UNIVERSITÉ DU QUÉBEC À MONTRÉAL

AU-DELÀ DE LA SIMILARITÉ DE SÉQUENCE : UTILISATION DE STRUCTURES D'ORDRE PARTIEL POUR L'ALIGNEMENT MULTIPLE DE GÉNOMES ENTIERS DE BACTÉRIOPHAGES

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
PAUL GUERTIN

DÉCEMBRE 2018

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

Avertissement

La diffusion de ce mémoire se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.07-2011). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

REMERCIEMENTS

En premier lieu, j'adresse de sincères remerciements à ma directrice de recherche, Anne Bergeron, pour m'avoir allumé à la bioinformatique en 2012. Merci aussi, Anne, pour toutes nos conversations, pour tes conseils, pour ton écoute.

Merci à Krister Swenson, codeur extraordinaire et principal développeur du programme Alpha, sans qui ce travail n'existerait pas.

Merci aux autres membres du groupe de recherche et coauteurs des articles qui sont à l'origine de ce travail : Sèverine Bérard, Annie Chateau, Hugo Deschênes et Nicolas Pompidor, ainsi qu'à tous les membres du Lacim pour leur accueil et les bons moments passés ensemble.

Merci au fonds de perfectionnement du Collège André-Grasset de m'avoir encouragé — moralement et financièrement — à renouer avec la recherche.

Finalement, un merci tout spécial à Kaimei et Laurent pour m'avoir toujours encouragé dans ce projet et s'être montrés compréhensifs les jours où les bactériophages accaparaient mon attention.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

RÉSUMÉ

Ce mémoire de bioinformatique vise à utiliser des structures mathématiques et des logiciels informatiques afin d'étudier le processus de recombinaison modulaire chez les bactériophages. On y décrit une méthode basée sur une structure d'ordre partiel pour construire des graphes qui permettent d'obtenir un alignement multiple d'ensembles de génomes de bactériophages, ainsi qu'un logiciel interactif qui permet de visualiser les graphes d'alignement.

Mots-clés : bactériophages, génomique comparative, alignement de génomes entiers, ordre partiel.

INTRODUCTION

Dans la dernière décennie du 20e siècle, les méthodes manuelles de séquençage génomique ont cédé la place à des méthodes automatiques dont la vitesse et l'exactitude n'ont cessé de croître. Un exemple suffira pour illustrer l'ampleur de cette évolution : le séquençage du premier génome humain, achevé en 2003, fut un projet multinational de 15 ans, avec un budget de trois milliards de dollars. En comparaison, séquencer un génome humain de nos jours coûte environ 70 dollars. Le nombre d'organismes dont le génome est entièrement séquencé et disponible dans une banque de données en ligne est en croissance exponentielle depuis le début du 21e siècle.

Cette explosion de données est une véritable révolution en biologie, une révolution dont les conséquences sont encore difficiles à entrevoir. Comment enregistrer, classifier et rendre disponibles toutes ces données ? Et surtout, comment les interpréter ? Pour avoir espoir de transformer ces données en information utile, il faut disposer d'outils automatisés qui permettent de donner un sens biologique à ces pétaoctets de séquences nucléotidiques.

Nous disposons de la séquence génomique de plusieurs mammifères, plantes et bactéries. Pour de tels organismes, de nombreuses approches basées sur l'identité de séquence permettent de faire du transfert d'annotation et d'affirmer — lorsque les portions identiques sont assez longues pour que leur similarité ne soit pas due au hasard — que deux séquences provenant du génome de deux organismes fort différents sont « la même », c'est-à-dire qu'elles proviennent d'un ancêtre commun.

Les bactériophages sont abondants mais leur variabilité est encore mal connue.

Chaque nouveau bactériophage séquencé amène de la nouveauté, sous la forme de séquences qui ne figurent nulle part dans les banques de données. À cause de la nature particulière du processus d'évolution chez les bactériophages, la nature des problèmes mathématiques et informatiques posés par l'interprétation de leurs séquences génomiques est différente : les méthodes efficaces avec les génomes de mammifères ou de bactéries ne sont que de peu d'utilité pour traiter les génomes de bactériophages.

Le problème spécifique dont il sera question dans ce mémoire est donc le suivant : développer une approche automatisée ou semi-automatisée permettant d'obtenir des alignements multiples de génomes de bactériophages qui soient biologiquement signifiants.

Le premier chapitre de ce mémoire décrit brièvement la structure des génomes de bactériophages et explique la nécessité d'une nouvelle approche pour comparer leurs séquences génomiques.

Dans le chapitre 2, on décrit une approche utilisant la notion mathématique d'ordre partiel permettant d'obtenir, à partir uniquement de la séquence génomique d'un ensemble de bactériophages, un alignement multiple possédant un sens biologique.

Le chapitre 3 explore les limites de cette approche et identifie les caractéristiques particulières des génomes de phages qui font qu'elle se prête bien à leur étude.

Dans le chapitre 4 sont donnés des détails sur les algorithmes permettant d'implanter la méthode décrite au chapitre 2.

Les algorithmes décrits au chapitre 4 ont été utilisés pour écrire un outil interactif d'alignement appelé Alpha. Le chapitre 5 montre des graphes d'alignement obtenus par Alpha à partir de données génomiques réelles et illustre comment cet

outil permet de comparer les génomes d'un ensemble de bactériophages.

Les résultats présentés dans ce mémoire ont fait l'objet de publications (Swenson *et al.*, 2013; Bérard *et al.*, 2016) que le lecteur est invité à consulter pour de plus amples informations.

En tant que membre de l'équipe de recherche, j'ai participé à l'ensemble des discussions concernant ce projet. Mes contributions spécifiques incluent notamment :

- les résultats présentés au chapitre 3 concernant la colinarité fonctionnelle, ainsi que les contre-exemples qui y sont donnés ;
- la conception et l'implantation d'un algorithme efficace, présenté au chapitre 4, pour la construction du graphe des colonnes ;
- la présentation du projet à une équipe de recherche du LIFL (Laboratoire d'informatique fondamentale de Lille).

CHAPITRE I

NOTIONS BIOLOGIQUES SUR LES PHAGES

1.1 Qu'est-ce qu'un bactériophage ?

Les bactériophages — littéralement : « mangeurs de bactéries »—, dits aussi « phages », sont des virus qui infectent les bactéries. Ils sont considérés comme les entités biologiques les plus abondantes sur la planète, avec une population estimée à 10^{31} phages. Cette estimation provient de deux études : l'une (Whitman *et al.*, 1998) estime le nombre de bactéries à 10^{30}, et l'autre (Weinbauer, 2004) estime que chaque bactérie est hôte de 10 bactériophages en moyenne, ce que l'article appelle le VBR ou *virus-to-bacteria ratio*. Les bactériophages sont présents dans l'ensemble de la biosphère, particulièrement dans les océans et la terre arable, et ont été détectés jusque dans des habitats extrêmes comme les sources d'eau chaude volcaniques (Ackermann, 2011).

Malgré cette abondance, ce n'est que depuis environ 100 ans qu'on connaît leur existence. En 1915, l'Anglais Frederick Twort a remarqué la présence de zones exemptes d'un certain type de bactérie dans ses cultures et a spéculé sur la raison de ce fait (Twort, 1915). En 1917, le Canadien Félix d'Hérelle a pris position (d'Herelle *et al.*, 1917) en faveur de l'existence de virus ultra-microscopiques s'attaquant aux bactéries, et qu'il a baptisés *bactériophages*. Des expériences subséquentes lui ont donné raison.

D'Hérelle s'intéressait aux phages surtout pour leurs propriétés thérapeutiques et il est considéré comme le fondateur de ce champ d'étude, appelé *phagothérapie.* La découverte des antibiotiques en 1941, moins spécifiques que les phages mais plus faciles à manipuler (Kutter *et al.*, 2010), a beaucoup diminué l'intérêt pour la phagothérapie sauf dans les pays de l'Est, qui n'avaient pas un accès facile aux antibiotiques durant la seconde Guerre mondiale et la guerre froide. De nos jours, la multiplication des souches de bactéries résistantes aux antibiotiques et les avancées en édition génomique ont renouvelé l'intérêt pour l'utilisation des phages à des fins thérapeutiques (Cox *et al.*, 2015).

