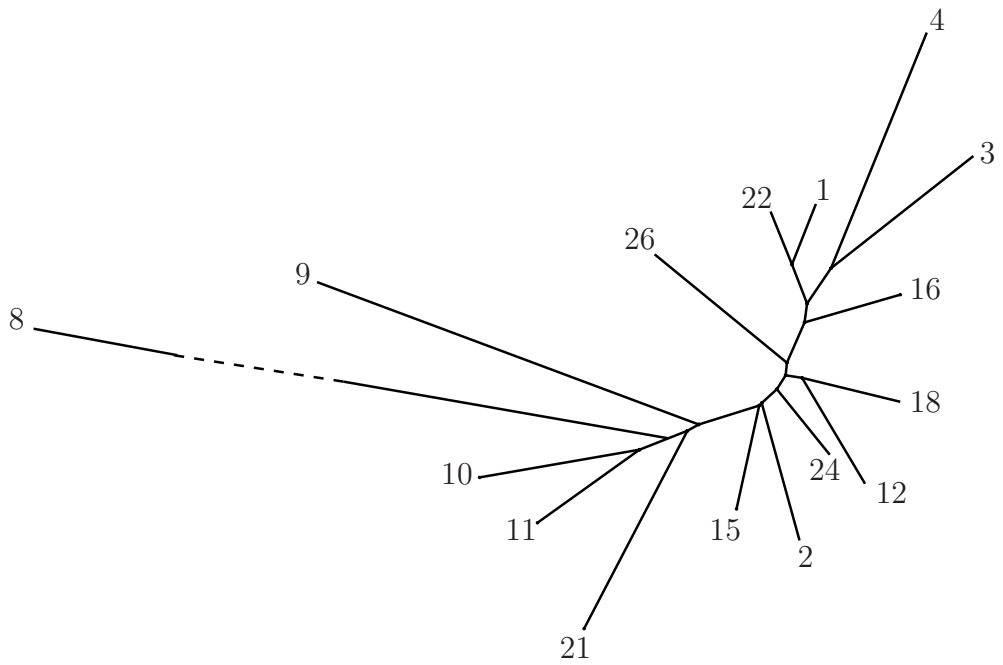


FIG. 8.1 – *Arbre phylogénétique des haplogroupes du chromosome Y, calculé à partir des cartes de MSY1 avec les distances moyennes inter-haplogroupes. La branche reliant l'haplogroupe* 8 *à l'arbre n'est pas représentée dans sa totalité (elle mesure en réalité environ* 4 *fois la taille de la branche reliant l'haplogroupe* 9*).*

8.3.2 Arbre des relations entre haplogroupes du chromosome Y

À partir de la matrice de distances entre individus, nous calculons une matrice des distances moyennes entre haplogroupes. On note $d(i, j)$ la distance entre les individus i et j. Soient A et B deux haplogroupes, la distance moyenne entre A et B est donnée par :

$$D(A, B) = \frac{\sum_{i \in A, j \in B} d(i, j)}{|A| \times |B|}$$

Nous utilisons le programme BIONJ [Gascuel, 1997] pour reconstruire un arbre évolutif non enraciné des haplogroupes avec ces distances moyennes en entrée. Cet arbre est montré à la figure 8.1. Par souci d'expressivité, nous avons seulement inclus les haplogroupes de notre jeu de données ayant au moins cinq cartes disponibles, c'est-à-dire les haplogroupes 1-4, 8-12, 15, 16, 18, 21, 22, 24 et 26.

L'arbre le plus parcimonieux des haplogroupes du chromosomes Y basé sur des polymorphismes de substitution est proposé par (Jobling et Tyler-Smith, 2000). Une représentation de cet arbre, incluant seulement les haplogroupes dont nous disposons dans notre jeu de données se trouve à la figure 8.2 (a). Ils ont enraciné l'arbre à l'aide d'un « *outgroup* ». L'arbre contient trois nœuds en étoile ($2, 26$ et 1) où tous les nœuds fils

se branchent directement sur leur père. Une des raisons qui expliquent cela est que les SNPs sont des données binaires et ne permettent pas de résoudre de tels nœuds (voir (Jobling et Tyler-Smith, 1995)).

Nous comparons l'arbre MSY1 avec l'arbre des haplogroupes de Jobling et Tyler. La figure 8.2 permet de visualiser les ressemblances. Le long de la branche centrale de notre arbre (b), on reconnaît une structure globale commune : trois segments correspondant à trois ensembles d'haplogroupes, (2, 15, 9, 21, 8, 10, 11), (26, 24, 12, 16, 18) et (1, 22, 3, 4). La branche centrale dans l'arbre des SNPs (a) relie 2, 26 et 1, chacun d'entre eux étant le centre d'un nœud étoile. À l'exception des haplogroupes 4 et 18 qui n'ont pas des positions similaires dans les deux arbres, chacun des trois sous-ensembles de notre arbre correspond à un haplogroupe du centre de l'étoile plus ses descendants. L'arbre MSY1 (b) amène de nouvelles informations car il résout l'évolution des descendants dans les haplogroupes du centre : 1 et 2. Dans le cas de l'haplogroupe 2, l'ordre de branchement indique que l'haplogroupe 15 s'est séparé le premier, puis les haplogroupes 9, 21 et 8 et finalement les haplogroupes 10 et 11.

L'arbre des haplogroupes est identique quand on utilise des paramètres d'alignement allant de $(M/A = 9)$ à $(M/A = 11)$. Il reste relativement stable entre $(M/A = 2)$ et $(M/A = 15)$ puisque seuls les haplogroupes 4, et puis 3 et 21 bougent, pour les rapports de plus en plus petits. Il est à noter qu'avec un rapport M/A plus élevé, par exemple $(M/A = 13)$ l'haplogroupe 4 vient se brancher sur le 26 et laisse l'ensemble $1, 22, 3$ monophylétique, comme c'est le cas dans l'arbre de Jobling et Tyler-Smith. En conclusion, l'arbre des haplogroupes MSY1 est correct pour les niveaux anciens de l'évolution et fournit une résolution plus fine que l'arbre basé sur les SNPs aux niveaux récents.

8.3.3 Relations évolutives à l'intérieur des haplogroupes

Nous avons calculé les arbres évolutifs pour tous les haplogroupes contenant au moins deux populations différentes et avec au moins deux cartes de chaque population. Les arbres ont donc été calculés pour les haplogroupes $1, 2, 3, 4, 8, 12, 16, 18, 21, 22, 24$ et 26. Dans ces arbres, nous avons pu observer à quel point les populations étaient séparées, et ainsi estimer la spécificité géographique du chromosome Y. Les figures 8.3, 8.4, 8.5 et 8.6 montrent les arbres des haplogroupes $4, 12, 16$ et 18 respectivement. Dans chacun de ces arbres un individu est représenté par son code, sa population d'origine entre crochets, son haplogroupe entre accolades et sa carte MSY1 entre parenthèses.

L'arbre de l'haplogroupe 4 (fig. 8.3, page 176) sépare parfaitement 10 japonais de 4 tibétains et 1 mongol. Cela est en concordance avec la distance géographique séparant ces populations.

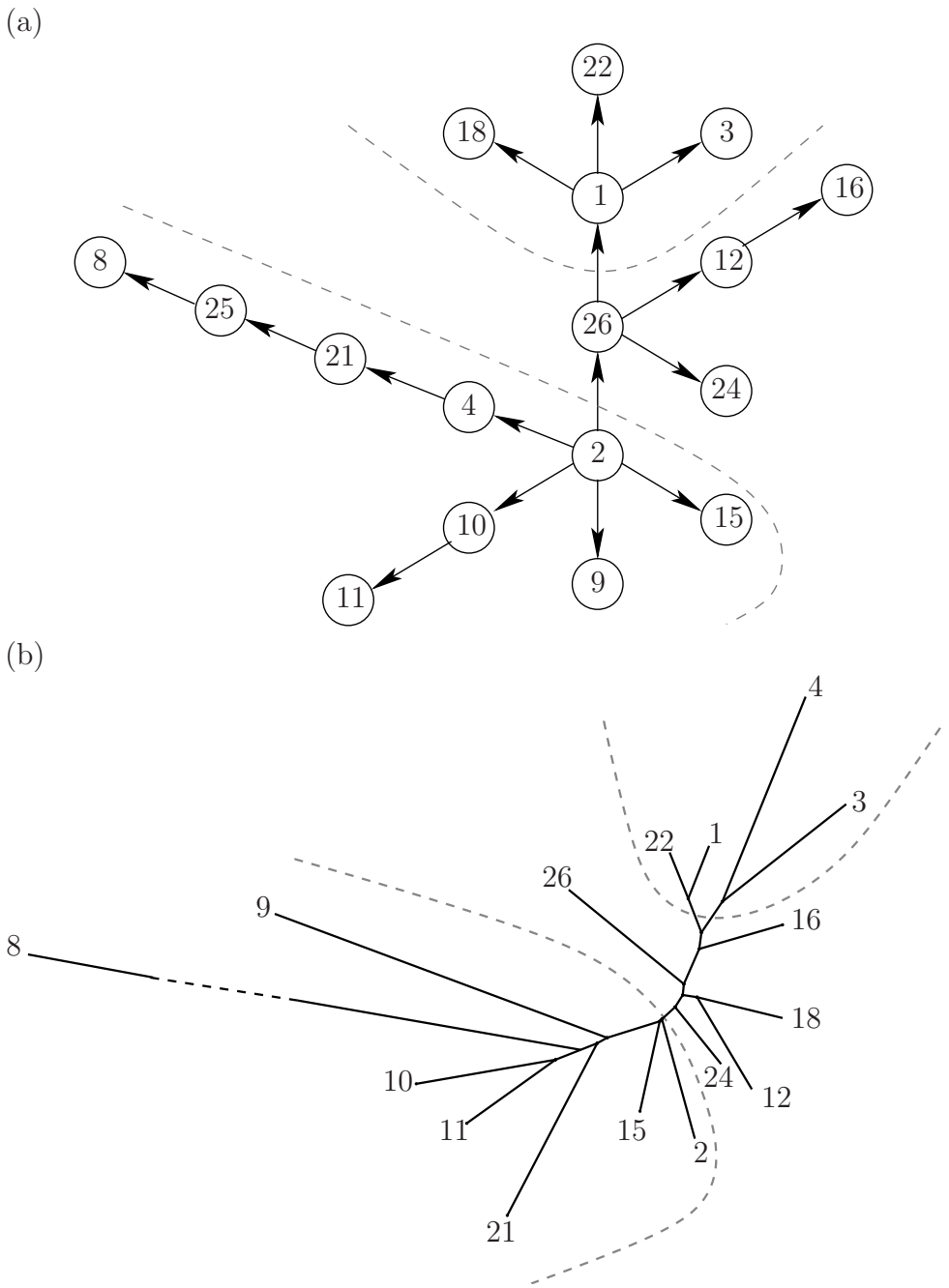


FIG. 8.2 – *(a) Arbre des haplogroupes du chromosome Y de Jobling et Tyler-Smith calculé à partir de marqueurs SNPs. Les flèches représentent les polymorphismes, elles sont orientées de l'état ancêtre vers l'état dérivé. Notons que l'haplogroupe* 25 *n'appartient pas à notre jeu de données. Il apparaît ici car il est l'intermédiaire entre les haplogroupes* 21 *et* 8*.*
(b) Arbre phylogénétique des haplogroupes fait à partir des cartes MSY1 (report de la figure 8.1). Les lignes en pointillés délimitent les trois ensembles d'haplogroupes.

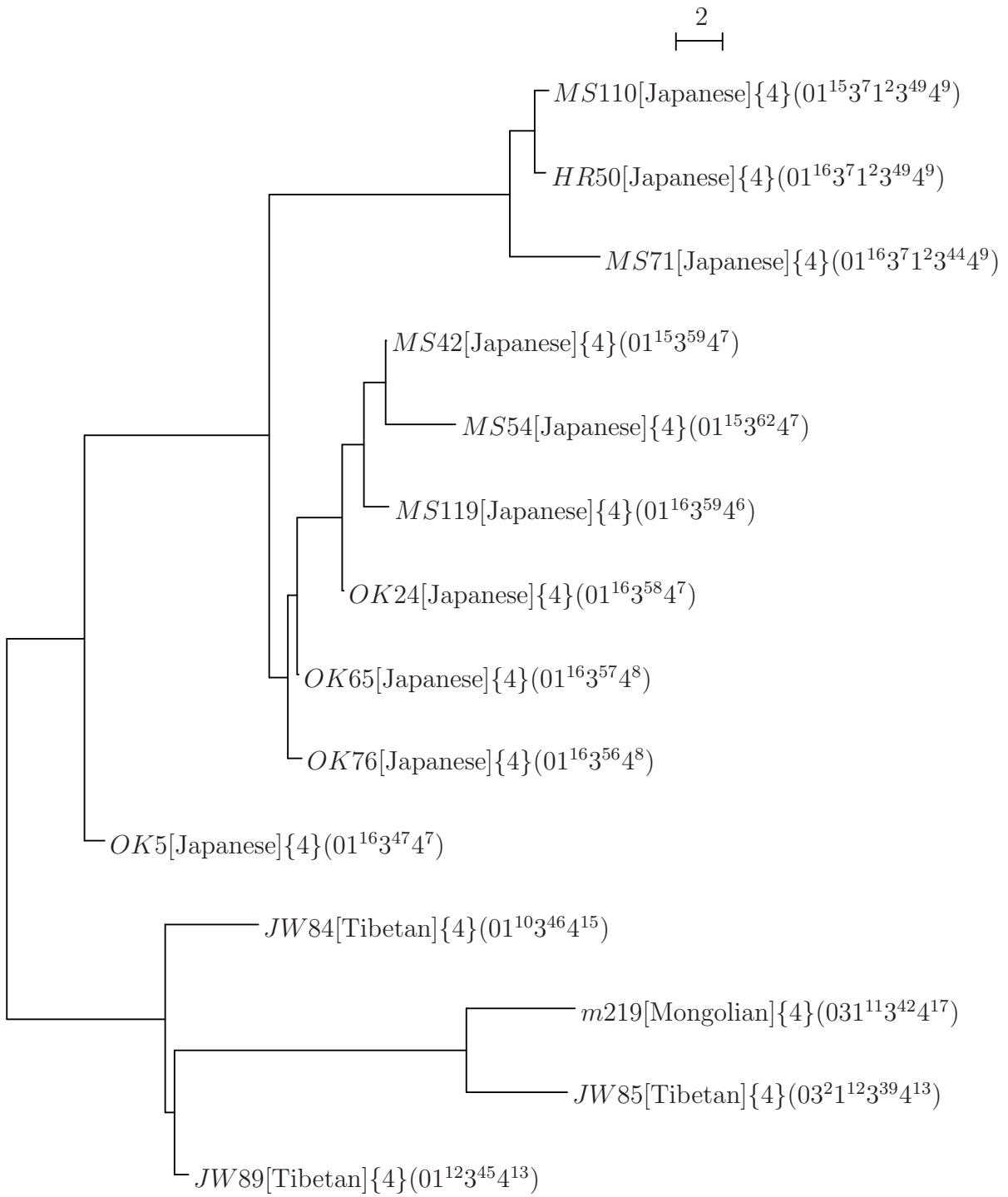


Fig. 8.3 – *Arbre phylogénétique de l'haplogroupe* 4.