L'organisme chargé de la taxonomie des bactériophages est le *International Committee of Taxonomy of Viruses*, ou ICTV, qui les classe en six ordres taxonomiques selon leur forme. Plus de 96 % des phages font partie de l'ordre *Caudovirales*, caractérisé par une morphologie à queue. La taxonomie se divise ensuite en quelque 87 familles et 19 sous-familles, sur la base de la morphologie et des propriétés physico-chimiques mais aussi, de plus en plus, sur la base de données génomiques (Ackermann, 2011). La figure 1.1 montre une représentation schématique d'un bactériophage de l'ordre *Caudovirales.* On peut distinguer, de haut en bas, la tête ou capside renfermant l'ADN viral, la longue queue verticale, et les pattes qui permettent au bactériophage d'agripper la cellule dans laquelle il introduira son ADN. Lors de l'introduction de l'ADN, la partie inférieure de la queue pénètre à l'intérieur de la cellule et l'ADN viral est transféré de la capside à la cellule en passant par l'intérieur de la queue.

1.2 Structure et théorie modulaire

Vu leur petite taille, les génomes de phages ont été rapidement ciblés par des projets de séquençage de génomes complets. En fait, le premier génome complet a avoir été séquencé, en 1977, est celui d'un bactériophage de 5375 nucléotides

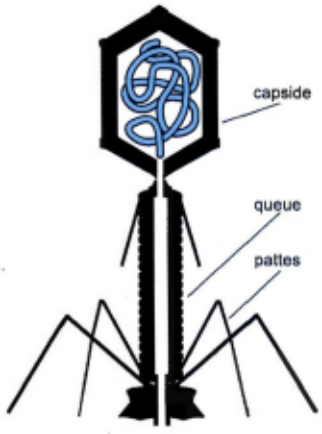

Figure 1.1 Représentation schématique d'un bactériophage (source : Wikimedia, licence Creative Commons)

(Sanger *et al.*, 1977). Depuis, le nombre de phages ayant eu leur génome séquencé au complet ou en partie n'a cessé de croître à un rythme de plus en plus rapide. En date d'avril 2018, la banque de données *PhagesDB* (Russell et Hatfull, 2016) répertorie 12 992 bactériophages, parmi lesquels 2534 ont leur génome complet séquencé. Des centaines de génomes sont ajoutés à cette banque de données chaque année. Cette abondance de données génomiques et la compacité du génome ont permis d'étudier de près la structure génomique des phages et de prendre conscience qu'elle présente des particularités dont nous allons maintenant discuter.

Chez tous les êtres vivants, y compris les phages, les mutations génétiques ponctuelles — en anglais, « single nucleotide polymorphisms » ou SNPs — sont des facteurs prépondérants de la variabilité génétique. Certaines espèces, comme les humains, utilisent aussi la recombinaison homologue pour échanger du matériel génétique entre des séquences d'ADN semblables situées soit sur deux molécules différentes ou sur la même molécule. Ici, le terme *semblables* veut dire que les

séquences en question contiennent la même séquence génomique, sauf en certains points isolés où se trouvent les mutations ponctuelles. Les modèles biologiques du phénomène de recombinaison sont bien étudiés et la recombinaison homologue de segments porteurs de mutations différentes est à l'origine de la variété génétique. La similarité de séquence étant habituellement corrélée avec la similarité fonctionnelle, elle est très utilisée en génétique des populations. La majorité des algorithmes utilisés pour comparer des organismes et reconstruire leur histoire évolutive utilisent la similarité de séquence comme un outil de base (Kececioglu et Gusfield, 1998).

L'étude du génome des phages montre une tout autre réalité. Lorsque plusieurs phages d'espèces différentes infectent la même bactérie, ils peuvent s'échanger du matériel génétique et ainsi former des nouveaux phages. Lorsqu'on compare ces nouveaux phages à leurs « parents », on constate que des séquences semblables alternent avec des séquences sans aucune similarité. Cette découverte a donné lieu, il y a une quarantaine d'années, à la théorie modulaire (Botstein, 1980) de la structure des phages. Selon cette théorie, le génome d'un phage est un assemblage de modules, chacun ayant une fonction biologique précise. Par exemple, le génome des phages *siphovirus* contient les modules suivants.

1. Lysogénie.
2. Métabolisme de l'ADN.
3. Morphologie de la tête.
4. Morphologie de la queue.
5. Lyse.

Chacun de ces modules a une fonction biologique précise. Ainsi, le premier module concerne l'intégration du génome du phage dans l'ADN circulaire de la bactérie au moyen d'un gène appelé *intégrase*. L'important pour un module est la fonction biologique et non la séquence génomique. Ainsi, deux séquences avec très peu

de similarité peuvent accomplir la même fonction biologique, de la même façon qu'on peut utiliser des outils fort différents pour obtenir un même résultat. Il existe donc pour chaque module un certain nombre de variantes, qui sont les différentes implémentations de celui-ci. Pour faire une analogie, le module *propulsion* d'un bateau peut être implémenté par la variante *voile*, la variante *rames*, la variante *moteur à essence*, etc. qui sont toutes des façons légitimes d'assurer la propulsion du bateau, même si elles ne partagent guère de similarité en ce qui concerne leur apparence ou leur principe de fonctionnement.

Pour le module *lysogénie* dont il a été question ci-haut, les différentes variantes correspondent à différents endroits dans le génome bactérien où le phage peut intégrer son ADN. Pour la bactérie *Staphylococcus aureus*, on connaît une dizaine de sites d'intégration sur le génome, chacun avec une variante du module *lysogénie* correspondant (Kahánková *et al.*, 2010). Bien que toutes les variantes du module accomplissent la même tâche, leur séquence génomique est spécifique à la séquence bactérienne à l'endroit d'insertion, et les variantes diffèrent entre elles au point où aucun alignement nucléotidique significatif n'est possible entre deux variantes.

La division du génome des phages en modules peut se faire à différents niveaux de finesse. Par exemple, le module *lysogénie* peut lui-même être divisé en sous-modules, parmi lesquels figure le gène codant pour la protéine *intégrase*. Tous les phages possèdent évidemment une variante de chacun des cinq modules ci-dessus, car l'absence de l'un d'entre eux rendrait le phage non viable. Par contre, si on définit les modules plus finement, alors il arrive que certains modules soient absents du génome de certains phages. Par contre, lorsque des modules sont présents, ils le sont toujours dans le même ordre. Toutefois, puisque plusieurs copies du génome d'un phage sont concaténées les uns aux autres durant l'infection d'une bactérie, il arrive que des recombinaisons se produisent entre la fin d'une copie du génome et le début de la copie suivante. C'est pourquoi, dans l'étude des recombinaisons,

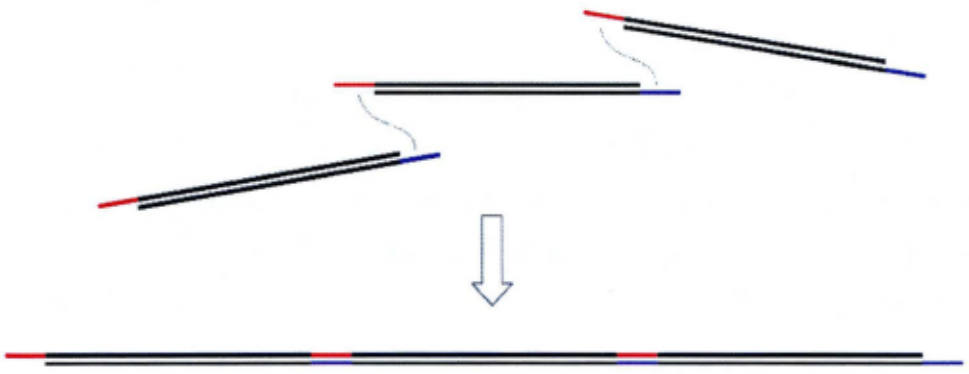

Figure 1.2 Concaténation de trois copies d'un génome de phage

il est nécessaire de faire l'hypothèse que les génomes de phages sont circulaires. L'ordre de modules doit donc toujours être compris à une permutation circulaire près.

La raison qui explique que de multiples copies du génome du phage sont concaténées dans la bactérie hôte est que l'ADN introduit dans la bactérie hôte est à double brin, mais avec deux extrémités à simple brin appelés *extrémités cohésives*. Ces extrémités sont complémentaires, de sorte qu'elles auront tendance à former des liens entre elles et ainsi former une longue molécule composée de plusieurs copies du génome. Ce processus a été d'abord observé avec le bactériophage λ (Hershey *et al.*, 1963) et ensuite généralisé à d'autres familles de bactériophages (Kaiser et Wu, 1968).