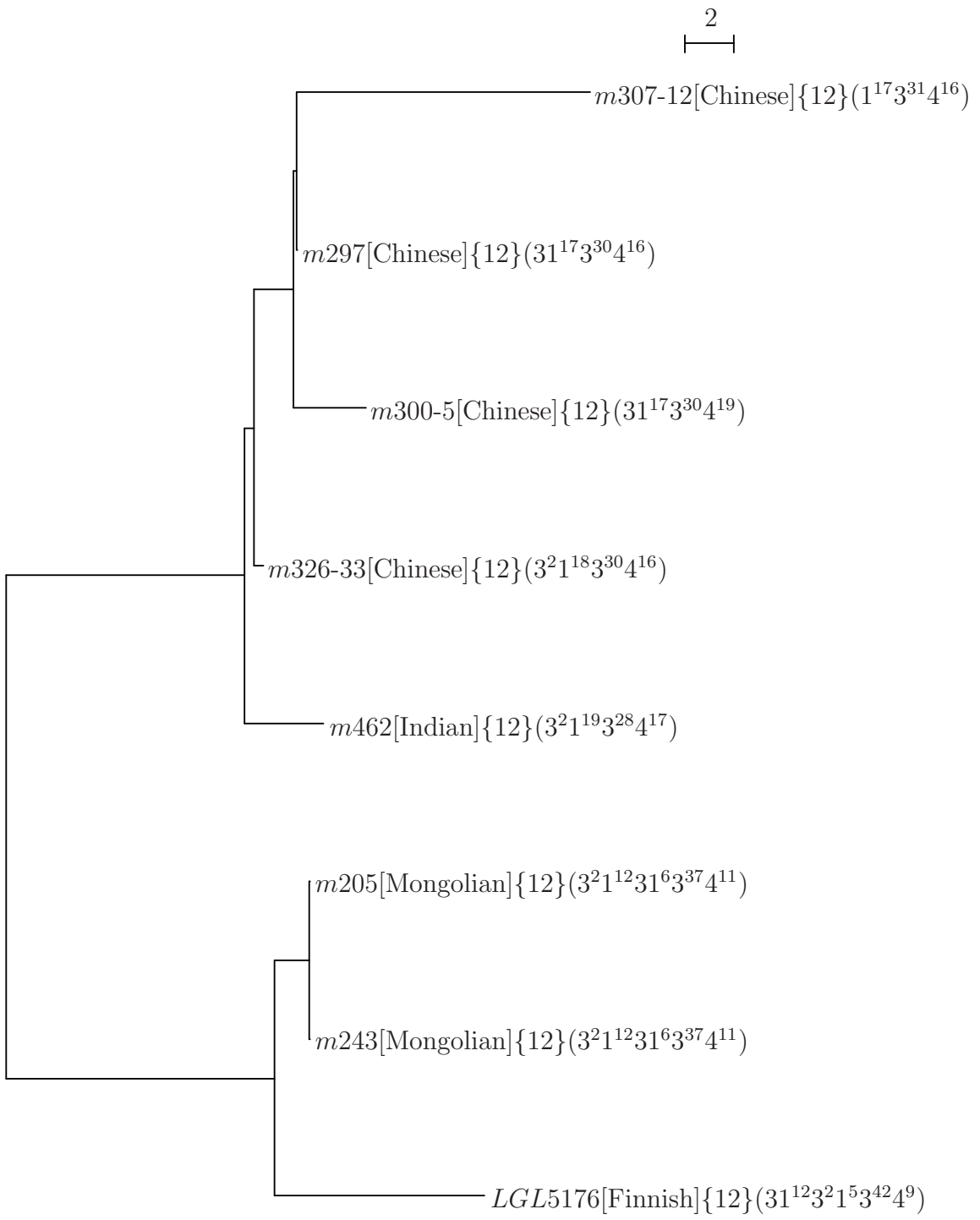


FIG. 8.4 – *Arbre phylogénétique de l'haplogroupe* 12.

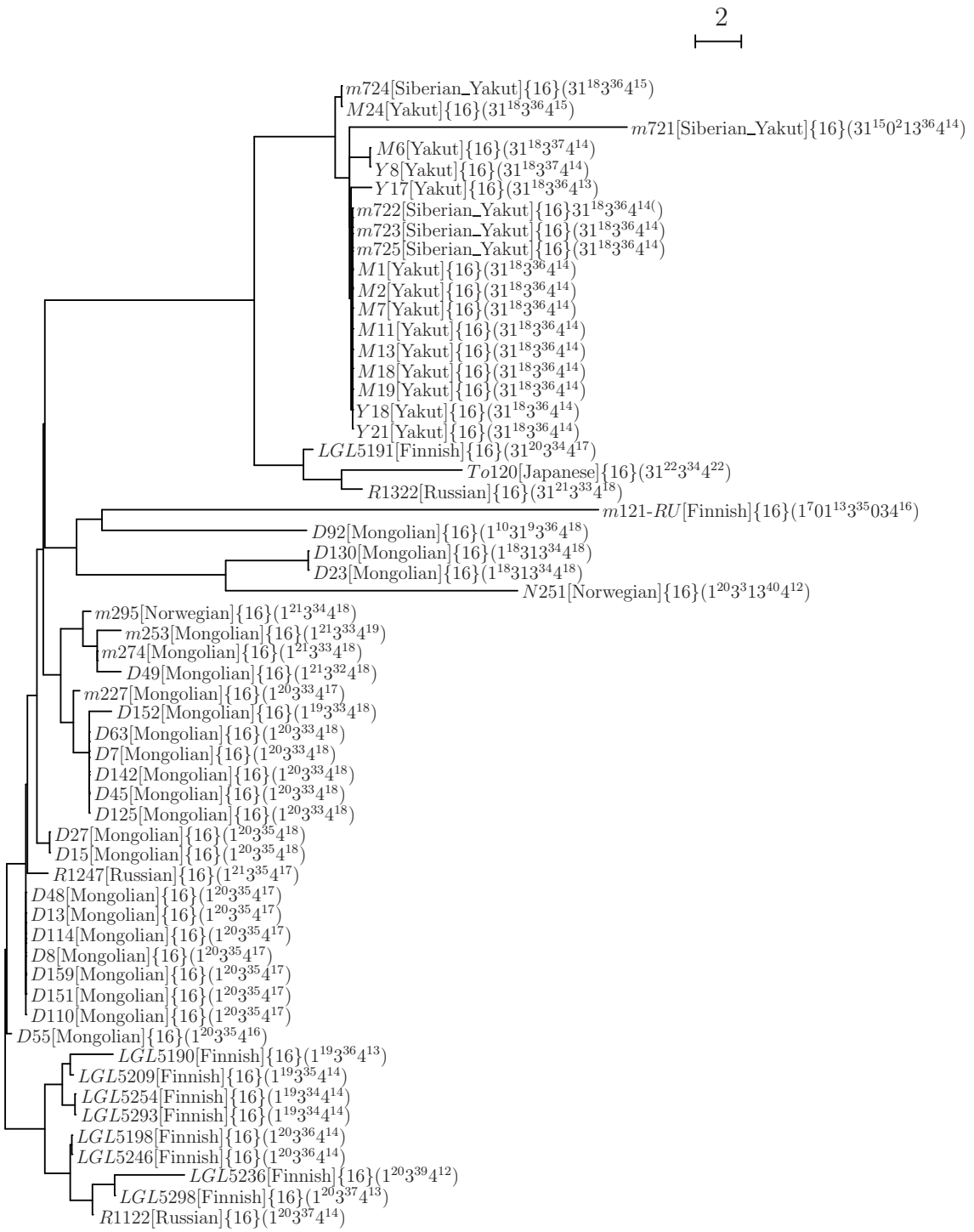


Fig. 8.5 – *Arbre phylogénétique de l'haplogroupe* 16.

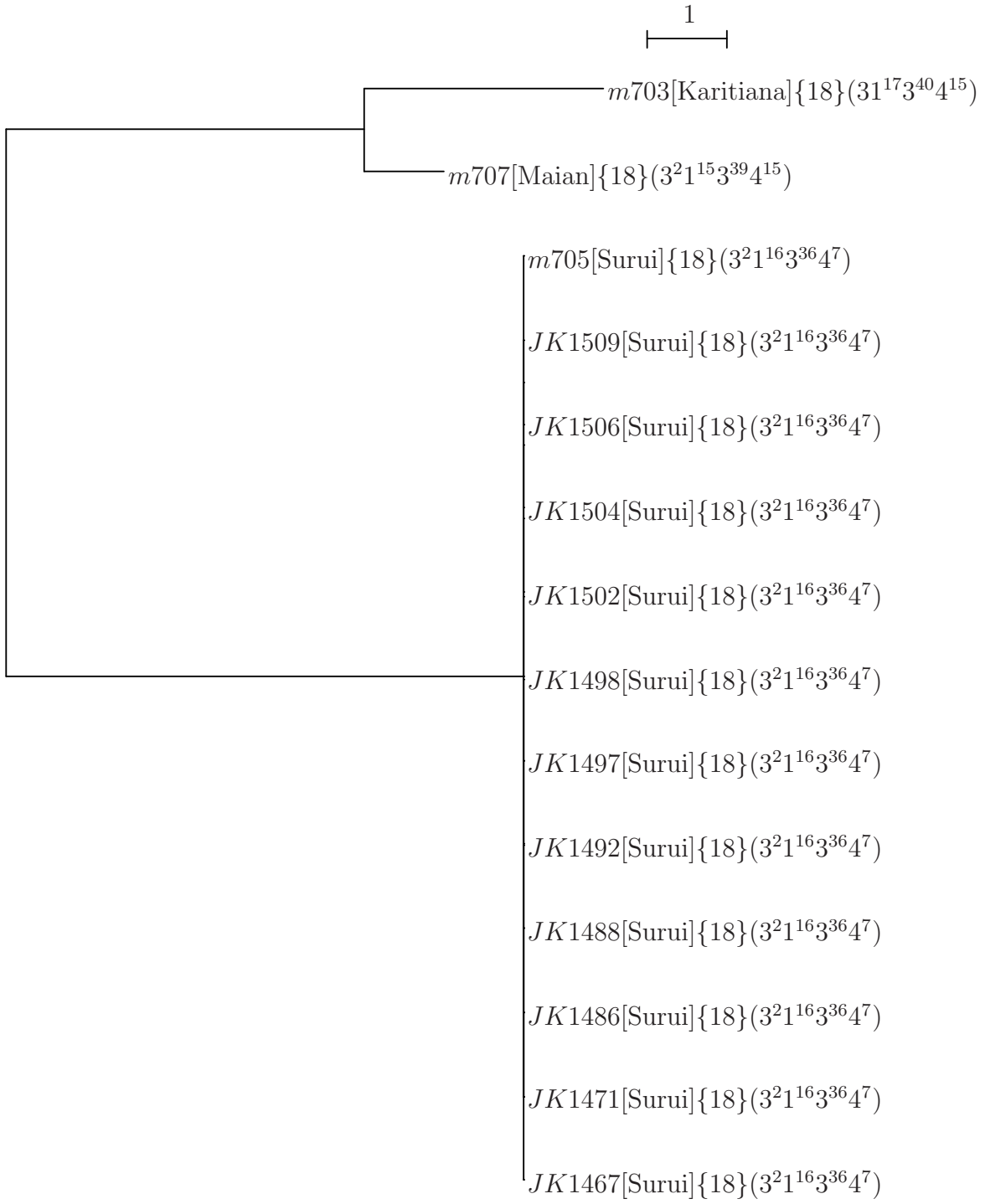


FIG. 8.6 – *Arbre phylogénétique de l'haplogroupe* 18.

L'arbre de l'haplogroupe 12 (fig. 8.4, page 177) sépare 4 chinois et 1 indien de 2 mongols et 1 finnois. Bien que la Mongolie soit plus près de la Chine que de la Finlande, ceci est en accord avec l'hypothèse de l'origine nordique des mongols et des finnois (Cavalli-Sforza, 2000), qui parlent tous deux des langages uraliques[1], (Ruhlen, 1987).

Parmi les 57 cartes disponibles de l'haplogroupe 16, la majorité appartiennent à trois groupes principaux : mongols (23), finnois (10) et yakuts et yakuts sibériens ($13+5$). Dans l'arbre représenté à la figure 8.5, page 178, les yakuts sont monophylétiques. Les mongols forment également un groupe (avec 1 finnois, $m121$-RU, 2 norvégiens, $N251$ et $m295$, et 1 russe, $R1247$), tandis que le reste des finnois s'agglomèrent ensemble. À l'exception d'un finnois, $LGL5191$, d'un japonais, $To121$, et d'un russe, $R1322$, qui se placent à côté des yakuts, la séparation géographique apparaît encore clairement dans cet arbre de MSY1.

Dans l'arbre de l'haplogroupe 18 (fig. 8.6, page 179), 12 suruis sont monophylétiques et opposés à 1 maya et 1 karitania. Les suruis et les karitanias sont toutes deux des ethnies d'Amazonie. Cela est en accord avec le fait que les suruis et les karitanias vivent retirés les uns des autres dans le sud du Brésil (Bahuchet, 1994).

Pour les haplogroupes qui incluent en majorité des populations européennes, nous ne percevons pas une telle séparation de populations. C'est aussi le cas dans l'haplogroupe 8 qui inclut des allèles africains. D'autres haplogroupes montrent une presque monophylie de la population principale avec à l'intérieur quelques individus d'autres populations.

8.3.4 Évolution interne dans deux grandes populations

Nous construisons deux arbres pour les populations finlandaises et mongoles (fig. 8.7 et 8.8). Dans ces arbres les individus sont représentés de la même manière que dans les arbres d'haplogroupe de la section précédente. Nous disposons pour chacune de ces populations de 19 et 48 cartes respectivement.

Une caractéristique remarquable est que les sous-groupes de cartes de même haplogroupe se groupent ensemble. Par exemple, les Finnois de l'haplogroupe 3, 16 et 2 respectivement sont tous monophylétiques (à l'exception de la carte $LGL5191$[Finnish]{16} qui était déjà mal positionnée dans l'arbre de l'haplogroupe 16, page 178). Un groupement similaire apparaît à l'intérieur de la population Mongole, l'haplogroupe 11 est monophylétique et vient se placer à l'intérieur du groupe de carte de l'haplogroupe 10. Les cartes de l'haplogroupe 12 viennent se placer à l'intérieur du groupe de carte de l'haplogroupe 26.

De plus, de bas en haut dans l'arbre des Finnois, (fig. 8.7, page 181), les haplogroupes sont $2, 12$ et 16, puis 1 et 3. Clairement, les structures des deux arbres reflètent la structure de l'arbre des haplogroupes de MSY1. L'évolution de ces populations ne corrobore pas seulement la partition en haplogroupes, mais également les relations entre ces haplogroupes.

[1]Tout comme l'indo-européen ou l'afro-asiatique, l'uralique est une grande catégorie de langages. Elle contient, entre autres, le finnois et le lapon.

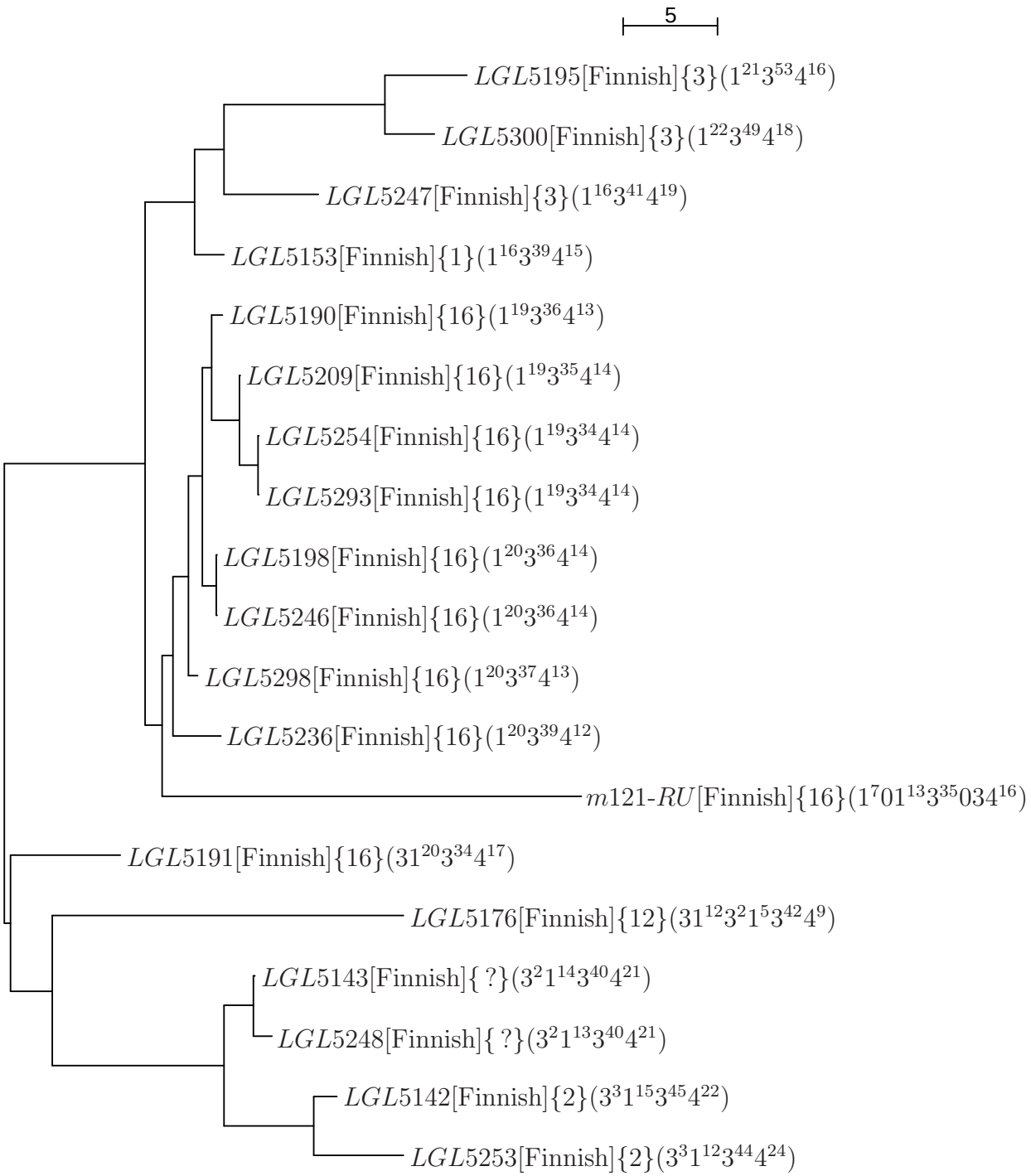


FIG. 8.7 – *Arbre phylogénétique des allèles finnois.*

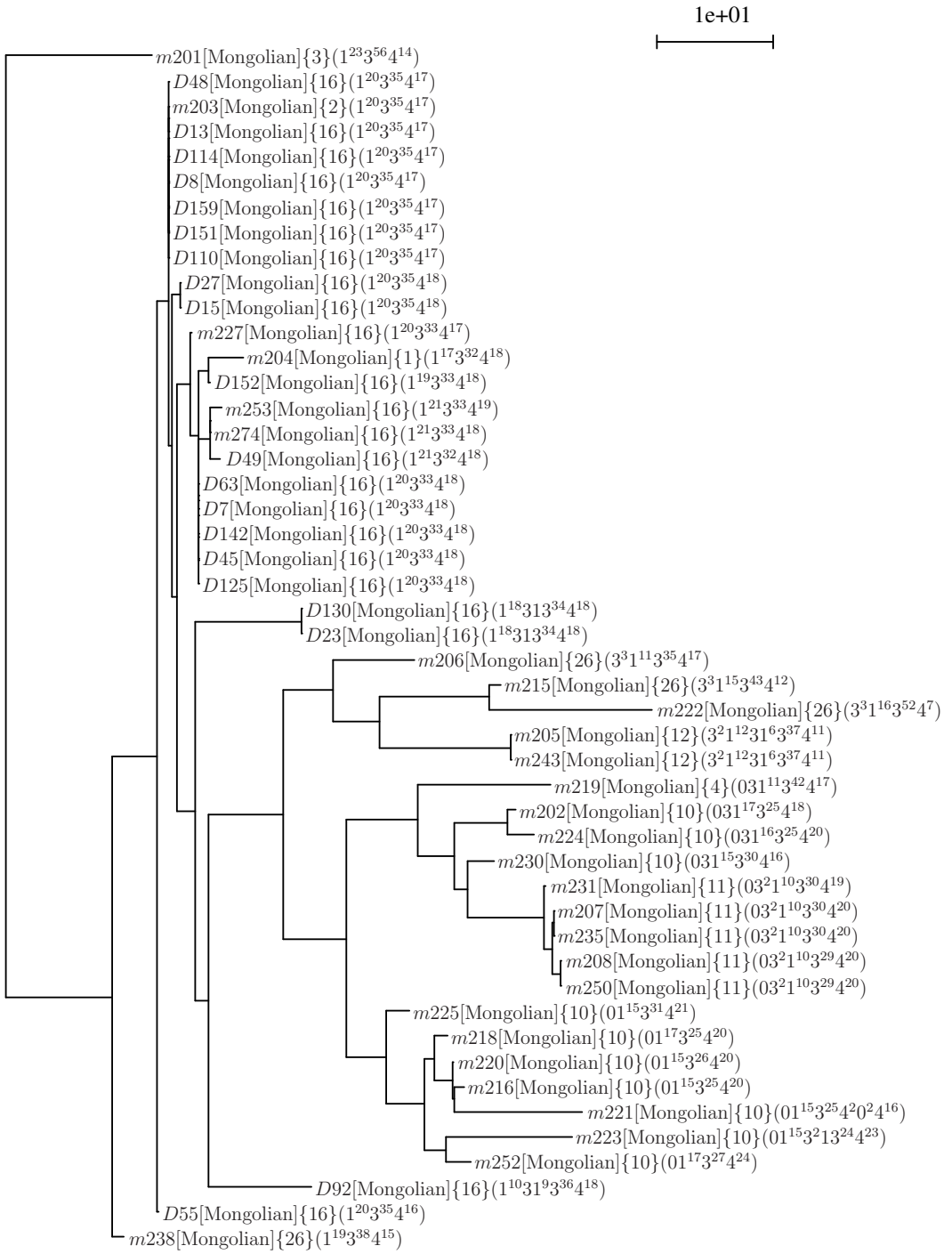


Fig. 8.8 – *Arbre phylogénétique des allèles mongols.*

8.4 Conclusion

Notre premier but est de tester si un signal phylogénétique peut être détecté à partir des cartes de MSY1. En utilisant des marqueurs stables, principalement des SNPs, Jobling et Tyler-Smith (Jobling et Tyler-Smith, 2000) définissent des haplogroupes du chromosome Y et reconstruisent un arbre parcimonieux de leurs relations. Notre approche consiste à calculer la distance entre chaque paire de cartes avec une méthode spécifique d'alignement MS_AlignN (cf. chapitre 5) et à utiliser une méthode basée sur les distances, BioNJ, pour retrouver le signal phylogénétique. Premièrement, avec une méthode simple de classification, nous sommes capables de prédire correctement l'haplogroupe de 80 % des allèles. Deuxièmement, nous reconstruisons un arbre évolutif pour les haplogroupes en utilisant une distance moyenne inter-haplogroupe. Notre arbre dépeint des relations similaires à l'arbre des SNPs. Troisièmement, nous construisons des arbres phylogénétiques pour les allèles d'une population. Dans de tels arbres, les hommes des mêmes haplogroupes se groupent ensemble et les relations entre ces groupes reproduisent les relations entre haplogroupes précédemment trouvées. Ces résultats s'accordent à montrer qu'avec l'alignement de cartes de MSY1 il est possible de retrouver les relations phylogénétiques qui ont amenées à la définition des haplogroupes du chromosome Y et une partie de relations plus anciennes entre haplogroupes. De plus, notre approche retrouve plus d'information pour les bas niveaux de l'arbre puisqu'elle résout les relations entre haplogroupes des nœuds en étoile de l'arbre des SNPs.

Le génotypage de marqueurs bi-alléliques assigne parfois à la même séquence des haplogroupes différents. Cela correspond soit à des erreurs expérimentales, soit à des cas d'homoplasie sur MSY1. Nos résultats donnent des preuves que le degré d'homoplasie est assez limité et permet de retrouver, à travers des alignements de minisatellites, des relations phylogénétiques connues. Récemment, dans une étude complète de la région non recombinante de Y, le « *Y Chromosome Consortium* » (YCC) a défini 153 haplogroupes en utilisant 247 sites polymorphes. Ils ont reconstruit un arbre parcimonieux pour ces haplogroupes et ont donné les correspondances avec les haplogroupes définis antérieurement (The Y Chromosome Consortium, 2002). Dans l'arbre du YCC, les haplogroupes $1, 2$ et 26 ne sont plus monophylétiques et appartiennent à différents clades majeurs définis par le YCC[2]. De plus, l'arbre du YCC n'est pas en accord avec la relation « l'haplogroupe 4 est père de l'haplogroupe 21, lui même ancêtre de l'haplogroupe 8 » comme c'était le cas dans l'arbre des SNPs de Jobling et Tyler-Smith. Ces contradictions nous empêchent de déduire les haplogroupes du YCC à partir de l'appartenance aux haplogroupes SNP de notre jeu de données et, par conséquent, de construire l'arbre des haplogroupes du YCC. Ces variations dans la définition de l'haplogroupe est une explication possible pour les 80 % de précision dans les prédictions d'haplogroupes, et également pour les différences entre l'arbre des SNPs et l'arbre de MSY1.

Dans (Bríon et al., 2002), une méthode statistique pour comparer des codes MVR a été définie. Cependant, la reconstruction de leur arbre n'a pas été validée par ailleurs avec des

[2]Le YCC définit en tout 19 clades majeurs.

relations connues entre haplogroupes. De plus, les résultats dépendent fortement des valeurs des paramètres. En opposition, nous avons montré que l'analyse phylogénétique avec les alignements de cartes de minisatellite est stable quand les paramètres de l'alignement varient et peut retrouver les relations connues entre haplogroupes (Jobling et Tyler-Smith, 2000). La méthode statistique mesure une similarité globale entre deux cartes MVR. L'algorithme d'alignement révèle une similarité plus détaillée puisqu'il calcule la série de mutations la plus parcimonieuse transformant une carte en une autre.

En plus des relations évolutives profondes, MSY1 peut être utilisé pour tracer l'histoire récente de la population humaine. Un arbre phylogénétique a été construit en utilisant les cartes MSY1 pour chaque haplogroupe. Pour beaucoup d'entre eux, $(4, 9, 12, 16, 18, 21, 24)$, des sous-groupes dans l'arbre MSY1 correspondent à des populations qui sont géographiquement distantes. Cela met en avant la spécificité géographique des haplogroupes Y déjà observée avec d'autres marqueurs (Seielstad et al., 1994), (Underhill et al., 1996) et (Underhill et al., 1997). Une exception est l'arbre construit pour les allèles MSY1 de l'haplogroupe 12 qui groupe les Finnois et les Mongols qui sont « linguistiquement » plus proches. Ces observations démontrent que les alignements entre cartes de MSY1 peuvent révéler des signaux micro-évolutifs résultant des migrations, séparations et mélanges récents des populations.

MSY1 est le site le plus variable du chromosome Y et des cas d'homoplasie ont été observés (Jobling et al., 1998). Comment est-il possible qu'à la fois les relations inter- et intra-haplogroupe soient détectées à partir des cartes de MSY1 ? Le premier type de relation est usuellement détecté en utilisant des marqueurs évoluant lentement, tels que les SNPs, tandis que le second requiert des marqueurs évoluant plus rapidement, typiquement des microsatellites. En fait, les cartes MSY1 adoptent une structure modulaire (Jobling et al., 1998) où les variants sont organisés en blocs. La structure la plus représentée est composée de trois blocs de variants de type 1, 3 et 4. Cette structure est stable et est présente dans 10 des 18 haplogroupes dont nous disposons. De plus, les cartes qui ne sont pas exactement $(1, 3, 4)$ ont un bloc supplémentaire de quelques variants en $5'$, et sont ainsi proches de cette structure. D'un autre côté, la longueur des allèles, et par là même, la longueur des blocs, varie grandement. La combinaison des longueurs de blocs caractérise certains haplogroupes ; cela explique pourquoi nous pouvons prédire l'appartenance à un haplogroupe à partir de la carte MVR. Nous suggérons que, parce que MSY1 subit deux types de variations, une variation lente au niveau de sa structure et une plus rapide au niveau de la longueur des blocs, les relations évolutives au niveau inter-haplogroupes aussi bien qu'au niveau intra-haplogroupes sont possibles à extraire des cartes MVR.