La figure 1.2 illustre ce processus. On y voit trois copies du génome viral comportant des extrémités cohésives en couleur (rouge et bleu étant complémentaires l'une à l'autre) qui s'apparient et forment une longue molécule formée de trois copies concaténées du génome viral. Les deux extrémités de la longue molécule peuvent aussi s'apparier de manière à former un génome circulaire, mais les détails biologiques ne sont pas pertinents pour notre modèle.

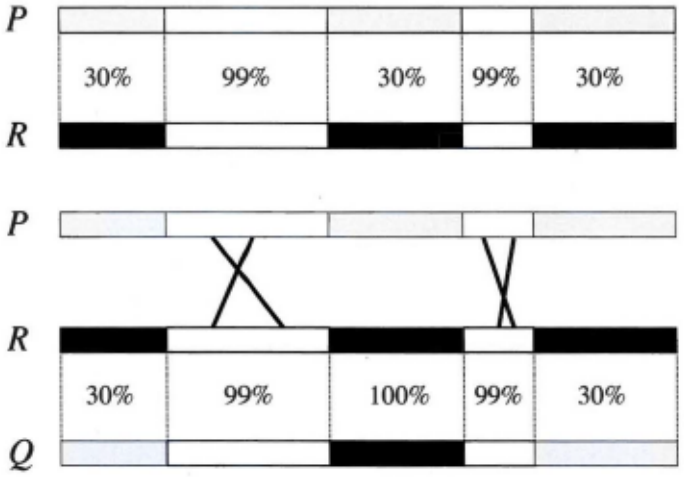

Figure 1.3 En haut, comparaison des génomes des phages P et R montrant un motif de régions fortement dissimilaires qui alternent avec des régions fortement similaires. En bas, un événement de recombinaison : les phages P et R utilisent leurs régions similaires pour produire le phage Q. Les pourcentages correspondent au pourcentage d'identité entre les régions.

1.3 Recombinaison modulaire

Les mécanismes de la recombinaison ne font pas l'unanimité chez les chercheurs. Une hypothèse de longue date (Weisberg et Adhya, 1977) est que les phages se recombinent de manière illégitime, c'est-à-dire à des endroits choisis aléatoirement dans le génome, et que les organismes non viables ainsi obtenus sont éliminés par sélection naturelle, laissant ceux que le hasard a faits viables. Cette hypothèse est toujours considérée par les chercheurs (Krupovic *et al.*, 2011), mais une deuxième voie est également explorée. Martinsohn (Martinsohn *et al.*, 2008) propose plutôt que de courtes séquences homologues flanquant les modules jouent un rôle d'ancres et permettent l'échange des séquences divergentes qu'elles encadrent, comme illustré à la figure 1.3.

Cette théorie a été vérifiée *in silico* (Swenson *et al.*, 2013) par l'étude de 31 génomes de phages infectant la bactérie *Staphylococcus aureus* dans le but de les aligner et d'identifier des modules et leurs variantes. Une approche semi-automatique a permis d'aligner les 31 génomes sur près de 80 % de leur longueur. Lors de cet alignement, de courts segments, mesurant autour de 50 nucléotides chacun, ont été identifiés entre les modules. Ces segments présentent un haut taux de conservation parmi tous ou presque tous les génomes étudiés, et apparaissent à intervalles réguliers. Ces segments ont été baptisés *ancres* et sont à la base des techniques d'alignement présentées dans (Swenson *et al.*, 2013) et (Bérard *et al.*, 2016).

CHAPITRE II

ALIGNEMENT DE GÉNOMES DE PHAGES À L'AIDE DE GRAPHES

Étant donné un ensemble de génomes de phages, notre but est de construire *ab initio* (c'est-à-dire à partir de la séquence génomique seulement, sans utiliser d'information supplémentaire telle que des annotations) un alignement multiple de ces génomes qui soit biologiquement signifiant. Le terme «biologiquement signifiant» veut dire ici que l'objectif est d'aligner des séquences génomiques lorsqu'elles correspondent à une fonction biologique semblable, même si elles ne présentent aucune similarité de séquence. L'alignement ainsi obtenu pourra être utilisé, par exemple, pour faire du transfert d'annotation. Les outils classiques d'alignement étant fondés sur la similarité de séquence, ils ne peuvent pas, à eux seuls, résoudre ce problème.

Pour atteindre notre but, nous utilisons trois particularités des génomes de phages :

1. La colinéarité fonctionnelle : à l'intérieur d'une même famille de phages, les fonctions biologiques sont présentées dans un ordre qui est le même d'un phage à l'autre (à une permutation circulaire près).
2. La pression d'efficacité sur le génome des phages, qui fait en sorte que pratiquement toute la séquence est codante et que les répétitions de segments sont rares chez les phages.
3. La présence de longues séquences identiques partagées par des sous-ensembles

de phages d'une même famille, et qui serviront d'«ancres locales» pour situer les séquences uniques entre celles-ci et inférer leur fonction à partir du principe de colinéarité.

La construction procède en trois étapes et utilise les graphes et multigraphes orientés comme structure de données servant de support à l'alignement. L'utilisation de graphes dans la construction d'alignements multiples de génomes entiers est une technique éprouvée (Kehr *et al.*, 2014) qui possède de nombreuses variantes. Dans la plupart des cas, la dernière étape consiste à « aplatir »le graphe de manière à obtenir un alignement unidimensionnel entre deux séquences, mais comme nous allons le voir, la structure modulaire des génomes de bactériophages fait qu'il y a un intérêt à exprimer leur alignement au moyen d'un graphe.

Les trois étapes de notre construction sont les suivantes. À la fin de chaque étape, on obtient un graphe ou un multigraphe orienté.

1. Dans un premier temps, nous construisons un multigraphe d'alignement multiple appelé «graphe des colonnes», basé sur des appariements de paires de génomes. Dans le graphe des colonnes, chaque sommet représente une colonne de l'alignement.
2. Dans un deuxième temps, les sommets successifs du graphe des colonnes qui couvrent le même ensemble de génomes sont fusionnés de manière à créer le «graphe d'alignement», qui est un multigraphe. Dans le graphe d'alignement, chaque sommet représente un alignement exact et maximal.
3. Finalement, le graphe d'alignement est contracté de manière à diminuer le nombre de sommets et augmenter la proportion de sommets qui représentent des alignements biologiquement signifiants, et les arêtes multiples entre deux sommets sont fusionnées de manière à obtenir un graphe et non un multigraphe. Les sommets du graphe contracté représentent des alignements sans trous mais pas forcément exacts.

Nous allons maintenant préciser les trois étapes de la construction et les illustrer avec des petits exemples, puis nous montrerons un exemple plus réaliste de graphe d'alignement obtenu à partir de données réelles sur six génomes de bactériophages.

2.1 Des appariements au graphe des colonnes

Soit $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ un ensemble de génomes de phages. Nous allons noter $G_i[s]$ le nucléotide en position s de G_i, et $G_i[s..t]$ la séquence formée des nucléotides de la position s à la position t de G_i. Les positions sont numérotées à partir de 1. Par exemple, si $G_1 = \texttt{ATTGCA}$ et $G_2 = \texttt{TGCCA}$, on a $G_1[1] = G_2[5] = \texttt{A}$ et $G_1[3..5] = G_2[1..3] = \texttt{TGC}$.

Notre outil de base est très simple : il s'agit de l'identité de séquence. Lorsque deux génomes différents G_i et G_j sont tels que $G_i[s..t] = G_j[u..v]$ (ce qui implique que $t - s = u - v$), on dit que nous avons un *appariement* $A = \{G_i[s..t], G_j[u..v]\}$ de longueur $t - s + 1$. Un appariement affirme que les deux séquences sont égales et véhicule l'idée que les deux séquences étaient auparavant «la même», autrement dit qu'elles correspondent à une même séquence dans le génome d'un ancêtre commun. Un appariement entre deux séquences correspond à un alignement trivial entre celles-ci. Par exemple, Pour $G_1 = \texttt{ATTGCA}$ et $G_2 = \texttt{TGCCA}$, l'alignement présenté à la figure 2.1 correspond à l'appariement $A = \{G_1[3..5], G_2[1..3]\}$.

```
A T T G C A
    T G C C A
```

Figure 2.1 Un alignement entre deux séquences

Quand on a un ensemble d'appariements couvrant plusieurs génomes, chaque appariement permet d'établir un alignement entre des positions de ces génomes, et par transitivité on peut en déduire un alignement entre d'autres positions, même si aucun appariement individuel n'implique ces positions.