Avec une méthode simple de classification, nous sommes capables de prédire l'haplogroupe de 80 % des allèles. Cette propriété de prédiction peut être utile pour identifier des individus. En effet, la recherche du meilleur alignement d'une carte de MSY1 parmi des centaines de cartes peut indiquer l'haplogroupe ou la population d'origine d'un allèle non caractérisé. De plus, cela peut servir à vérifier les tests de marqueurs binaires qui déterminent les haplogroupes. Avec une méthode plus sensible et un ensemble non biaisé de codes MSY1, la précision des prédictions serait certainement améliorée.

Une perspective pour l'approche de l'alignement est de considérer des modèles muta-

tionnels plus complexes, comme des amplifications de plusieurs variants adjacents en une fois. Ces événements sont moins fréquents que les amplifications unaires mais ils sont probablement importants dans la génération de nouvelles structures (Andreassen et al., 2002) ; les considérer améliorerait la détection des relations profondes entre allèles.

Plusieurs questions concernant le comportement de la méthode se posent lorsqu'elle est appliquée à d'autres minisatellites. Est-elle est également adaptée aux classes de minisatellites riches en *GC* dont les mécanismes évolutifs diffèrent de ceux de MSY1 riche en *AT* ? Comment la conversion génique, en œuvre sur des loci diploïdes, perturbe-t-elle le signal phylogénétique ? Pour les minisatellites autosomaux, une extension adéquate du modèle est d'autoriser à la fois les réarrangements de minisatellites intra- et inter-alléliques. Cela nous permettrait de mesurer l'importance de la conversion allélique dans l'évolution des minisatellites. Un candidat approprié pour une telle étude est le minisatellite de l'insuline, INS, qui évolue à la fois à travers des événements de type glissement et de type conversion génique (Stead et Jeffreys, 2000). Il a été montré que les classes de taille d'allèles de ce minisatellite sont associées à différentes prédispositions aux diabètes de type I (Bennett et al., 1997), (Vafiadis et al., 1997), (Dunger et al., 1998), (Ong et al., 1999) et (Stead et al., 2000). Un grand nombre d'allèles ont été typés jusque là par MVR-PCR (Stead et al., 2000), (Stead et Jeffreys, 2000) et (Stead et Jeffreys, 2002), reconsidérer les analyses en utilisant l'alignement des cartes MVR pourrait fournir un aperçu plus détaillé des relations entre l'évolution du minisatellite INS et les diabètes de type I.

Conclusion

Nous nous sommes intéressés dans cette thèse à l'alignement de séquences répétées en tandem dans le but d'appliquer cette méthode de comparaison à des séquences d'ADN particulières : les minisatellites.

Dans un premier temps nous avons modélisé l'évolution de telles séquences. Cela a donné lieu à un premier modèle nommé SSE. Sous ce modèle, nous avons mis au point un algorithme d'alignement exact permettant de prendre en compte les deux événements évolutifs spécifiques à ces séquences : l'amplification et la contraction. Cet algorithme a été intégré dans un logiciel nommé MS_ALIGN. Grâce à MS_ALIGN, nous avons pu étudier un jeu de données biologiques composé de cartes du minisatellite humain MSY1 situé sur le chromosome Y. Les résultats de cette étude montre d'une part que le modèle SSE, qui suppose l'absence de recombinaison et des variations de longueur unitaires, est bien adapté. Ceci corrobore les études directes des événements de mutations au locus MSY1 (Jobling et al., 1998), (Andreassen et al., 2002). Les résultats indiquent aussi que l'étude des minisatellites de manière automatisée permet de détecter des signaux micro-évolutifs du chromosome Y, signaux qui sont cohérents avec les caractéristiques connues de l'évolution de ce chromosome.

Nous avons ensuite essayé d'étendre le modèle SSE pour mieux prendre en compte le mode d'évolution des séquences répétées en tandem. Pour cela nous avons relâché les contraintes pesant sur les opérations d'amplification et de contraction. Nous n'avons pas trouvé de méthode exacte pour le modèle étendu, mais proposons plusieurs pistes pour des algorithmes heuristiques.

* *
*

Les perspectives de ce travail sont de plusieurs ordres. Concernant l'étude des minisatellites, nous voudrions étudier avec notre logiciel MS_ALIGN, des minisatellites autosomaux, c'est-à-dire subissant la recombinaison. Ces études permettraient d'approfondir les connaissances sur la part des événements inter-alléliques dans l'évolution des minisatellites. Une étude du minisatellite autosomal MS205 a été effectuée par É. Fontanillas (Fontanillas, 2002) et donne des résultats encourageants.

En outre, il existe aujourd'hui un nouvel arbre des haplogroupes du chromosome Y (The Y Chromosome Consortium, 2002) et il serait intéressant de pouvoir comparer nos

résultats liés aux haplogroupes définis pas Jobling et Tyler-Smith avec l'arbre des 153 haplogroupes du YCC. Pour cela il nous faut reclasser les cartes de MSY1 dont nous disposons suivant la nouvelle nomenclature des haplogroupes.

Une autre perspective de ce travail serait d'ajouter au modèle d'autres événements évolutifs tels que la recombinaison, l'homogénéisation ou les réarrangements. Par exemple, pour le cas de MSY1 discuté aux chapitres 7 et 8, nous pourrions prendre en compte la création de variants par recombinaison entre deux variants existants. Dans MSY1, il y a 4 variants principaux, les types 1, 2, 3 et 4 (fig. 7.2, page 163), qui résultent de seulement deux événements de substitution de nucléotides. Le type 3 peut par exemple se déduire d'un événement de recombinaison entre un type 1 et un type 4. L'apparition de trois nouveaux variants, les types $1'$, $3'$ et $4'$ (fig. 7.3, page 166), peut s'expliquer par un seul événement de substitution suivi d'événements de recombinaison dans les zones frontières. La prise en compte de tels événements permet de mieux modéliser l'évolution des séquences.

Enfin, il serait intéressant d'évaluer l'amélioration apportée par des algorithmes heuristiques permettant de prendre en compte des modèles évolutifs plus généraux (extensions des événements d'amplification et de contraction ou ajout de nouvelles opérations).

Du point de vue informatique, nous aimerions classer algorithmiquement les problèmes liés aux extensions du modèle SSE. D'autre part, dans ces modèles le coût M de l'opération de mutation est indépendant de la séquence nucléotidique des variants et il peut être intéressant de le faire dépendre des variants. Ceci pose un problème lors du calcul du coût des arches (cf. p. 95), en effet prendre en compte la séquence nucléotidique des variants lors de la compression d'une arche équivaut à retrouver l'arbre le plus parcimonieux (au sens des substitutions de nucléotides) ayant pour feuilles les variants de l'arche et pour racine la graine de l'arche ; ce problème est NP-complet. Cependant, en se limitant à des variants intermédiaires contraints d'appartenir à l'alphabet de départ, ce problème peut se résoudre de manière polynomiale. Une piste est de combiner l'algorithme de Sankoff [Sankoff, 1975] et un algorithme de [Elémento et Gascuel, 2003] à base d'arbres de duplication, définis par ailleurs pour résoudre le problème de l'histoire des duplications. Il est donc intéressant de considérer les rapprochements possibles entre le problème de l'alignement de séquences répétées et le problème de l'histoire des duplications.

En ce qui concerne le logiciel MS_Align, nous souhaitons intégrer rapidement les coûts de mutation dépendants de la séquence nucléotidique des variants ainsi que résoudre le problème lié à l'interface graphique de MS_AlignN. Nous voudrions également que notre logiciel MS_AlignN soit intégré à la grille de calcul dédiée à la génomique, nommée GénoGRID et coordonnée par Dominique Lavenier [`genogrid`].

Nous envisageons une généralisation de notre méthode d'alignement à l'alignement multiple de cartes de minisatellite. Une autre extension de ce travail concerne la recherche d'un ancêtre commun à deux séquences répétées en tandem dans le but de construire directement une phylogénie, sans passer par une matrice de distances.

Annexe A

Détails du jeu de données biologiques

Nous présentons ici, sous forme d'un tableau récapitulatif, le jeu de cartes du minisatellite MSY1 que nous a fourni M. Jobling. C'est sur ce jeu de données que nous avons effectué nos expérimentations. En lignes, on retrouve toutes les populations et en colonne, les haplogroupes contenant au moins cinq cartes. La colonne **Ind.** donne le nombre de cartes dont l'haplogroupe n'est pas déterminé. Un total général de chaque colonne est donné à la fin du tableau. Les haplogroupes 6 et 19 qui contiennent chacun une carte ne sont pas représentés en colonne, mais les individus de ces deux haplogroupes, un San et un Surui sont présents en ligne.

Population	**Nb**	**1**	**2**	**3**	**4**	**8**	**9**	**10**	**11**	**12**	**15**	**16**	**18**	**21**	**22**	**24**	**26**	**Ind.**
Adygean	2		2															
Altai	5																	5
Australian	16	1				1												14
Bantu East	7					4								3				
Bantu SAf	1					1												
Bantu West	1					1												
Basque	38	21	2												15			
Belgian	1																	1
Berber Morocco	1													1				
Biaka Pygmy	1																	1
Bulgarian	13	4	4															5
Bulgarian Gypsy	49	3	34				4											8
Cambodian	1																1	
Cameroon	5																	5
Catalan	7														7			
Caucasian	1																	1
Chinese	12									4								8

à suivre ...

Population	**Nb**	**1**	**2**	**3**	**4**	**8**	**9**	**10**	**11**	**12**	**15**	**16**	**18**	**21**	**22**	**24**	**26**	**Ind.**
Chinese HK	1																	1
Chinese Miao	10																	10
Chinese Turjia	8																	8
Cook Islander	15	6	8														1	
English	25	17	4	1											1			2
European	35	7																28
Finnish	19	1	2	3						1		10						2
French	27	3	7	5			1								4			7
French Bearnais	2														2			
Gambian	8					2								6				
German	2														1		1	
German Bavarian	1	1																
German Gypsy	1		1															
Ghanaian	1																	1
Greek	2																	2
Greek Cypriot	1																	1
Indian	29	10	1	5		1				1	5			1			1	4
Indon Sarawak	18			1												1		16
Irish	3	2	1															
Italian Gypsy	1						1											
Japanese	16				10							1						5
Jewish Ashkenazi	1		1															
Jewish Iraqi	1						1											
Kamba	1																	1
Karitiana	1												1					
Kenyan	8					6								1				1
Maian	1												1					
Malay	1																	1
Malaysian	1																	1
Mbuti	1													1				
Melanesian	1															1		
Mongolian	48	1	1	1	1			10	5	2		23					4	
NigerianYoruba	2					1												1
nk	2	1		1														
Norwegian	3		1									2						
PNG	55							1								34	13	7
Portuguese	1						1											
Quechua	1																	1
Russian	3											3						
Saami	7																	7

à suivre . . .

Population	**Nb**	**1**	**2**	**3**	**4**	**8**	**9**	**10**	**11**	**12**	**15**	**16**	**18**	**21**	**22**	**24**	**26**	**Ind.**
San	1																	hg6
Scottish	10	5		1														4
Siberian Yakut	5											5						
Spanish	5	1													4			
Spanish Galician	2														2			
Spanish Gypsy	1	1																
Spanish Madrid	2														2			
Sri Lankan	1			1														
Surui	14												12					hg19+1
Swedish	1	1																
Tibetan	3				3													
Toh O'Odham	1	1																
UK	6																	6
Ukrainian	1																	1
USA	10	6	3			1												
Utah Mormon	3																	3
Welsh	2																	2
Yakut	13											13						
Zimbabwean	3																	3
Total général	609	93	72	19	14	18	8	11	5	8	5	57	14	13	38	36	21	177

Annexe B

Entrée de MSY1 dans GenBank

Nous donnons ici l'entrée de MSY1 dans la banque de donnée GenBank. On y trouve notamment sont numéro d'accession (ligne 3) et le code nucléotidique de sa séquence (dernier paragraphe).

LOCUS	HSMSATY1 1464 bp DNA PRI 20-JUL-1999
DEFINITION	H.sapiens minisatellite MSY1 (DYF155S1).
ACCESSION	Y07727
VERSION	Y07727.1 GI:2281940
KEYWORDS	minisatellite.
SOURCE	human.
ORGANISM	Homo sapiens
	Eukaryota ; Metazoa ; Chordata ; Craniata ; Vertebrata ; Mammalia ;
	Eutheria ; Primates ; Catarrhini ; Hominidae ; Homo.
REFERENCE	1 (bases 1 to 1464)
AUTHORS	Jobling,M.A., Bouzekri,N. and Taylor,P.G.
TITLE	Hypervariable digital DNA codes for human paternal lineages:
	MVR-PCR at the Y-specific minisatellite, MSY1 (DYF155S1)
JOURNAL	Hum. Mol. Genet. 7 (4), 643-653 (1998)
MEDLINE	98167849
REFERENCE	2 (bases 1 to 1464)
AUTHORS	Jobling,M.A.
TITLE	Direct Submission
JOURNAL	Submitted (30-AUG-1996) M.A. Jobling, University of Leicester,
	Department of Genetics, University Road, Leicester, LE1 7RH, UK
REMARK	Revised by [3]
REFERENCE	3 (bases 1 to 1464)
AUTHORS	Jobling,M.A.

TITLE Direct Submissino
JOURNAL Submitted (25-JUL-1997) M.A. Jobling, University of Leicester, Department of Genetics, University Road, Leicester, LE1 7RH, UK
COMMENT On Jul 28, 1997 this sequence version replaced gi:1524083.
FEATURES Location/Qualifiers
source 1..1464
/organism="Homo sapiens"
/db_xref="taxon:9606"
/chromosome="Y"
/cell_line="human-hamster somatic cell hybrid J640-51 (Jones et al. [1980] Cytogenet. Cell Genet. 28: 181-194)"
/clone_lib="LL0YNC03"
/clone="cosmid M29G11"
/sub_clone="pMSY1"
/note="interval 3E of deletion map (Foote et al. [1992] Science 258: 60-66)"
satellite 393..768
/note="hypervariable minisatellite MSY1 (DYF155S1)"
/rpt_unit=394..419
BASE COUNT 513 a 270 c 210 g 471 t
ORIGIN

1	aagctttggt	ttgtaggaat	tagtgtcaac	attctaataa	ttatttggta	ataaccctgt
61	agttggttga	attctctaaa	aaaaaaaaaa	aaaaaaaaaa	ctatgttgaa	ataaccctga
121	ggacctgtga	ctataacctt	atttggaaat	ataaactttt	caaatataat	caagttaagg
181	taagtcacat	tacattatgg	tataccctaa	attcaatgtc	ttatattgtt	ataaggatag
241	agagttttag	agacacagag	acacagaggg	aagacagcca	tgtgaagaca	gaggtagatg
301	ctgaagcggt	atagctacaa	gctgatgaat	gccaaggata	tattatgcca	atacatatat
361	gccaatacat	atactatgta	tatgtataat	atacacaata	tacatgatgt	atattataca
421	caatatacat	gatgtatatt	atacacaata	tacatgatgt	atattataca	caatatacat
481	gatgtatatt	atacacaata	tacatgatgt	atattataca	caatatacat	gatgtatatt
541	atacacaata	tacatgatgt	atattataca	caatatacat	gatgtatatt	atacacaata
601	tacatgatgt	atattataca	caatatacat	gatgtatatt	atacacaata	tacatgatgt
661	atattataca	caatatacat	gatgtatatt	atacacaata	tacatgatgt	atattataca
721	caatatacat	gatgtatatt	atacataata	tacatcatgt	atattataca	tatgcacaca
781	taaacccctt	tgaataaata	gtattacgct	tccctttccc	tttgtcctag	cttgagttgc
841	ccagaaaaca	agtctgaggt	ttgtttagtg	gggtatatgc	gatacaggac	aagcaacaca
901	tggaaggaag	gagatttact	ataacatttt	atcaagtagg	ccagtttgaa	aggaatttgg
961	gcaattctag	agttaatact	ttatggcaaa	taaactcatc	tgtatgcaca	acaacctcaa
1021	ttaataacct	tcaacctttg	tgcctgaaat	aacacagttt	ctccatagag	gtcactactt
1081	ctcttgcagc	ccacccaagg	agaggccact	tattccctca	cactccatat	actgcaaggc
1141	ctgagcgagg	tcacagattc	ttcttcgtcc	ctttctaata	tcatatcaag	tacactttcc
1201	taaatttatg	acccactccc	tttattctca	caatcttcct	gtccattcca	attcctgtca
1261	atgattacag	tgagtttaac	attcacacaa	aggcccacca	aatcatacca	gttcctgaag
1321	ttcctagttt	ccaaagacct	tctcttacac	tttagttcag	ccatcatctt	tcactgtttt
1381	ctcttaacta	tgttgtttcc	tgatataggt	tctcacgctg	ttgtccaggc	tggagcgcag
1441	tggcatgatt	atagttccct	gcag			

//

Annexe C

Publications

Cette annexe contient la liste de toutes les publications en rapport avec le travail effectué lors de cette thèse. Les articles et éventuellement les présentations qui leur sont associées sont disponibles à l'adresse :

http://www.lirmm.fr/~berard/ dans la rubrique Recherche/Publications.

- Article dans revue
  - Sèverine Bérard, Éric Rivals. **Comparison of Minisatellites**, *Journal of Computational biology (JCB)*, 10(3-4), pages 357-72, Mary-Ann Liebert Inc. publishers, 2003 ;

- Conférence internationale
  - Sèverine Bérard, Éric Rivals. **Comparison of Minisatellites**, *Proceedings of the Sixth Annual International Conference on Computational Biology (RECOMB)*, pages 67-76, Washington, DC, USA, ACM press, 2002 ;

- Conférences nationales
  - Sèverine Bérard, Éric Rivals. **Comparaison de minisatellites**, *Actes des Journées Ouvertes Biologie Informatique Mathématiques (JOBIM)*, pages 261-262, Saint-Malo, 2002 ;
  - Sèverine Bérard, Éric Rivals. **Comparaison de minisatellites**, *Actes de la 24ème réunion annuelle du Groupe de Génétique et Biologie des Populations (PPD)*, page 128, Montpellier, 2002 ;

– Sèverine Bérard, Éric Rivals. **Comparaison de séquences avec amplifications et contractions**, *5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, pages 169-170, Avignon, 2003 ;

- Divers

– Sèverine Bérard. **Reconstruction d'histoire de répétitions en tandem**, *Rapport de DEA*, Tuteur de stage : Éric Rivals, Université Montpellier II, LIRMM, 2000 ;

– Sèverine Bérard. **Alignement de séquences : application à la comparaison de chaînes d'ADN**, *Proceedings J'Docs 2003 XIèmes Journées des Doctorants*, Université Montpellier II, 2003.

Références : informatique et bioinformatique

[Apostolico et al., 1992] Alberto Apostolico, Mikhail J. Atallah, et Susanne E. Hambrusch (1992). New clique and independent set algorithms for circle graphs. *Discrete Applied Mathematics*, 36 : 1–24. — Cité page(s) xvii, 60, 62, 63, 64, 95, 96, 97.

[Bayart, 1995] Benjamin Bayart (1995). Joli manuel pour LaTeX 2_ε, Guide local de l'ESIEE. Connu aussi sous le nom de « l'œil ». — Cité page(s) xvii.

[Behzadi et Steyaert, 2003] Beshshad Behzadi et Jean-Marc Steyaert (2003). An improved algorithm for generalized comparison of minisatellites. Dans *Proceedings of the 14th Annual Symposium of Computational Pattern Matching (CPM03)*, éditeurs : Ricardo Baeza-Yates, Edgar Chávez, et Maxime Crochemore, volume 2676, pages 32–41, Mexico. LNCS Springer. — Cité page(s) 72, 130, 132.

[Benson, 1997] Gary Benson (1997). Sequence alignment with tandem duplication. *Journal of Computational Biology*, 4(3) : 351–67. — Cité page(s) 72, 74, 77.

[Benson, 1999] Gary Benson (1999). Tandem repeats finder : a program to analyze DNA sequences. *Nucleic Acids Research*, 27(2) : 573–580. — Cité page(s) 17.

[Benson et Dong, 1999] Gary Benson et Lan Dong (1999). Reconstructing the duplication history of a tandem repeat. *ISMB*, pages 44–53. — Cité page(s) 17, 72.

[Bérard, 2000] Sèverine Bérard (2000). Mémoire de DEA. Reconstruction d'histoire de répétitions en tandem. Tuteur de stage : Éric Rivals. Université Montpellier II, LIRMM. — Cité page(s) 72.

[Bérard et Rivals, 2002] Sèverine Bérard et Éric Rivals (2002). Comparison of minisatellites. Dans *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 67–76. — Cité page(s) 4, 72, 80, 171.

[Bérard et Rivals, 2003] Sèverine Bérard et Éric Rivals (2003). Comparison of minisatellites. *Journal of Computational Biology*, 10(3-4) : 357–72. — Cité page(s) 4, 72, 80, 171.

[Bergeron et al., 1997] François Bergeron, Gilbert Labelle, et Pierre Leroux (1997). *Combinatorial Species and Tree-like Structures*, volume 67 de *Encyclopedia of mathematics and its applications*. Cambridge University Press. — Cité page(s) 44.

[Cormen et al., 1994] Thomas Cormen, Charles Leiserson, et Ronald Rivest (1994). *Introduction à l'algorithmique*. Practical Approach. DUNOD. — Cité page(s) 34, 44, 154.

[Crochemore et al., 2001] Maxime Crochemore, Christophe Hancart, et Thierry Lecroq (2001). *Algorithmique du texte*. Vuibert, Paris. — Cité page(s) 36, 43, 45.

[Elémento et Gascuel, 2002] Olivier Elémento et Olivier Gascuel (2002). An efficient and accurate distance-based algorithm to reconstruct tandem duplication trees. *Bioinformatics*, 18(Suppl. 2, Proceedings of ECCB'2002) : 92–9. — Cité page(s) 73.

[Elémento et Gascuel, 2003] Olivier Elémento et Olivier Gascuel (2003). An exact and polynomial distance-based algorithm to reconstruct single copy tandem duplication trees. Dans *Proceedings of the 14th Annual Symposium of Combinatorial Pattern Matching (CPM03)*, éditeurs : Ricardo Baeza-Yates, Edgar Chávez, et Maxime Crochemore, volume 2676, pages 96–108, Mexico. LNCS Springer. — Cité page(s) 73, 188.

[Elémento et al., 2002] Olivier Elémento, Olivier Gascuel, et Marie-Paule Lefranc (2002). Reconstructing the duplication history of tandemly repeated genes. *Molecular biology and evolution*, 19(3) : 278–288. — Cité page(s) 73.

[Erlebach et Spieksma, 2002] Thomas Erlebach et Frits C. R. Spieksma (2002). Interval selection : Applications, algorithms, and lower bounds. Rapport technique, ETH, Eidgenössische Technische Hochschule Zürich, Institut für Technische Informatik und Kommunikationsnetze. *http://e-collection.ethbib.ethz.ch/show?type=incoll&nr=711*. — Cité page(s) 62.

[Felsner et al., 1997] Stefan Felsner, Rudolf Müller, et Lorenz Wernish (1997). Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Applied Mathematics*, 74 : 13–32. — Cité page(s) 67, 68, 69, 71, 151.

[Fischetti et al., 1992] Vincent A. Fischetti, Gad M. Landau, Jeanette P. Schmidt, et Peter H. Sellers (1992). Identifying periodic occurrences of a template with applications to a protein structure. Dans *Proceedings of the third annual Symposium of Combinatorial Pattern Matching (CPM92)*, éditeurs : Alberto Apostolico, Maxime Crochemore, Zvi Galil, et Udi Manber, volume 644, pages 111–120, Tucson, Arizona, USA. Springer-Verlag. — Cité page(s) 75.

[Fitch, 1977] Walter M. Fitch (1977). Phylogenies constrained by cross-over process as illustrated by human hemoglobins in a thirteen cycle, eleven amino-acid repeat in human apolipoprotein A-I. *Genetics*, 86(3) : 623–44. — Cité page(s) 73.

[Frank, 1976] András Frank (1976). Some polynomial algorithms for certain graphs and hypergraphs. Dans *Proceedings of the 5th British Combinatorial Conference*, pages 211–26. — Cité page(s) 62.

[Garey et Johnson, 1979] Michael R. Garey et David S. Johnson (1979). *Computers and Intractability – A Guide to the Theory of NP-Completeness*. Freeman, San Francisco. — Cité page(s) 34, 53, 59, 153, 154.

[Gascuel, 1997] Olivier Gascuel (1997). BioNJ : an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular Biology and Evolution*, 14(7) : 685–95. — Cité page(s) 120, 172, 173.

[Gascuel et al., 2003] Olivier Gascuel, Michael Hendy, Alain Jean-Marie, et Robert McLachlan (2003). The combinatorics of tandem duplication trees. *Systematic Biology*, 52 : 110–8. — Cité page(s) 73.

[Gavril, 1973] Fanica Gavril (1973). Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3 : 261–273. — Cité page(s) 55, 56, 60.

[genogrid] Génogrid, une grille pour la génomique. Irisa. *http://www.irisa.fr/symbiose/genogrid/*. — Cité page(s) 188.

[Gordon, 1999] Allan D. Gordon (1999). *Classification*. Chapman & Hall, CRC, Boca Raton, FL, 2e édition. — Cité page(s) 172.

[Gotoh, 1982] Osamu Gotoh (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3) : 705–8. — Cité page(s) 49.

[Gusfield, 1999] Dan Gusfield (1999). *Algorithms on strings, trees and sequences - Computer science and computational biology*. Cambridge university press, 2e édition. — Cité page(s) 36, 37, 42.

[Habib, 2003] Michel Habib (2003). Notes de cours : Complexité Algorithmique. Université Montpellier II, LIRMM. — Cité page(s) 154.

[Hamel, 2002] Sylvie Hamel (2002). *Algorithmes vectoriels et bioinformatique*. Thèse de doctorat, Université du Québec à Montréal. — Cité page(s) 44.

[Higgins et Taylor, 2002] éditeurs : Des Higgins et Willie Taylor (2002). *Bioinformatics - Sequence, structure and databanks*. Practical Approach. Oxford University Press. http://www.oup.co.uk/pas. — Cité page(s) 49.

[Imprimerie Nationale, 2002] éditeur : Imprimerie Nationale (2002). *Lexique des règles typographiques en usage à l'imprimerie nationale*. 5e édition. — Cité page(s) xvii.

[Jaitly et al., 2002] Deep Jaitly, Paul E. Kearney, Guo-Hui Lin, et Bin Ma (2002). Methods for reconstructing the history of tandem repeats and their application to the human genome. *Journal of Computer and System Sciences*, 65(3) : 494–507. — Cité page(s) 73.

[Letondal, 2001] Catherine Letondal (2001). A web interface generator for molecular biology programs in unix. *Bioinformatics*, 17(1) : 73–82. — Cité page(s) 120, 121.

[Levenshtein, 1966] Vladimir I. Levenshtein (1966). Binary codes capable of correcting insertions and reversals. *Soviet Physics Doklady*, 10(8) : 707–10. — Cité page(s) 42.

[Markey, 2002a] Nicolas Markey (2002a). Introduction à LaTeX. — Cité page(s) xvii.

[Markey, 2002b] Nicolas Markey (2002b). Tame the BeaST, BibTeX de B à X .... — Cité page(s) xvii.

[Morgenstern, 1999] Burkhard Morgenstern (1999). DIALIGN 2 : improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15(3) : 211–8. — Cité page(s) 49.

[Muller, 2003] Philippe Muller, Représentation et résolution de problèmes. IRIT (2003). *http://www.irit.fr/~Philippe.Muller/Cours/RO_si/trsp_cours_ia.html*. — Cité page(s) 34.

[Myers et Miller, 1989] Eugene Webb Myers et W. Miller (1989). Approximate matching of regular expressions. *Bulletin of mathematical biology*, 51(1) : 5–37. — Cité page(s) 75.

[Needleman et Wunsch, 1970] Saul B. Needleman et Christian D. Wunsch (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48 : 443–53. — Cité page(s) 44.

[Perrière et Gouy, 1996] Guy Perrière et Manolo Gouy (1996). WWW-Query : An on-line retrieval system for biological sequence banks. *Biochimie*, 78 : 364–369. http://pbil.univ-lyon1.fr/software/njplot.html. — Cité page(s) 172.

[Phylip] PHYLIP : free package of programs for inferring phylogenies. University of Washington. *http://evolution.genetics.washington.edu/phylip.html*. — Cité page(s) xvii, 120, 172.

[Rivals, 2004] Éric Rivals (2004). A survey on algorithmic aspects of tandem repeats evolution. *International Journal of Foundations of Computer Science*, Special Issue "Combinatorics on Words with Applications". — Cité page(s) 17.

[Sankoff, 1975] David Sankoff (1975). Minimal mutation trees of sequences. *SIAM Journal of Applied Mathematics*, 28 : 35–42. — Cité page(s) 188.

[Sankoff, 2000] David Sankoff (2000). The early introduction of dynamic programming into computational biology. *Bioinformatics*, 16(1) : 41–7. — Cité page(s) 44.

[Sankoff et Kruskal, 1999] éditeurs : David Sankoff et Joseph B. Kruskal (1999). *Time Warps, String Edits and Macromolecules : the Theory and Practice of Sequence Comparison*. CSLI Publications, 2^{e} édition. — Cité page(s) 36, 37, 38, 41, 82, 87.