Exemple 1. *Considérons les trois génomes*

$$G_1 = \texttt{ATTGCA}, \qquad G_2 = \texttt{TGCCA}, \qquad G_3 = \texttt{CAGGCA}$$

et les deux appariements

$$A_1 = \{G_1[3..5], G_2[1..3]\}, \qquad A_2 = \{G_2[2..5], G_3[3..6]\}.$$

Comme le montre la figure 2.2, ces deux appariements permettent ensemble d'établir l'égalité entre les deux séquences GC *en grisé dans G_1 et G_3, même si aucun appariement n'implique G_1 et G_3.*

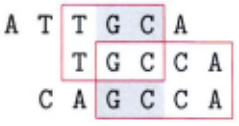

Figure 2.2 Un alignement entre trois séquences

Un ensemble d'appariements permet donc de regrouper les positions de tous les génomes étudiés en *classes d'équivalence*. Deux positions sont dans une même classe s'il existe un appariement, ou un ensemble d'appariements, qui permet d'en établir l'égalité (autrement dit, qui permet de les aligner). D'un point de vue biologique, une classe d'équivalence correspond à une *position* dans le génome ancestral d'où proviennent les génomes étudiés : deux positions qui sont dans la même classe sont réputées provenir de la même position du génome ancestral.

Exemple 1 (suite). *La figure 2.3 montre comment on peut former les classes d'équivalence avec avec les positions des trois génomes G_1, G_2 et G_3, à partir des deux appariements A_1 et A_2 donnés plus haut. Dans la figure 2.3, chaque encadré correspond à une classe d'équivalence.*

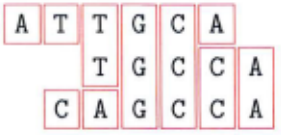

Figure 2.3 Classes d'équivalence formées à partir de deux appariements

Nous noterons p_{ij} la position j du génome G_i et $[\![p_{ij}]\!]$ la classe d'équivalence de cette position. Nous appellerons *colonnes* ces classes d'équivalence, puisqu'elles constitueront les sommets du graphe des colonnes et correspondront aux colonnes de l'alignement multiple que nous construisons. Le support $\mathrm{sp}\,[\![p_{ij}]\!]$ d'une colonne est l'ensemble des génomes qui ont une position dans $[\![p_{ij}]\!]$.

Exemple 1 (suite). *Selon la notation que nous venons de définir, les dix colonnes de cet exemple sont*

$$\{p_{11}\},\ \{p_{12}\},\ \{p_{31}\},\ \{p_{13}, p_{21}\},\ \{p_{32}\},\ \{p_{14}, p_{22}, p_{33}\},\ \{p_{15}, p_{23}, p_{34}\},$$
$$\{p_{16}\},\ \{p_{24}, p_{35}\},\ \{p_{25}, p_{36}\}.$$

Le graphe des colonnes est formé à partir de l'ensemble des colonnes (c'est-à-dire des classes d'équivalence) de la manière suivante. Les sommets du graphe sont les colonnes, et il y a une arête entre les colonnes $[\![p_{ij}]\!]$ et $[\![p_{kl}]\!]$ chaque fois qu'un certain génome G_g a une position p_{gs} dans $[\![p_{ij}]\!]$ et la position suivante $p_{g(s+1)}$ dans $[\![p_{kl}]\!]$. Puisque tous les nucléotides d'un génome, sauf le dernier, ont un successeur, le nombre d'arêtes dans le graphe des colonnes est donc égal à la taille totale de l'ensemble des génomes, moins le nombre de génomes.

Exemple 1 (suite). *Le graphe des colonnes correspondant à notre exemple est montré à la figure 2.4.*

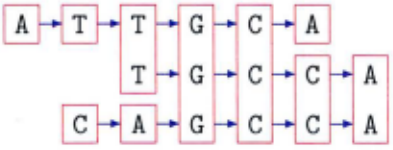

Figure 2.4 Graphe des colonnes pour trois génomes

2.2 Du graphe des colonnes au graphe d'alignement

Une fois le graphe des colonnes établi, la deuxième étape consiste à fusionner certains de ses sommets pour obtenir le graphe d'alignement, qui est plus compact.

Lorsque deux sommets adjacents dans le graphe des colonnes ont le même support, cela veut dire qu'ils alignent des positions successives dans les mêmes génomes. On peut donc les fusionner. Le graphe obtenu en effectuant toutes les fusions de sommets adjacents de même support dans le graphe des colonnes s'appelle le graphe d'alignement. C'est un multigraphe dont chaque sommet représente un alignement exact et maximal impliquant un sous-ensemble des génomes.

Exemple 1 (suite). *Le graphe d'alignement de notre exemple comporte 6 sommets et 7 arêtes. Il est représenté à la figure 2.5.*

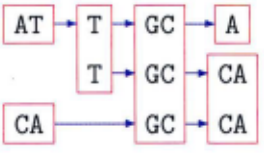

Figure 2.5 Graphe d'alignement pour trois génomes

2.3 Du graphe d'alignement au graphe contracté

À partir du graphe d'alignement, on obtient le graphe contracté en effectuant deux opérations de fusion : on fusionne d'abord des sommets pour obtenir un

graphe intermédiaire, puis on fusionne des arêtes dans ce graphe intermédiaire pour obtenir le graphe contracté.

La fusion des arêtes consiste à transformer le multigraphe en graphe en fusionnant toutes les arêtes qui vont d'un sommet donné à un autre.

Exemple 1 (suite et fin). *En fusionnant les arêtes du graphe d'alignement de notre exemple, on obtient le graphe d'alignement montré à la figure 2.6.*

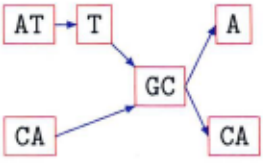

Figure 2.6 Graphe contracté pour trois génomes

Avant cette étape, on procède à la fusion des sommets. La fusion des sommets simplifie le graphe et aide à donner un sens biologique à un maximum de sommets du graphe. En effet, jusqu'à maintenant, les sommets sont formés d'alignements exacts, c'est-à-dire à 100 % d'identité. En relaxant cette contrainte de manière à permettre des alignements avec un peu moins de 100 % d'identité, mais sans insertion ni délétion, on permet à des séquences qui diffèrent seulement par quelques mutations ponctuelles d'être représentées par le même sommet. La raison de cette opération est qu'il se peut que des séquences génomiques diffèrent en certaines positions suite à des mutations ponctuelles, mais qu'elles partagent une fonction biologique. Dans le graphe des colonnes, ces séquences seront représentées par plusieurs sommets. Il est donc souhaitable de fusionner ces derniers, même si l'alignement n'est plus exact.

Tableau 2.1 Portions du génome de six bactériophages de *S. aureus*

Nom	Numéro d'accession Genbank	Positions dans le génome	Longueur
Bactériophage 85	AY954953.1	27207–30036	2830
Bactériophage 92	AY954967.1	25124–27683	2560
Bactériophage 53	AY954952.1	26006–29079	3074
Bactériophage 29	AY954964.1	25241–28577	3337
Bactériophage 88	AY954966.1	25193–27946	2754
Bactériophage 187	AY954950.1	21462–22997	1536

2.4 Un alignement impliquant six génomes

Pour illustrer un graphe d'alignement construit à partir des séquences génomiques réelles, nous avons considéré six génomes de bactériophages de *Staphylococcus aureus*. Nous en avons extrait la séquence commençant au gène codant pour la protéine *intégrase* et s'arrêtant après le gène codant pour la protéine *invertase*. Ces deux gènes ont été choisis parce qu'ils sont communs aux six génomes et ils sont bien annotés dans la banque de données Genbank. Grâce aux annotations de Genbank, nous avons eu la confirmation que les génomes sont colinéaires : les protéines codées par chacun des génomes sont dans le même ordre pour les six. Les noms des bactériophages, les numéros d'accession et les positions exactes dans le génome sont données dans le tableau 2.1.