[Setubal et Meidanis, 1997] João Carlos Setubal et João Meidanis (1997). *Introduction to computational biology*. PWS publishing company. — Cité page(s) 36.

[Smith et Waterman, 1981] Temple F. Smith et Michael S. Waterman (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147 : 195–197. — Cité page(s) 44.

[Stewart] Lorna Stewart, Graph class. University of Alberta. *http://www.cs.ualberta.ca/~stewart/GRAPH/*. — Cité page(s) 51.

[Stoye et al., 1997] Jens Stoye, Vincent L. Moulton, et Andreas W. M. Dress (1997). DCA : an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. *Computer Applications in the Biosciences*, 13(6) : 625–6. — Cité page(s) 49.

[Tang et al., 2002] Mengxiang Tang, Michael Waterman, et Shibu Yooseph (2002). Zinc finger gene clusters and tandem gene duplication. *Journal of computational biology*, 5(3) : 429–446. — Cité page(s) 73.

[Thompson et al., 1994] Julie D. Thompson, Desmond G. Higgins, et Toby J. Gibson (1994). CLUSTAL W : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22 : 4673–80. — Cité page(s) 49.

[Varré et al., 1999] Jean-Stéphane Varré, Jean-Paul Delahaye, et Éric Rivals (1999). Transformation distances : a family of dissimilarity measures on movements of segments. *Bioinformatics*, 15 : 194–202. — Cité page(s) 74.

[Vialette, 2001] Stéphane Vialette (2001). *Aspects algorithmiques de la prédiction des structures secondaires d'ARN*. Thèse de doctorat, Université Paris 7 - Denis Diderot. — Cité page(s) 67, 68, 150.

[Yannakakis, 1978] Mihalis Yannakakis (1978). Node- and edge-deletion NP-complete problems. Dans *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 253–64, San Diego, California. — Cité page(s) 153.

Références : biologie

(Alonso et Armour, 1998) Santos Alonso et John A. Armour (1998). MS205 minisatellite diversity in Basques : Evidence for a pre-neolithic component. *Genome Research*, 8(12) : 1289–98. — Cité page(s) xvii, 169, 170.

(Amarger et al., 1998) Valérie Amarger, Dominique Gauguier, Martine Yerle, Françoise Apiou, Philippe Pinton, Fabienne Giraudeau, Sylvaine Monfouilloux, Mark Lathrop, Bernard Dutrillaux, Jérôme Buard, et Gilles Vergnaud (1998). Analysis of distribution in the human, pig, and rat genomes points toward a general subtelomeric origin of minisatellite structures. *Genomics*, 52(1) : 62–71. — Cité page(s) 17.

(Andreassen et al., 2002) Rune Andreassen, Jan Lundsted, et Bjornar Olaisen (2002). Mutation at minisatellite locus DYF155S1 : allele length mutation rate is affected by age of progenitor. *Electrophoresis*, 23(15) : 2377–83. — Cité page(s) 157, 162, 165, 170, 171, 185, 187.

(Armour et al., 1996) John A. Armour, Tiiu Anttinen, Celia A. May, Emilce E. Vega, Antti Sajantila, Judith R. Kidd, Kenneth K. Kidd, Jaume Bertranpetit, Svante Pääbo, et Alec J. Jeffreys (1996). Minisatellite diversity supports a recent African origin for modern humans. *Nature Genetics*, 13(2) : 154–60. — Cité page(s) 29.

(Armour et al., 1993) John A. Armour, Peter C. Harris, et Alec J. Jeffreys (1993). Allelic diversity at minisatellite MS205 (D16S309) : evidence for polarized variability. *Human Molecular Genetics*, 2 : 1137–1145. — Cité page(s) 169, 170.

(Bahuchet, 1994) Serge Bahuchet (1994). *Situations des populations indigènes des forêts denses et humides.* Office des publications officielles des communautés européennes, Luxembourg. — Cité page(s) 180.

(Bennett et al., 1997) Simon T. Bennett, Amanda J. Wilson, Laura Esposito, Nourdine Bouzekri, Dag E. Undlien, Francesco Cucca, Lorenza Nisticó, Raffaella Buzzetti, the IMDIAB Group, Emanuele Bosi, Flemming Pociot, Jørn Nerup, Anne Cambon-Thomsen, Alberto Pugliese, Julian P.H. Shield, Patricia A. McKinney, Stephen C. Bain, Constantin Polychronakos, et John A. Todd (1997). Insulin VNTR allele-specific effect in type I diabetes depends on identity of untransmitted paternal allele. *Nature Genetics*, 17(3) : 350–52. — Cité page(s) 185.

(`Bernot et Alibert`) Alain Bernot et Olivier Alibert, La naissance de la biologie moléculaire. Genoscope. *http://www.genoscope.cns.fr/externe/HistoireBM/.* — Cité page(s) 8.

(`Biotech-canada`) Centre scientifique de la biotechnologie - Composantes biologiques. Industrie Canada. *http://strategis.ic.gc.ca/SSGF/tc00007f.html*. — Cité page(s) 9.

(Boden et al., 1989) J.-P. Boden, J.-N. Cloarec, B. Gaudin, J. Lamarque, P. Lamarque, C. Lizeaux, R. Tavernier, A. Vareille, M. Vareille, et A. Videaud (1989). *Biologie Terminale D*. Bordas, Paris. — Cité page(s) 9, 11, 13, 15.

(Bouzekri et al., 1998) Nourdine Bouzekri, Paul G. Taylor, Michael F. Hammer, et Mark A. Jobling (1998). Novel mutation processes in the evolution of a haploid minisatellite, MSY1 : array homogenization without homogenization. *Human Molecular Genetics*, 7(4) : 655–9. — Cité page(s) 157, 165, 166.

(Bríon et al., 2002) María Bríon, Ricardo Cao-Abad, Antonio Salas, M^a Victoria Lareu, et Angel Carracedo (2002). New method to measure minisatellite variant repeat variation in population genetic studies. *American Journal of Human Biology*, 14 : 421–28. — Cité page(s) 157, 159, 183.

(Buard et al., 1998) Jérôme Buard, Agnès Bourdet, Jane Yardley, Yuri E. Dubrova, et Alec J. Jeffreys (1998). Influences of array size and homogeneity on minisatellite mutation. *The EMBO Journal*, 17(12) : 3495–502. — Cité page(s) 158, 159.

(Buard et al., 2002) Jérôme Buard, Charles Brenner, et Alec J. Jeffreys (2002). Evolutionary fate of an unstable human minisatellite deduced from sperm-mutation spectra of individual alleles. *American Journal of Human Genetics*, 70 : 1038–43. — Cité page(s) 158.

(Buard et Jeffreys, 1997) Jérôme Buard et Alec J. Jeffreys (1997). Big, bad minisatellites. *Nature Genetics*, 15(4) : 327–8. — Cité page(s) 17, 25.

(Buard et al., 2000) Jérôme Buard, Angela C. Shone, et Alec J. Jeffreys (2000). Meiotic recombination and flanking marker exchange at the highly unstable human minisatellite CEB1 (D2S90). *American Journal of Human Genetics*, 67 : 333–44. — Cité page(s) 25, 26.

(Buard et Vergnaud, 1994) Jérôme Buard et Gilles Vergnaud (1994). Complex recombinaison events at the hypermutable minisatellite CEB1 (D2S90). *Embo Journal*, 13 : 3203–10. — Cité page(s) 169.

(Cavalli-Sforza, 2000) Luigi Luca Cavalli-Sforza (2000). *Genes, Peoples, and Languages*. University of California Press, Ltd. Traducteur : Mark Seielstad. — Cité page(s) 180.

(Cohen et al., 1993) Daniel Cohen, Ilia Chumakov, et Jean Weissenbach (1993). A first-generation physical map of the human genome. *Nature*, 366(6456) : 698–701. — Cité page(s) 29.

(de Vienne, 1998) Dominique de Vienne (1998). *Les marqueurs moléculaires en génétique et biotechnologies végétale*. INRA, Versailles, France. — Cité page(s) 15.

(`Diop, 2002`) Michel Bakar Diop, Bébés confondus de Abass Ndao : le test ADN est-il nécessaire ? AllAfrica Global Media (2002). *http://fr.allafrica.com/stories/-200212030631.html*. — Cité page(s) 9.

(Dunger et al., 1998) David B. Dunger, Ken K. L. Ong, Stewart J. Huxtable, Andrea Sherriff, Kathryn A. Woods, Marion L. Ahmed, Jean Golding, Marcus E. Pembrey, Sue Ring, the ALSPAC Study Team, Simon T. Bennett, et John A. Todd (1998). Association of the INS VNTR with size at birth. *Nature Genetics*, 19(1) : 98–100. — Cité page(s) 185.

(Fontanillas, 2002) Éric Fontanillas (2002). Utilisation d'un modèle informatique d'alignement de cartes MVR pour l'étude phylogénétique du minisatellite humain MS205. Mémoire de Maîtrise *Physiologie Végétale Appliquée*. Tuteur de stage : Éric Rivals. Université Montpellier II, LIRMM. — Cité page(s) 80, 121, 187.

(`Gayon`) Jean Gayon, Histoire de l'hérédité et de la génétique. Cité des Sciences et de l'Industrie. *http://www.cite-sciences.fr/francais/ala_cite/expo/tempo/defis/histoire/-index2.htm*. — Cité page(s) 8.

(`Genopole`) Bref historique de la génétique. Génopole. *http://www.genopole.org/html/fr/-comprendre/*. — Cité page(s) 30.

(`Genoscope`) Le séquençage des génomes. Genoscope. *http://www.genoscope.cns.fr/-externe/Francais/Sequencage/*. — Cité page(s) 2, 16, 17.

(Gill et al., 1985) Peter Gill, Alec J. Jeffreys, et David J. Werrett (1985). Forensic applications of DNA fingerprints. *Nature*, 318(6046) : 577–9. — Cité page(s) 29.

(Goldstein et Schlotterer, 1999) éditeurs : David B. Goldstein et Christian Schlotterer (1999). *Microsatellites : Evolution and Applications*. Oxford University Press. — Cité page(s) 17.

(`Gourde`) Sylvie Gourde, La génétique : de Mendel au clonage - Notions de bases. Cybersciences. *http://www.cybersciences.com/cyber/1.0/1_171_Menu.asp*. — Cité page(s) 9.

(Graur et Li, 2000) Dan Graur et Wen-Hsiung Li (2000). *Fundamentals of molecular evolution*. Sinauer, Sunderland, Massachusetts, 2^{e} édition. — Cité page(s) 9, 22, 23, 24, 25.

(Hill et Jeffreys, 1985) A. V. Hill et Alec J. Jeffreys (1985). Use of minisatellite DNA probes for determination of twin zygosity at birth. *Lancet*, 2(8469-70) : 1394–5. — Cité page(s) 29.

(Holmes et Page, 1998) Edward C. Holmes et Roderic D. Page (1998). *Molecular evolution : a phylogenic approach*. Blackwell publishing. — Cité page(s) 17.

(Hurles et al., 1998) Matthew E. Hurles, Catherine Irven, Jayne Nicholson, Paul G. Taylor, Fabricio R. Santos, John Loughlin, Mark A. Jobling, et Bryan C. Sykes (1998). European Y-chromosomal lineages in Polynesians : a contrast to the population structure revealed by mtDNA. *American Journal of Human Genetics*, 63(6) : 1793–806. — Cité page(s) 169, 170.

(Hurles et al., 2002) Matthew E. Hurles, Jayne Nicholson, Elena Bosch, Colin Renfrew, Bryan C. Sykes, , et Mark A. Jobling (2002). Y chromosomal evidence for the origins of oceanic-speaking peoples. *Genetics*, 160 : 289–303. — Cité page(s) 170.

(Jacob et Monod, 1961) François Jacob et Jacques Monod (1961). Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3 : 318–56. — Cité page(s) 9.

(Jeffreys et al., 1997) Alec J. Jeffreys, Philippe Bois, Jérôme Buard, Andrew Collick, Yuri E. Dubrova, Caroline R. Hollies, Celia A. May, John Murray, David L. Neil, Rita Neumann, John D. Stead, Keiji Tamaki, et Jane Yardley (1997). Spontaneous and induced minisatellite instability. *Electrophoresis*, 18(9) : 1501–11. — Cité page(s) 27, 30, 157, 158.

(Jeffreys et al., 1991) Alec J. Jeffreys, Annette MacLeod, Keiji Tamaki, David L. Neil, et Darren G. Monckton (1991). Minisatellite repeat coding as a digital approach to DNA typing. *Nature*, 354(6350) : 204–9. — Cité page(s) 30, 157, 169.

(Jeffreys et al., 1994) Alec J. Jeffreys, Keiji Tamaki, Annette MacLeod, Darren G. Monckton, David L. Neil, et John A. Armour (1994). Complex gene conversion events in germline mutation at human minisatellites. *Nature Genetics*, 6 : 136–145. — Cité page(s) 169.

(Jeffreys et al., 1985a) Alec J. Jeffreys, Vicky Wilson, et Swee Lay Thein (1985a). Hypervariable minisatellite regions in human DNA. *Nature*, 314(6006) : 67–73. — Cité page(s) 29.

(Jeffreys et al., 1985b) Alec J. Jeffreys, Vicky Wilson, et Swee Lay Thein (1985b). Individual-specific fingerprints of human DNA. *Nature*, 316(6023) : 76–9. — Cité page(s) 29.

(Jobling et al., 1998) Mark A. Jobling, Nourdine Bouzekri, et Paul G. Taylor (1998). Hypervariable digital DNA codes for human paternal lineages : MVR-PCR at the Y-specific minisatellite, MSY1 (DYF155S1). *Human Molecular Genetics*, 7(4) : 643–53. — Cité page(s) 17, 80, 133, 157, 161, 163, 170, 171, 184, 187.

(Jobling et Tyler-Smith, 1995) Mark A. Jobling et Chris Tyler-Smith (1995). Fathers and sons : the Y chromosome and human evolution. *Trends in Genetics*, 11(11) : 449–56. — Cité page(s) 170, 174.

(Jobling et Tyler-Smith, 2000) Mark A. Jobling et Chris Tyler-Smith (2000). New uses for new haplotypes the human Y chromosome, disease and selection. *Trends in Genetics*, 16(8) : 356–62. — Cité page(s) 171, 173, 183, 184.

(`Laurewce`) Jean-Jacques Laurewce, Génome, mode d'emploi. Centre de Culture Scientifique Technique et Industrielle de Grenoble. *http://www.ccsti-grenoble.org/webtemp/-conf5.htm*. — Cité page(s) 8.

(Li, 1997) Wen-Hsiung Li (1997). *Molecular evolution*. Sinauer Associates, incorporated. — Cité page(s) 17.