En analysant ces séquences nucléotidiques à l'aide de l'outil de comparaison de séquence *Blast* (Altschul *et al.*, 1990), on peut constater que certains bactériophages ont de longues séquences nucléotidiques en commun, suivies de régions où il y n'a pas d'homologie détectable. La structure en mosaïque du génome des phages, où des régions presque identiques alternent avec des régions fortement dissimilaires, est bien apparente.

La figure 2.7 montre un exemple d'alignement des six séquences construit par *Blast*. Chaque ligne colorée représente un bactériophage, dans le même ordre que le tableau 2.1. Les portions des lignes qui sont en couleur représentent des alignements exacts ou presque exacts entre le bactériophage 85 (la ligne du haut) et les autres bactériophages, tandis que les portions des lignes qui sont minces et en noir représentent des régions sans aucune homologie.

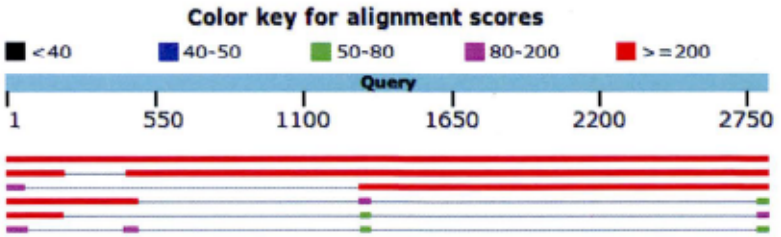

Figure 2.7 Alignement de six génomes de bactériophages avec *Blast*

Au vu des alignements construits par *Blast*, on peut diviser chaque séquence génomique en régions d'au moins 50 nucléotides qu'on identifiera par des lettres. La présence d'une même lettre dans plusieurs séquences génomiques indique que les régions en question s'alignent avec près de 100 % d'identité. Par contre, deux lettres différentes ne présente aucune homologie détectable. Ainsi, chaque séquence génomique devient une séquence de variantes comme indiqué dans le tableau 2.2.

Tableau 2.2 Séquences de variantes extraites de bactériophages de *S. aureus*

Nom	Séquence de variantes
Bactériophage 85	ABCDEFGN
Bactériophage 92	ABDEFGN
Bactériophage 53	AHDMFGN
Bactériophage 29	ABCLFJN
Bactériophage 88	ABIFJN
Bactériophage 187	AKFJN

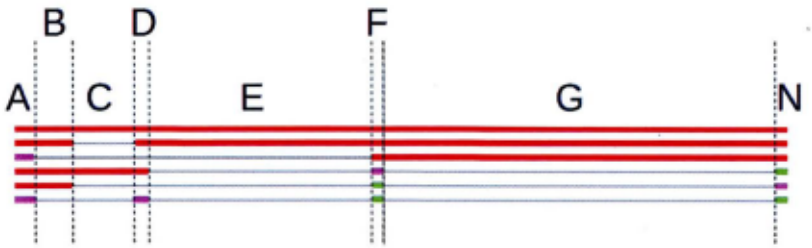

Figure 2.8 Alignement de six génomes de bactériophage avec *Blast*, avec les identifications de certaines variantes

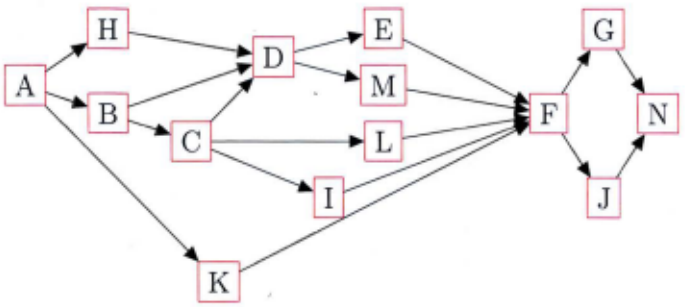

Figure 2.9 Le graphe d'alignement de six séquences extraites de bactériophages de *S. aureus*

Dans l'alignement de la figure 2.7, on peut distinguer certaines de ces variantes, comme indiqué à la figure 2.8. Pour les parties encore non identifiées, on peut utiliser à nouveau *Blast* pour aligner un autre bactériophage que 85 avec tous les autres, et ainsi décomposer tous les génomes en variantes, petit à petit.

Une fois les génomes représentés comme des suites de variantes, on peut appliquant les procédures décrites dans les sections 2.1 à 2.3 pour construire le graphe des colonnes et le graphe d'alignement. On trouve que le graphe d'alignement contracté correspondant à ces séquences est celui présenté à la figure 2.9.

Cette méthode de construction du graphe d'alignement présente de similitudes

avec l'aligneur *Threaded Blockset Aligner* (Blanchette *et al.*, 2004), utilisé entre autres à l'étape de reconstruction des régions synténiques dans un contexte de reconstruction d'un génome ancestral (Diallo *et al.*, 2009). Il ne s'agit pas d'alignement de génomes de bactériophages, mais les idées de ce chapitre s'appliquent dès lors qu'on peut garantir la colinéarité.

2.5 Identification des ancres et constats initiaux

Notre but est d'obtenir un alignement. En d'autres termes, on souhaite identifier des variantes et dire : « ces variantes font partie d'un même module ». Le graphe d'alignement nous permet déjà de poser un certain nombre de constats concernant l'alignement des six séquences, et de prédire que certaines variantes appartiennent à un même module. On remarque d'abord que toutes les six séquences commencent par la variante A, se terminent par la variante N, et contiennent la variante F quelque part entre les deux. Ces variantes A, F, N sont donc des ancres.

Entre les ancres F et N, chaque génome contient exactement une variante parmi G et J. Ces variantes sont donc identifiées comme des variantes d'un même module, même si elles ne présentent aucune similarité de séquence. Nous avons donc identifié quatre modules jusqu'à maintenant. Les voici avec leurs variantes. Comme nous ne savons pas encore le nombre de modules, on note n le dernier module.

- Module 1 : A
- Module $n-2$: F
- Module $n-1$: G, J
- Module n : N

Considérons maintenant la variante D. Elle n'est présente que dans quatre des six génomes, donc on l'appellera une ancre locale. Lorsqu'elle est présente dans un génome, alors il y a exactement une variante parmi E et M qui est entre l'ancre locale D et l'ancre F. Cela nous permet d'identifier E et M comme variantes d'un

même module, et D comme une variante du module précédent. On peut donc poursuivre :

— Module $n-3$: E, M

— Module $n-4$: D

Si on veut poursuivre ce raisonnement, cela devient plus compliqué. Par exemple, les variantes B et C précèdent toutes deux la variante D. Doit-on donc les placer dans un même module ? Non, puisque C est aussi un successeur de B. Et que dire de la variante L ? Bien que le graphe nous incite à la placer dans le module $n-3$ avec les variantes E et M, il se pourrait que les génomes où L est présente n'aient aucune variante du module $n-3$ et que L soit une variante d'un module précédent.

Pour clarifier les choses et éviter de devoir se fier à un raisonnement *ad hoc* à chaque fois, on peut étudier les propriétés combinatoires de l'ordre partiel représenté par le graphe d'alignement. On aboutira ainsi à une procédure qui permet de regrouper les variantes en modules.

2.6 Du graphe d'alignement à l'alignement

Nous utiliserons la notion d'antichaîne pour nous aider à définir les modules et à y placer les variantes.

Définition 1. *Une* antichaîne *d'un ensemble partiellement ordonné $\mathcal{V}$ est un sous-ensemble de $\mathcal{V}$ tel que deux éléments quelconques de ce sous-ensemble ne sont jamais comparables.*

S'il y a des antichaînes, c'est qu'il y a des chaînes. Nous utiliserons ce concept plus tard, alors aussi bien le définir dès maintenant.

Définition 2. *Une* chaîne *d'un ensemble partiellement ordonné $\mathcal{V}$ est un sous-ensemble de $\mathcal{V}$ tel que deux éléments quelconques de ce sous-ensemble sont toujours*

comparables.

Dans le graphe d'alignement, une chaîne est un ensemble de sommets ayant la propriété qu'il existe un chemin dans le graphe qui passe par tous les sommets de la chaîne. Cet ensemble est totalement ordonné et possède donc un élément maximal (celui qui est « au bout » de la chaîne).

Nous utiliserons le graphe d'alignement pour partitionner l'ensemble de ses sommets en antichaînes.

Définition 3. *Soit $\mathcal{V}$ un ensemble partiellement ordonné. Nous dirons que M_1, M_2, $\ldots, M_k$ forme une* partition en antichaînes compatible *de $\mathcal{V}$ si les quatre conditions suivantes sont satisfaites :*

- *chaque M_i est une antichaîne,*
- $M_1 \cup M_2 \cup \cdots \cup M_k = \mathcal{V}$,
- $M_i \cap M_j = \emptyset$ *si* $i \neq j$,
- *Pour tous* $x, y \in \mathcal{V}$, $x < y$ *et* $y \in M_i$ *implique* $x \in M_1 \cup M_2 \cup \cdots \cup M_{i-1}$.

Étant donné un ensemble partiellement ordonné $\mathcal{V}$, on peut trivialement en obtenir une partition en antichaînes compatible en partitionnant $\mathcal{V}$ en singletons qu'on ordonne selon n'importe quelle extension linéaire. Pour les six génomes étudiés à la section 2.4, une possibilité est

$$AHBKCDIEMLFGJN.$$

Cette partition est associée à un alignement trivial des six génomes qui correspond à placer chaque variante dans son propre module, comme montré à la figure 2.10.

Comme chaque variante forme son propre module, cet alignement est inintéressant d'un point de vue biologique : selon l'interprétation habituelle des alignements

1	2	3	4	5	6	7	8	9	10	11	12	13	14
A		B		C	D		E			F	G		N
A		B			D		E			F	G		N
A		B				I				F		J	N
A		B		C					L	F		J	N
A	H				D			M		F	G		N
A			K							F		J	N

Figure 2.10 Un alignement trivial de six génomes

multiples, chaque module qui n'est pas présent dans tous les génomes correspond à un gain d'une fonction biologique pour les génomes qui le possède, et à la perte d'une fonction biologique pour les génomes qui ne le possèdent pas. L'alignement trivial de la figure 2.10 affirme donc l'existence de 11 événements de gain ou de perte, puisque seuls trois des 14 variantes sont partagées par l'ensemble des génomes.

On peut faire beaucoup mieux, c'est-à-dire obtenir des alignements avec un plus petit nombre de colonnes, autrement dit de modules. Le théorème de Mirsky (Mirsky, 1971) relie le nombre minimal de modules avec la longueur maximale d'une chaîne de $\mathcal{V}$. Nous donnerons la preuve de ce théorème puisqu'elle est constructive et nous permettra de donner un algorithme pour construire deux alignements particuliers contenant un nombre minimal de modules.

Théorème 1 (Mirsky). *Le nombre minimal d'antichaînes dans une partition en antichaînes d'un ensemble $\mathcal{V}$ est égal à la longueur maximale d'une chaîne de $\mathcal{V}$.*

Ce théorème se démontre par récurrence sur m, la longueur maximale d'une chaîne de $\mathcal{V}$.

Le cas de base $m = 1$ est trivial : si chaque chaîne de $\mathcal{V}$ est de longueur 1, cela veut dire que tous les éléments de $\mathcal{V}$ sont incomparables, et donc que $\mathcal{V}$ forme une

antichaîne.

Maintenant, soit $m > 1$ et supposons que le théorème est vrai pour tout ensemble dont la longueur maximale d'une chaîne est $m - 1$. Considérons un ensemble $\mathcal{V}$ dont la longueur maximale d'une chaîne est m, et formons le sous-ensemble M de tous les éléments de $\mathcal{V}$ qui sont les éléments maximaux d'une chaîne de longueur m. Puisqu'il existe dans $\mathcal{V}$ une chaîne de longueur m, alors M est non vide. De plus, puisque m est la longueur maximale d'une chaîne de $\mathcal{V}$, les différents éléments de M sont incomparables. En d'autres termes, M est une antichaîne. Par construction, la longueur maximale d'une chaîne dans $\mathcal{V} \setminus M$ est $m - 1$. Par l'hypothèse de récurrence, $\mathcal{V} \setminus M$ peut donc être partitionné minimalement en $m - 1$ antichaînes. Ces $m - 1$ antichaînes, plus l'antichaîne M elle-même, forment donc une partition de $\mathcal{V}$ en m antichaînes. □

On peut aussi présenter cette preuve sous la forme d'un algorithme pour construire une partition en antichaînes de taille minimale, comme montré à l'algorithme 2.1.

Algorithme 2.1 Partition en un nombre minimal d'antichaînes d'un ensemble partiellement ordonné représenté par un graphe orienté acyclique

```
fonction PARTITIONNE-EN-ANTICHAÎNES(G : graphe)
    partition ← ∅
    tant que G ≠ ∅ faire
        maximaux ← sommets
        pour chaque sommet s du graphe faire
            pour chaque successeur v de s faire
                maximaux ← maximaux \{v}
        partition ← partition ∪ maximaux
        G ← G \ maximaux
    retourner partition
```

Dans la preuve du théorème de Mirsky, on construit une partition de $\mathcal{V}$ en antichaînes comportant un nombre minimal d'antichaînes, en retirant itérativement de $\mathcal{V}$ tous les éléments maximaux et en plaçant ceux-ci dans une antichaîne. En

1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	N
A	B		D	E	F	G	N
A	B	I			F	J	N
A	B	C	L		F	J	N
A	H		D	M	F	G	N
A	K				F	J	N

1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	N
A	B		D	E	F	G	N
A	B			I	F	J	N
A	B	C		L	F	J	N
A		H	D	M	F	G	N
A				K	F	J	N

Figure 2.11 Alignements canoniques gauche et droit de six génomes

retirant plutôt les éléments minimaux au lieu des maximaux, on a une deuxième façon de partitionner $\mathcal{V}$ en antichaînes qui donne aussi une partition minimale, mais généralement différente de la première.

Par exemple, dans le graphe d'alignement des six bactériophages de la section 2.4, on remarque que la longueur maximale d'une chaîne est de 8 sommets. Un exemple de telle chaîne est ABCDEFGN. Selon le théorème de Mirsky, on doit donc pouvoir partitionner les sommets du graphe d'alignement en 8 antichaînes, c'est-à-dire en 8 modules plutôt que 14 comme dans l'alignement trivial de la figure 2.10. En fait, nous allons former deux telles partitions.

La *partition canonique gauche* est obtenue en choisissant itérativement les ensembles m_i d'éléments minimaux de l'ensemble partiellement ordonné $\mathcal{V}$, et en retirant ces éléments de $\mathcal{V}$. La partition canonique gauche pour l'exemple est illustrée dans la partie de gauche de la figure 2.11.

La *partition canonique droite* est obtenue en choisissant itérativement les ensembles M_i d'éléments maximaux de $\mathcal{V}$, et en retirant ces éléments de $\mathcal{V}$. La partition canonique droite pour l'exemple est illustrée dans la partie de droite de la figure 2.11.

Les deux partitions canoniques montrent les positions extrêmes que peuvent occu-

per les variantes. On voit que certaines variantes sont à la même position (dans le même module) dans les deux alignements canoniques, mais que d'autres changent de position.

Définition 4. *Une* variante fixe *est une variante dont la position est identique dans les deux alignements canoniques. Une* variante mobile *est une variante qui n'est pas fixe.*

Par exemple, A, B, F et J sont des variantes fixes, tandis que I est une variante mobile puisqu'elle fait partie du module 3 dans l'alignement canonique gauche mais du module 5 dans l'alignement canonique droit.

Théorème 2. *Une variante est fixe si et seulement si elle fait partie d'une chaîne de longueur maximale.*

Direction « seulement si » : supposons qu'une variante x est fixe, c'est-à-dire qu'elle fait partie du même module i dans la partition canonique gauche et dans la partition canonique droite d'un ensemble $\mathcal{V}$ dont la longueur maximale d'une chaîne, c'est-à-dire le nombre de modules, est m. La variante x ayant été choisie à l'étape i dans la construction de la partition canonique gauche, il existe donc une chaîne de longueur i qui va d'un élément minimal de $\mathcal{V}$ jusqu'à x. De même, la variante x ayant été choisie à l'étape $m-i+1$ dans la construction de la partition canonique droite, il existe une chaîne de longueur $m-i+1$ qui va de x à un élément maximal de $\mathcal{V}$. Mais en concaténant ces deux chaînes, on trouve que x fait partie d'une chaîne de longueur m qui va d'un élément minimal de $\mathcal{V}$ à un élément maximal de $\mathcal{V}$, donc qu'elle fait partie d'une chaîne de longueur maximale.