(Maftah et Julien, 1999) Abderrahman Maftah et Raymond Julien (1999). *Biologie moléculaire*. DUNOD, Paris, 2^{e} édition. — Cité page(s) 9.

(May et al., 1996) Celia A. May, Alec J. Jeffreys, et John A. Armour (1996). Mutation rate heterogeneity and the generation of allele diversity at the human minisatellite MS205 (D16S309). *Human Molecular Genetics*, 5 : 1823–33. — Cité page(s) 169.

(Mendel, 1866) Grégor Mendel (1866). Versuche über pflanzen-hybriden. *Verhandlungen des Naturforschenden Vereins*, 4 : 3–47. — Cité page(s) 8.

(Mendel, 1961) Grégor Mendel (1961). L'œuvre de Grégor Mendel : Recherches sur divers hybrides végétaux. *Bulletin de « l'Union des Naturalistes de l'Enseignement Public »*, 3 : 1–37. http://www.cndp.fr/magsvt/genes/mendel.htm. — Cité page(s) 8.

(Nirenberg et al., 1963) Marshall W. Nirenberg, J. Heinrich Matthaei, Oliver W. Jones Jr., Robert G. Martin, et Samuel H. Barondes (1963). Approximation of genetic code *via* cell-free protein synthesis directed by template RNA. *Federation Proceedings*, 22 : 55–61. — Cité page(s) 9.

(Ong et al., 1999) Ken K.L. Ong, David I. Phillips, Caroline Fall, Jo Poulton, Simon T. Bennett, Jean Golding, John A. Todd, et David B. Dunger (1999). The insulin gene VNTR, type II diabetes and birth weight. *Nature Genetics*, 21(3) : 262–63. — Cité page(s) 185.

(Quintana-Murci et al., 1999) Lluís Quintana-Murci, Reiner Veita, Silvana Santachiara-Benerecetti, Ken McElreavey, Marc Fellous, et Thomas Bourgeron (1999). L'ADN mitochondrial, le chromosome Y et l'histoire des populations humaines. *Médecine/sciences (MS)*, 15 : 974–82. — Cité page(s) 27.

(Ruhlen, 1987) Merritt Ruhlen (1987). *A guide to the world's languages*. Stanford University Press. — Cité page(s) 180.

(Sancristobal-Gaudy et al., 2000) M. Sancristobal-Gaudy, G. Renand, Y. Amigue, M.-Y. Boscher, H. Leveziel, et B. Bibe (2000). Traçabilité individuelle des viandes bovines à l'aide de marqueurs génétiques. *Productions animales*, 13 : 269–76. — Cité page(s) 29.

(Sanger et al., 1977) Frederick Sanger, S. Nicklen, et A. R. Coulson (1977). DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Science, USA (PNAS)*, 74(12) : 5463–7. — Cité page(s) 28.

(Seielstad et al., 1994) Mark T. Seielstad, J. M. Hebert, Alice A. Lin, Peter A. Underhill, M. Ibrahim, Douglas Vollrath, et Luigi Luca Cavalli-Sforza (1994). Construction of human Y-chromosomal haplotypes using a new polymorphic A to G transition. *Human Molecular Genetics*, 3(12) : 2159–61. — Cité page(s) 184.

(Semino et al., 1996) Ornelles Semino, G. Passarino, A. Brega, M. Fellous, et A. S. Santachiara-Benerecetti (1996). A view of the neolithic demic diffusion in europe through two Y chromosome-specific markers. *American Journal of Human Genetics*, 59(4) : 964–8. — Cité page(s) 27.

(Stead et al., 2000) John D. Stead, Jérôme Buard, John A. Todd, et Alec J. Jeffreys (2000). Influence of allele lineage on the role of the insulin minisatellite in susceptibility to type I diabetes. *Human Molecular Genetics*, 9(20) : 2929–35. — Cité page(s) 185.

(Stead et Jeffreys, 2000) John D. Stead et Alec J. Jeffreys (2000). Allele diversity and germline mutation at the insulin minisatellite. *Human Molecular Genetics*, 9(5) : 713–23. — Cité page(s) 169, 185.

(Stead et Jeffreys, 2002) John D. Stead et Alec J. Jeffreys (2002). Structural analysis of insulin minisatellite alleles reveals unusually large differences in diversity between Africans and non-Africans. *American Journal of Human Genetics*, 71(6) : 1273–84. — Cité page(s) 170, 185.

(Tamaki et al., 1999) Keiji Tamaki, Celia A. May, Yuri E. Dubrova, et Alec J. Jeffreys (1999). Extremely complex repeat shuffling during germline mutation at human minisatellite B6.7. *Human Molecular Genetics*, 8(5) : 879–88. — Cité page(s) 169.

(The Y Chromosome Consortium, 2002) The Y Chromosome Consortium (2002). A nomenclature system for the tree of human Y-chromosomal binary haplogroups. *Genome Research*, 12 : 339–48. — Cité page(s) 162, 183, 187.

(Underhill et al., 1997) Peter A. Underhill, Li Jin, Alice A. Lin, S. Qasim Mehdi, Trefor Jenkins, Douglas Vollrath, Ronald W. Davis, Luigi Luca Cavalli-Sforza, et Peter J. Oefner (1997). Detection of numerous Y chromosome biallelic polymorphisms by denaturing high performance liquid chromatography. *Genome Research*, 7 : 996–1005. — Cité page(s) 184.

(Underhill et al., 1996) Peter A. Underhill, Li Jin, Rachel Zemans, Peter J. Oefner, et Luigi Luca Cavalli-Sforza (1996). A pre-columbian human Y chromosome-specific C to T transition and its implications for human evolution. *Proceedings of the National Academy of Science, USA (PNAS)*, 93 : 196–200. — Cité page(s) 184.

(Vafiadis et al., 1997) Petros Vafiadis, Simon T. Bennett, John A. Todd, Joseph Nadeau, Rosemarie Grabs, Cynthia G. Goodyer, Saman Wickramasinghe, Eleanor Colle, et Constantin Polychronakos (1997). Insulin expression in human thymus is modulated by INS VNTR alleles at the IDDM2 locus. *Nature Genetics*, 15(3) : 289–92. — Cité page(s) 185.

(Vergnaud et Denoeud, 2000) Gilles Vergnaud et France Denoeud (2000). Minisatellites : mutability and genome architecture. *Genome Research*, 10(7) : 899–907. — Cité page(s) 17, 30.

Index

Symboles

Σ 36
α 52
α_ω 53
. 36

A

acide aminé 14
ADN 8–16, 18–23, 25, 28–30
alignement 35, 39, 43–45, 85, 87, 90
  avec brèches 49
  global 44, 48
  local 48
  multiple 49
  semi-global 48
allèle 8, 14, 15, 20, 26, 29
amplification 125
arche 84, 85, 125
  approchée 127, 130
  complexe 93, 127
  d'ordre supérieur 133, 139, 140
  EOS 138
  exacte 127, 130
  fantôme 107–109
  intérieur 143, 144
  pied 126
  pied généré 143, 144
  pied source 143, 144
  simple 93
  visible 108
arité 125, 127, 130
ARNm 9
autosome 10

B

brassage génétique 18
brèche 39, 49

C

carte de minisatellite . . 157, 159, 163, 166
carte génétique 8, 29
cartographie 9
cellule 9, 10, 18, 19, 30
  eucaryote 9
  germinale 10, 20
  haploïde 19, 20
  procaryote 9
  somatique 10, 20
centromère 10, 16, 18
chaîne 36
chromatide 10, 14, 18–21, 23
chromosome . . 8, 10, 14, 18–21, 23, 25, 29
  homologue 10, 14, 19–21
chronologie 147
  de poids maximal 147–149
chronologiquement compatible 144
codage α 63, 97
code génétique 9, 14
codon 14, 21
compatible 93
complexe 93
complexité algorithmique 34
compression 86, 92, 109, 127
concaténation 36
contraction 125
conversion génique 21, 22
crossing-over 21, 22, 25
crossing-over inégal 22, 23, 25
cytoplasme 10

D

densité 54

2-intervalles 67, 150
diploïde 10
distance 35, 41, 44
    d'édition 41–43
    d'édition généralisée 42–44
    de Levenshtein 42
    SSE 81
dominant 8, 14, 15
double intervalle 68, 70, 151

E

empreinte génétique 29, 30
exon 14

F

facteur 36
famille d'intervalles 54–56, 97
    densité 54

G

génotype 18
gamète 10
gène 8, 9, 14, 17, 18, 20, 29, 30
génération 86, 92, 127
génétique 8, 9, 30
génie génétique 9, 30
génome 8–10, 17, 29, 30
glissement à la réplication 23
graine 84
graphe 51
    arête 51
    sommet 51
    sous-graphe 52
    sous-graphe induit 52
    stable 52
graphe d'intervalles 53, 54
graphe de 2-intervalles 67, 150
graphe de chevauchement . . 54–56, 60, 96
graphe de cordes 55, 56
graphe de croisement 68–70, 151
graphe de trapézoïdes circulaires . 69, 151

H

haploïde 10
hétérozygote 14, 15
homoplasie 164
homozygote 14, 15

I

incompatible 94
intron 14, 16

L

listing 38, 40, 41, 87, 126
locus 14, 15, 29

M

marqueur génétique 28, 29
matrice de prog. dyn. 44–46, 88, 89
méiose 19, 20, 28
microsatellite 16, 17, 29
minisatellite 15–17, 29, 30
mitose 18, 19
mutation 13, 18, 20, 25, 28, 30, 31
    délétion 20, 22
    insertion 20, 22
    inversion 20, 25
    mutation ponctuelle 20, 21
    recombinaison 20–22
    substitution 20, 21
MVR-PCR 30

N

nombre de stabilité 52
NP-complet 35
nucléotide 11–15, 20, 21, 28, 29

O

opération d'édition 41
opération élémentaire 37, 38, 80, 125
    A_M 82, 125
    amplification 38, 81, 125
    contraction 38, 81, 125
    délétion 38, 42, 80, 125
    insertion 38, 42, 80, 125
    inversion 38
    M_C 82, 125
    mutation 80, 125

substitution 37, 38, 42
ordre 125, 127, 130

P

paire alignée 39, 43
PCR 29, 30
phénotype 14
pied d'une arche 126
pied généré 143, 144
pied source 143, 144
pointeur 45, 46
polymorphisme 15, 17, 28, 29
polynômial 35
préfixe 36
programmation dynamique 44
matrice 44–46, 88, 89
protéine 9, 14, 21
synthèse 14

R

récessif 8, 15
recombinaison 20–22
réparation 13, 20, 22
répétition en tandem 17, 23
réplication 12, 13, 20, 23, 27
RFLP 29

S

satellite 16
séquençage 9, 28, 30
séquence 35, 36
séquence répétée *(bio)* 16, 29
dispersée 16
en tandem 16
séquence répétée en tandem *(info)* 72
alignement 73
carte de minisatellite 130
histoire des duplications 72
simple 93
SNP 15, 29
sous-chaîne 36
sous-séquence 36
SSE 80, 81
stable 52
stable max 53, 59
graphe de 2-intervalles 67, 150
graphe de chevauchement 60, 96
graphe de croisement 69, 151
synthèse de protéines 14

T

télomère 10, 16, 17
trace 38, 40, 41
traduction 9
transcription 9
trapézoïde circulaire 69, 151
trou 39, 49

Glossaire

Σ : Symbole représentant un alphabet. — Page 36.

. : Symbole de la concaténation entre deux chaînes. — Page 36.

Acide aminé : Petites molécules chimiques qui composent les protéines. Il en existe 20. — Page 14.

ADN : ACIDE DÉSOXYRIBONUCLÉIQUE. Molécule géante (macromolécule) formée de l'assemblage linéaire de quatre nucléotides et dont la structure a été élucidée par J. Watson et F. Crick en 1953. C'est le principal composant des chromosomes et le support biochimique des caractères héréditaires. — Page 11.

Allèle : Contraction d'allélomorphe. Les allèles sont les différentes formes sous lesquelles peut apparaître un locus de la séquence d'ADN. — Page 14.

Arche : Soit s une carte de longueur n et i, j deux entiers tels que $1 \leq i < j \leq n$. $s[i..j]$ est une arche de s si $s[i] = s[j]$. — Page 85.

Arité : L'arité d'une amplification est le nombre de copie(s) qu'elle produit. — Page 125.

Autosome : Se dit d'un chromosome qui n'est pas un chromosome sexuel. — Page 10.

Carte génétique : Représentation de la disposition des gènes le long de la molécule d'ADN en utilisant un ensemble de repères moléculaires identifiés et positionnés le long de chaque chromosome. Ces repères (marqueurs) permettent de localiser plus facilement les gènes sur les différents chromosomes. — Page 29.

Cellule : Unité structurale et fonctionnelle de tous les êtres vivants. — Page 9.

Cellule germinale : Cellule reproductrice comme les gamètes. — Page 10.

Cellule somatique : Cellule du corps par opposition aux cellules germinales qui sont les cellules reproductrices. — Page 10.

Centromère : Zone par laquelle sont reliées les deux chromatides d'un chromosome. — Page 10.

Chaîne : Une chaîne s de longueur n définie sur une alphabet Σ est une séquence de n symboles de Σ indexés de 1 à n. *(synonyne(s) : séquence).* — Page 36.

Chromatide : Chacune des unités résultant de la réplication d'un chromosome et observables lors de la mitose ou de la méiose. — Page 10.

Chromosome : Forme que prend l'ADN pendant la division cellulaire (aspect de fins bâtonnets). Il se présente souvent sous la forme de 2 bras : un bras long et un bras court. — Page 10.

Code génétique : Ensemble des règles de correspondance et des signes de ponctuation permettant la traduction du « langage » des acides nucléiques (codons formés de trois nucléotides dans l'ADN) en celui des protéines (acides aminés). — Page 14.

Codon : Unité du code génétique composée par un groupe de trois nucléotides présent sur une molécule d'ADN. Un codon traduit un acide aminé précis ou possède une fonction de régulation (exemple : signal de fin de traduction ou codon stop). — Page 14.

Compatible : Se dit de deux arches qui peuvent être compressées ou générées dans le même alignement. VOIR AUSSI : ARCHE, GÉNÉRATION, COMPRESSION. — Page 94.

Complexe : Une arche est dite complexe si elle contient d'autres arches. Complexe est le contraire de simple. — Page 93.

Compression : Opération d'arches consistant à réduire une arche en son variant ancêtre. Cette opération est notée K. VOIR AUSSI : ARCHE, GÉNÉRATION. — Page 86.

Concaténation : Opération sur les chaînes notée par un . ; la concaténation d'une chaîne s de longueur n et d'une chaîne r de longueur m est la chaîne de longueur $n + m$ $s.r = s[1]s[2]\dots s[n]r[1]\dots r[m-1]r[m]$. VOIR AUSSI : CHAÎNE. — Page 36.

Conversion génique : Transfert non réciproque d'information génétique. — Page 21.

Crossing-over : Mélange entre deux chromosomes homologues conduisant à un échange de fragments d'ADN. Il entraîne d'importants remaniements du patrimoine génétique. — Page 21.

Crossing-over inégal : Crossing-over dans lequel l'échange de matériel génétique n'est pas équitable, une chromatide perd un segment d'ADN, tandis que l'autre le gagne. — Page 22.

Cytoplasme : C'est une substance homogène dans laquelle baignent les organites (noyau, mitochondries, etc.). Le rôle du cytoplasme est la circulation des matériaux nécessaires à la cellule, le transport des substances élaborées par la cellule ainsi que le transport des déchets liés à l'activité cellulaire. — Page 10.

Délétion : Mutation entraînant la perte d'un fragment d'ADN (ou d'un gène). — Page 22.

Densité : La densité d'une famille d'intervalles est le nombre maximum d'intervalles couvrant un point de la droite réelle. — Page 54.

Diploïde : Qui comporte deux représentants homologues de chaque chromosome. — Page 10.

Dominant : Le fait, pour un allèle, de s'exprimer phénotypiquement à l'état hétérozygote comme à l'état homozygote. — Page 14.

Empreinte génétique : L'empreinte digitale génétique (*DNA-fingerprinting*) est un procédé de la biologie moléculaire qui permet d'établir les caractéristiques génétiques propres à un individu. Mis à part chez les vrais jumeaux, chaque empreinte génétique est unique. Les comparaisons d'empreintes génétiques permettent ou peuvent simplifier par exemple l'identification dans les procès en paternité ou dans la recherche d'un criminel. — Page 29.

Exon : Partie codante de l'ADN au sein d'un gène. — Page 14.

Facteur : Voir sous-chaîne. — Page 36.

Famille d'intervalles : Une famille d'intervalles est un ensemble d'intervalles de la droite réelle. — Page 53.

Fluorochrome : Les fluorochromes sont des molécules cycliques capables d'émettre un rayonnement si elles sont excitées par une source (par exemple un faisceau laser). La longueur d'onde d'émission est spécifique du fluorochrome et est détectable par un système d'analyse. Elles sont utilisées dans les séquenceurs automatiques. — Page 28.

Gamète : Cellule sexuelle haploïde spécialisée dans la réalisation de la fécondation. — Page 10.

Gène : Une séquence d'ADN possédant une fonction spécifique, codant pour une protéine. Il correspond à un locus sur le génome. Un gène peut comporter de quelques centaines à plusieurs centaines de milliers de bases. — Page 14.

Génération : Opération d'arches consistant à générer une arche depuis son variant ancêtre. Cette opération est notée G. voir aussi : arche, compression. — Page 86.

Génétique : Partie de la science du ou des gènes (structure, fonctions, évolution) et de la transmission des caractères héréditaires. — Page 8.

Génie génétique : Partie de la biologie qui utilise les outils de la biologie moléculaire pour modifier la structure ou le fonctionnement d'un ou de plusieurs gènes. Par exemple, certains segments de chromosomes contenant un gène d'intérêt peuvent être isolés et intégrés au génome d'autres organismes (bactéries, levures, plantes, etc.). Le génie génétique permet ainsi d'obtenir, à partir de bactéries, la synthèse et la fabrication industrielle de substances telles que les hormones, les vaccins, etc. — Page 30.

Génome : Ensemble de toute l'information génétique d'un individu contenue dans chacune de ses cellules. — Page 10.

Génotype : Ensemble de tous les gènes d'un individu, c'est-à-dire l'ensemble des allèles qu'il porte, qu'ils s'expriment ou non. — Page 18.

Graine : Nom donné au variant ancêtre d'une arche. voir aussi : arche, variant. — Page 84.

Haploïde : Qui ne comporte qu'un seul exemplaire de chaque chromosome. — Page 10.

Hétérozygote : Se dit, pour une cellule ou un individu diploïde, du fait de posséder 2 copies différentes du même gène. — Page 14.

Homologue : Se dit de deux chromosomes de la même paire. — Page 10.

Homoplasie : Se dit de séquences ou sites présentant des états identiques mais ayant subi différentes étapes évolutives ou différents types de mutations. La présence d'homoplasie conduit à sous-estimer le nombre total de mutations s'étant produites au cours du temps. — Page 164.

Homozygote : Se dit, pour une cellule ou un individu diploïde, du fait de posséder deux copies identiques du même gène. — Page 14.

Incompatible : Contraire de compatible. Se dit de deux arches qui ne peuvent pas être compressées ou générées dans le même alignement. VOIR AUSSI : ARCHE, GÉNÉRATION, COMPRESSION. — Page 94.

Insertion : Mutation résultant de l'ajout d'un ou plusieurs nucléotides dans la séquence d'ADN. — Page 22.

Intron : Partie non codante du gène. — Page 14.

Inversion : Type de mutation dans lequel un segment du chromosome se détache et se rattache à l'envers. — Page 25.

Listing : Un listing est une table dans laquelle la première et la dernière ligne sont deux séquences à aligner, chaque ligne intermédiaire montre le résultat de l'application d'un événement évolutif sur la séquence de la ligne précédente. — Page 87.

Locus : Un endroit précis du génome. — Page 14.

Marqueur génétique : C'est segment d'ADN pouvant présenter différentes formes d'un individu à l'autre. Ces marqueurs sont utilisés pour l'identification d'un individu ou d'une cellule porteuse, ou en tant que sonde pour le repérage d'un chromosome ou d'un locus. — Page 28.

Méiose : Ensemble de deux divisions précédées d'une seule synthèse d'ADN conduisant à quatre cellules haploïdes à partir d'une cellule mère diploïde. Lors de la première division, dite réductionnelle, les chromosomes homologues se séparent (ségrégation indépendante des chromosomes) tandis que lors de la seconde division, dite équationnelle, ce sont les chromatides de chaque chromosome qui se séparent. — Page 19.

Minisatellite : Structure associée aux régions télomériques et constituées de la répétition en tandem d'un motif élémentaire de longueur allant d'une dizaine à une centaine de nucléotides. *(synonyne(s) : VNTR).* — Page 17.

Mitose : Mécanisme de reproduction conforme d'une cellule mère en deux cellules filles au cours duquel apparaissent les chromosomes. — Page 18.

Mutation : Modification de la structure de la molécule d'ADN, donc du matériel génétique. Elle peut être due à un dysfonctionnement de la machinerie cellulaire lors de la fabrication de la molécule d'ADN (mutation dite spontanée) ou elle peut être induite par des agents chimiques, physiques ou biologiques dits mutagènes. — Page 20.

MVR-PCR : Technique PCR spécifique aux minisatellites. Cette méthode fournit la succession des variants d'un minisatellite sous forme de carte. VOIR AUSSI : PCR. — Page 30.

Nombre de stabilité : Le nombre de stabilité d'un graphe $\mathcal{G}$, noté $\alpha(\mathcal{G})$, est le cardinal d'un stable max de $\mathcal{G}$. — Page 52.

Nucléotide : Composants de base des molécules d'ADN formés de l'assemblage de molécules d'acide phosphorique, d'un sucre (ribose ou désoxyribose) et d'une base (A, T, C, G). — Page 11.

Ordre : L'ordre d'une amplification est le nombre de variant(s) copié(s). — Page 125.

PCR : Polymerase chain reaction. Réaction de polymérase en chaîne : technique permettant, par un phénomène d'amplification, de produire un grand nombre de copies d'un fragment d'ADN donné. — Page 29.

Phénotype : Ensemble des manifestations observables, visibles du génome (couleur des yeux, ou des cheveux, taille ...). Les caractères récessifs ne seront exprimés que s'ils sont sous forme homozygote. Les caractéristiques observables d'un individu, dépendant de son génotype et, souvent, de son environnement. — Page 14.

Pied d'une arche : Se rapporte aux arches d'ordre supérieur. Les pieds d'une arche correspondent soit au motif amplifié, soit au résultat de cette amplification. — Page 126.

Polymorphisme : Variations courantes de la séquence d'ADN entre les individus. — Page 15.

Préfixe : Une chaîne p est un préfixe de longueur m d'une chaîne s de longueur n, avec $m \leq n$, si $p = s[1..m]$. — Page 36.

Protéine : Macromolécule formée d'un enchaînement spécifique de très nombreux acides aminés (de quelques dizaines à quelques centaines). — Page 14.

Récessif : Le fait pour un allèle de ne s'exprimer dans le phénotype qu'à l'état homozygote. — Page 15.

Recombinaison : Échange physique de portions de chromosomes lors de la méiose entraînant un réarrangement des combinaisons génétiques. — Page 21.

Réparation : Ensemble des processus qui permettent la reconstitution d'un brin d'ADN à partir de structures présentant des anomalies diverses. — Page 13.

Réplication : Duplication à l'identique, propriété caractéristique de la molécule d'ADN. — Page 12.

RFLP : Restriction Fragments Length Polymorphism. Variation entre individus ou souches dans le profil d'ADN obtenu après coupure par diverses enzymes de restriction. Le polymorphisme de taille des fragments de restriction (RFLP) reflète directement des variations de séquence de segments précis d'ADN et est utilisé comme marqueur sur les cartes génétiques et physiques. — Page 29.

Séquençage : Détermination de la structure de base de l'ADN, c'est-à-dire de l'ordre précis dans lequel les nucléotides se succèdent dans l'ADN. — Page 28.

Séquences répétées : Séquences apparaissant plusieurs fois dans la molécule d'ADN côte à côte ou dispersées. — Page 16.

Simple : Une arche est dite simple si elle ne contient pas d'autres arches. Simple est le contraire de complexe. — Page 93.

SNP : Single Nucleotide Polymorphism. Les SNPs (prononcé snips) sont des mutations ponctuelles isolées, c'est-à-dire le polymorphisme d'un seul nucléotide. Ce sont des variations stables de la séquence d'ADN, portant sur une seule base, toutes les 100 à 300 bases environ du génome et affectant au moins 10% de la population. Beaucoup de SNPs n'ont pas d'implications fonctionnelles mais ils définissent un locus unique dans le génome et sont polymorphes. — Page 15.

Sous-chaîne : Une sous-chaîne s' de longueur m d'une chaîne s de longueur n, avec $m \leq n$, est une séquence de m symboles adjacents dans s. $s' = s[i]s[i+1]\dots[i+m-1]$ avec $1 \leq i \leq n-m+1$. *(synonyne(s) : facteur).* VOIR AUSSI : CHAÎNE. — Page 36.

Sous-séquence : Une sous-séquence w d'une chaîne s de longueur n, est une chaîne de longueur $1 \leq m \leq n$ construite en ôtant de s $n-m$ caractères. — Page 36.

Stable : Un stable d'un graphe $\mathcal{G}$ est un sous-ensemble de sommets de $\mathcal{G}$ qui n'a pas d'arêtes entre ses sommets. — Page 52.

Stable max : Un stable max d'un graphe $\mathcal{G}$ est un stable de $\mathcal{G}$ de cardinalité maximale, c'est-à-dire qu'il n'existe pas de stable de $\mathcal{G}$ ayant un plus grand nombre de sommets. — Page 53.

Substitution : Remplacement d'une base par une autre dans la molécule d'ADN conduisant à une mutation dans le génome. — Page 21.

Télomère : Extrémité d'un chromosome. — Page 10.

Traduction : Ensemble des mécanismes moléculaires conduisant à l'expression d'un ARNm en protéine. — Page 9.

Transcription : Ensemble des mécanismes moléculaires conduisant à la synthèse d'un ARNm à partir d'ADN. — Page 9.

Résumé

Les séquences répétées en tandem sont constituées de motifs adjacents. Elles constituent une classe de séquences génétiques dont font partie microsatellites et minisatellites. Dans cette thèse, nous traitons le problème de la comparaison de séquences répétées en tandem sous un modèle évolutif particulier. Plus précisément, nous nous intéressons au problème de leur alignement dans lequel, en plus des trois opérations classiques, mutation, insertion et délétion, nous considérons l'amplification en tandem et la contraction en tandem. L'amplification copie un facteur de la séquence, c'est-à-dire un ou plusieurs caractère(s), et met le ou les exemplaire(s) du facteur copié à côté du facteur original, la contraction est l'événement inverse. L'amplification (resp. la contraction) est dite « n-aire d'ordre m », si elle copie (resp. retire) m motif(s) n fois. Nous proposons une méthode donnant un score d'alignement, qui est une métrique, entre deux séquences répétées en tandem, sous un modèle comprenant les cinq opérations précédemment citées où l'amplification et la contraction sont unaires d'ordre 1. Le problème est difficile car les opérations ne sont pas commutatives. Notre solution fait appel à de l'algorithmique de graphe. Nous avons réalisé un programme nommé MS_Align qui implémente cette méthode. Il s'agit du premier programme capable d'aligner des cartes de minisatellites. À l'aide de ce programme, nous avons étudié des données biologiques provenant du minisatellite humain MSY1. Comme nous le montrons, notre modèle évolutif s'applique bien à ce type de séquences d'ADN. Nous avons construit à partir de nos résultats des arbres phylogénétiques semblables à ceux obtenus grâce à d'autres marqueurs du chromosome Y indépendants de MSY1, nos arbres offrent une meilleure résolution. Une partie de cette thèse est consacrée au problème général où nous relaxons les contraintes sur les amplifications et contractions.

mots-clefs : alignement, bioinformatique, graphe de chevauchement, minisatellite, séquence répétée en tandem.

Abstract

Tandem repeats consists of a heterogeneous tandem array of a repeat unit. Microsatellites and minisatellites belong to this class of genetic sequences. In this thesis, we deal with the problem of alignment of tandem repeats under a specific evolutionary model. This model, termed SSE, involves 5 operations : mutation, insertion and deletion (the classical operations in string alignment) plus tandem amplification and tandem contraction. An amplification duplicates a character to produce an identical character next to it. Contraction removes a character if and only if it is next to an identical character. The amplification (resp. contraction) is said "of arity n and order m" if it copies (resp. deletes) m motifs n times. We propose an algorithm to compute the distance between two tandem repeats under the SSE model where amplifications and contractions are of arity 1 and order 1. This problem is difficult because of the non-commutativity of operations. We have integrated our algorithm in a software named MS_Align. It is the first software to align minisatellites maps. We have study biological data from human minisatellite MSY1. Our evolutionary model well fit this kind of DNA sequences. We have construct phylogenetic trees similar to those constructed with other markers on Y chromosome. We observed that our tree shows a better resolution. A part of this thesis is devoted to the general problem obtained when relaxing the constraints on amplifications and contractions.

keywords : alignment, bioinformatics, minisatellite, overlap graph, tandem repeat.