Pour montrer l'autre direction, démontrons d'abord un lemme.

Lemme 1. *Un élément x faisant partie d'une chaîne de longueur maximale m de $\mathcal{V}$ apparaît en même position dans toutes les chaînes de longueur maximale dont*

il fait partie.

Preuve du lemme, par contradiction. Supposons que x apparaisse en position i dans une chaîne de longueur maximale $a \cdots x \cdots b$ et qu'il apparaisse en position $j < i$ dans une autre chaîne de longueur maximale $c \cdots x \cdots d$. Alors on pourrait former une chaîne $a \cdots x \cdots d$ qui serait plus longue qu'une chaîne maximale, ce qui est une contradiction. □

Terminons maintenant la preuve du théorème.

Direction « si » : supposons qu'une variante x fait partie d'une chaîne de longueur maximale m de $\mathcal{V}$. Par le lemme, il est en même position i dans chaque chaîne de longueur maximale dont il fait partie. Donc x sera choisie à l'étape i pour la construction de la partition canonique gauche et fera partie du i-ème module à partir de la gauche. De même x sera choisie et à l'étape $m - i + 1$ pour la construction de la partition canonique droite, et fera partie du $m - i + 1$-ième module à partir de la droite, qui est la même chose que le i-ème module à partir de la gauche. □

On peut inférer que les variantes fixes sont des variantes du module correspondant à leur position. Nous avons donc placé toutes les variantes fixes :

- Module 1 : A
- Module 2 : B
- Module 3 : C
- Module 4 : D
- Module 5 : E, M
- Module 6 : F
- Module 7 : G, J
- Module 8 : N

Pour les variantes mobiles, on ne peut pas les placer de manière univoque dans

un module : il reste plusieurs possibilités. Cependant, on peut faire la liste des inférences qui sont compatibles avec les alignements canoniques.

— H est dans le module 2 ou 3.

— I est dans le module 3, 4 ou 5.

— K est dans le module 2, 3, 4 ou 5.

— L est dans le module 4 ou 5.

On peut en déduire un résultat négatif : les variantes H et L n'appartiennent pas au même module.

*
* *

L'utilisation des ancres, c'est-à-dire de séquences similaires communes à plusieurs génomes, comme point de départ pour obtenir un alignement multiple, est une idée commune à de nombreux algorithmes d'alignement multiple. Certains de ces algorithmes utilisent des graphes d'alignement semblables aux nôtres, qui définissent un ordre partiel sur les différents segments des génomes impliqués (Kehr *et al.*, 2014). Toutefois, la plupart de ces algorithmes n'utilisent le graphe que comme une étape intermédiaire dans la construction de l'alignement et effectuent rapidement une linéarisation du graphe pour se ramener à une comparaison de séquences, comme nous l'avons fait ci-dessus avec les partition canoniques. L'aligneur *progressiveMauve* (Darling *et al.*, 2010), par exemple, identifie les ancres locales — des ancres qui sont présentes dans plusieurs génomes mais pas forcément tous — pour établir une relation d'ordre partiel, mais les alignements générés par Mauve sont linéarisés, ce qui fait perdre les propriétés combinatoires de l'ordre partiel.

Dans (Lee *et al.*, 2002), l'idée est mentionnée de travailler directement avec la relation d'ordre partiel au lieu de passer par la linéarisation. Cette approche est celle que nous avons adoptée pour l'étude des génomes de phages. Au lieu de

produire un alignement sous une forme linéaire, nous générons un graphe qui permet d'apprécier plus finement les particularités de l'ordre partiel de l'ensemble des génomes de phages étudiés.

CHAPITRE III

COLINÉARITÉ FONCTIONNELLE

Dans tous les exemples du chapitre précédent, le graphe engendré par les appariements ne contient pas de cycle. L'acyclicité du graphe des colonnes est une propriété essentielle pour la suite de notre étude, car elle garantit qu'on ne retrouvera jamais deux positions différentes d'un même génome dans une même classe d'équivalence. Mettons un nom sur cette propriété souhaitable.

Définition 5. *Lorsque le graphe des colonnes engendré par un ensemble d'appariements $\mathcal{A}$ sur un ensemble de génomes $\mathcal{G}$ est acyclique, nous dirons que $\mathcal{G}$ est* fonctionnellement colinéaire *par rapport à $\mathcal{A}$.*

Le choix du terme *fonctionnellement colinéaire* s'explique par le fait qu'idéalement, les sommets du graphe contracté seront associées à des fonctions biologiques. Un ensemble de génomes fonctionnellement colinéaire est donc un ensemble de génomes dans lesquels les fonctions biologiques associées aux sommets du graphe sont présentes dans le même ordre.

La colinéarité fonctionnelle d'un ensemble de génomes est, par définition, une condition suffisante pour qu'on puisse en faire l'étude au moyen des techniques décrites ici.

Dans la pratique, on vérifie la colinéarité fonctionnelle en construisant le graphe

d'alignement et en vérifiant qu'il est acyclique. Si un cycle est présent, on procède à certains ajustements qui seront détaillés dans les sections suivantes pour tenter d'éliminer le cycle. Dans la pratique, les génomes des bactériophages possèdent de bonnes propriétés qui font que le graphe d'alignement est souvent acyclique dès le départ, ou peut le devenir avec un minimum d'ajustements.

Dans un souci d'économie de calculs, il est intéressant de se demander si on peut identifier les conditions nécessaires sur l'ensemble $\mathcal{A}$ des appariements pour qu'un ensemble $\mathcal{G}$ de génomes soit fonctionnellement colinéaire par rapport à $\mathcal{A}$, et réfléchir à ce qui pourrait mal tourner lors de la construction du graphe des colonnes et du graphe d'alignement. Intéressons-nous d'abord au cas le plus simple, celui où $\mathcal{G}$ ne contient que deux génomes, G_1 et G_2.

3.1 Le problème des duplications

La duplication est un phénomène par lequel une section génomique est reproduite exactement à un autre endroit du génome. Les duplications sont des événements fréquents qui sont une source de variabilité génétique. Lorsque la copie apparaît immédiatement à la suite de l'original, on parle de duplication en tandem.

Le phénomène biologique de duplication correspond à l'existence d'appariements non disjoints. Dans la figure 3.1, une duplication fait que la séquence `TCAC` apparaît deux fois dans le génome du haut, conduisant à deux appariements non disjoints.

L'absence de duplications est une condition nécessaire pour avoir la colinéarité

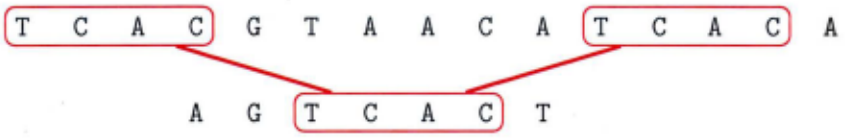

Figure 3.1 Une duplication conduisant à deux appariements non disjoints

fonctionnelle.

Théorème 3. *Pour qu'un ensemble $\mathcal{G}$ de génomes soit fonctionnellement colinéaire par rapport à un ensemble $\mathcal{A}$ d'appariements, il est nécessaire que les appariements de $\mathcal{A}$ soient disjoints, c'est-à-dire que chaque position dans $\mathcal{G}$ ne fasse partie que d'un seul appariement au maximum.*

En effet, si les appariements ne sont pas disjoints, cela veut dire qu'il y a une position d'un génome (sans perte de généralité, disons que c'est G_1) qui se retrouve dans la même classe d'équivalence que deux positions dans G_2, disons p_{2i} et p_{2j} avec $i < j$. De par la construction du graphe des colonnes, il y aura un arc de $[\![p_{2i}]\!]$ à $[\![p_{2(i+1)}]\!]$, un autre de $[\![p_{2(i+1)}]\!]$ à $[\![p_{2(i+2)}]\!]$, et ainsi de suite jusqu'à $[\![p_{2j}]\!]$ mais cette classe est la même que $[\![p_{2i}]\!]$, ce qui forme un cycle. □

On peut écrire formellement la contrainte des appariements disjoints de la façon suivante : si $\mathcal{A}$ contient $A_1 = \{G_1[s_1..t_1], G_2[u_1..v_1]\}$ et $A_2 = \{G_1[s_2..t_2], G_2[u_2..v_2]\}$, alors les deux conditions suivantes doivent être remplies : d'abord $s_2 > t_1$ ou $s_1 > t_2$, et ensuite $u_2 > v_1$ ou $u_1 > v_2$.

Pour éviter d'avoir des appariements non disjoints, il faudra imposer des conditions aux appariements. Si on n'imposait aucune condition, alors (pour prendre un exemple extrême) chaque nucléotide `A` d'un génome pourrait être mis en appariement avec chaque nucléotide `A` de tous les autres génomes. Une condition qui permet d'éviter d'avoir des appariements non disjoints est de calculer la longueur ℓ de la plus longue séquence génomique qui apparaît plus d'une fois dans l'ensemble des génomes, et d'exiger que pour faire partie d'un appariement, une séquence soit plus longue que ℓ.

Le calcul de ℓ peut se faire en temps linéaire en la somme des longueurs des génomes, par exemple en utilisant un tableau suffixe. De nombreux outils en ligne

et hors ligne existent pour effectuer ce calcul.

Pour l'exemple de la section 2.4, où nous avons construit le graphe d'alignement de six génomes de bactériophages, nous avons utilisé l'outil en ligne *Reputer* (Kurtz et Schleiermacher, 1999) pour calculer la longueur des plus longues répétitions dans chacun des génomes. Nous avons permis une distance de Hamming allant jusqu'à 5 lors de la comparaison, de manière à tenir compte de la présence possible de mutations ponctuelles entre les répétitions. Les résultats sont présentés dans le tableau 3.1. La longueur maximale d'une répétition étant de 40 nucléotides, nous avons donc utilisé 50 nucléotides comme longueur minimale pour une variante.

Tableau 3.1 Longueur des répétitions maximales dans les séquences génomiques extraites de *S. aureus*, pour des distances de Hamming variant de 0 à 5

Distance de Hamming	Phage 85	Phage 92	Phage 88	Phage 29	Phage 53	Phage 187
0	16	15	15	31	15	35
1	28	22	16	22	18	19
2	25	27	27	21	27	21
3	33	24	27	28	21	24
4	37	26	26	**40**	26	24
5	30	30	31	30	30	27

3.2 Le problème des transpositions

Le critère des appariements disjoints est une condition nécessaire pour garantir la colinéarité fonctionnelle, mais ce n'est pas une condition suffisante. Considérons les deux appariements de la figure 3.2. Dans ce cas, les deux appariements sont disjoints mais ils n'apparaissent pas dans le même ordre dans G_1 et G_2. Le phénomène biologique associé à ce phénomène est la *transposition*. On dira donc que les deux appariements sont en transposition.

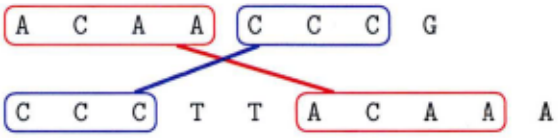

Figure 3.2 Deux appariements en transposition

Formellement, si $\mathcal{A}$ contient les deux appariements $A_1 = \{G_1[s_1..t_1], G_2[u_1..v_1]\}$ et $A_2 = \{G_1[s_2..t_2], G_2[u_2..v_2]\}$, alors l'absence de transposition entre les appariements A_1 et A_2 signifie que ou bien $s_1 < s_2$ et $u_1 < u_2$, ou bien $s_1 > s_2$ et $u_1 > u_2$.

L'absence de transposition est une condition nécessaire pour avoir la colinéarité fonctionnelle.

Théorème 4. *Si $\mathcal{A}$ contient deux appariements en transposition, cela conduit à un cycle dans le graphe des colonnes.*

Nous écrirons $[\![p_{ij}]\!] \rightsquigarrow [\![p_{kl}]\!]$ pour dire qu'il y a un chemin (un arc ou une séquence d'arcs) de $[\![p_{ij}]\!]$ à $[\![p_{kl}]\!]$ dans le graphe des colonnes. Notons que, par la construction du graphe des colonnes, si $i < j$ sont deux positions dans un génome donné g, alors $[\![p_{gi}]\!] \rightsquigarrow [\![p_{gj}]\!]$.

Sans perte de généralité, supposons que $\mathcal{A}$ contient $A_1 = \{G_1[s_1..t_1], G_2[u_1..v_1]\}$ et $A_2 = \{G_1[s_2..t_2], G_2[u_2..v_2]\}$, avec $s_1 < s_2$ et $u_1 > u_2$. Alors :

- Puisque s_1 précède s_2 dans le génome 1, on doit avoir $[\![p_{1s_1}]\!] \rightsquigarrow [\![p_{1s_2}]\!]$.
- Puisque u_2 précède u_1 dans le génome 2, on doit avoir $[\![p_{2u_2}]\!] \rightsquigarrow [\![p_{2u_1}]\!]$.
- À cause de l'appariement A_1, on doit avoir $[\![p_{1s_1}]\!] = [\![p_{2u_1}]\!]$.
- À cause de l'appariement A_2, on doit avoir $[\![p_{1s_2}]\!] = [\![p_{2u_2}]\!]$.

En combinant ces quatre résultats, on a donc le cycle

$$[\![p_{1s_1}]\!] \rightsquigarrow [\![p_{1s_2}]\!] = [\![p_{2u_2}]\!] \rightsquigarrow [\![p_{2u_1}]\!] = [\![p_{1s_1}]\!]$$

ce qu'il fallait montrer. □

Comme nous l'avons déjà expliqué, les génomes des phages d'une même famille ont la propriété que les modules apparaissent dans le même ordre. Toutefois, comme les génomes des phages possèdent une phase circularisée, la notion de début du génome n'est pas immédiatement claire, et si deux génomes de phages ne s'accordent pas quant à la position du début, il risque d'apparaître des transpositions fantômes, dont l'existence est seulement due à la permutation circulaire.

Pour pallier ce problème, il suffit d'identifier, dans l'ensemble des phages que nous étudierons, un module particulier qui est commun à tous les phages. Un bon candidat est le module qui code pour la protéine intégrase, puisqu'un bactériophage doit posséder une variante de l'intégrase pour pouvoir intégrer le génome de sa bactérie hôte. De plus, ce module est très souvent annoté comme tel dans les banques de données, donc son identification est facile même en présence d'une variante nouvelle.

En permutant circulairement les génomes de manière à toujours les faire débuter par un même module, on fait disparaître les transpositions fantômes. S'il reste des vraies transpositions, ce qui est rare, alors le plus simple est de considérer que les phages en question font partie d'une famille différente, ou bien encore de régler le problème « à la main » en détransposant les modules en question. L'outil informatique que nous décrivons dans ce mémoire ne permet pas de traiter automatiquement les transpositions. Le mieux qu'il puisse faire est de diviser les génomes en modules plus grossiers dans l'espoir que les transpositions se fassent absorber à l'intérieur d'un même super-module.

3.3 Comparaison de plus de deux génomes

Si l'ensemble $\mathcal{G}$ ne contient que deux génomes, les deux cas ci-dessus (duplication et transposition) sont les seuls qui peuvent faire en sorte que $\mathcal{G}$ ne soit pas fonctionnellement colinéaire. Mais si on a plus que deux génomes, les choses ne sont pas aussi simples. En particulier, même en l'absence de transposition et de duplication, le fait que deux génomes quelconques de $\mathcal{G}$ soient toujours fonctionnellement colinéaires n'implique pas que $\mathcal{G}$ est fonctionnellement colinéaire, comme le montre la figure 3.3 avec trois génomes. Dans cet exemple, les lettres a, b, c représentent des séquences de nucléotides.

Deux génomes quelconques parmi les trois sont certainement colinéaires, puisqu'il n'y a qu'un appariement entre eux. Mais quand on regroupe les trois génomes, les appariements provoquent l'apparition d'un cycle $a \to b \to c \to a$ dans le graphe d'alignement.

En généralisant cet exemple, on peut construire pour tout $n \geq 3$ un ensemble non fonctionnellement colinéaire de $n+1$ génomes, mais dont tout sous-ensemble de n génomes est fonctionnellement colinéaire. La figure 3.4 montre une façon de construire un tel ensemble.

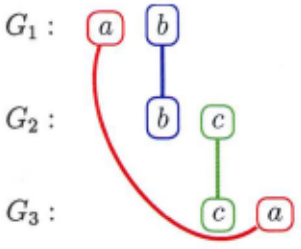

Figure 3.3 Ensemble de trois génomes non fonctionnellement colinéaires

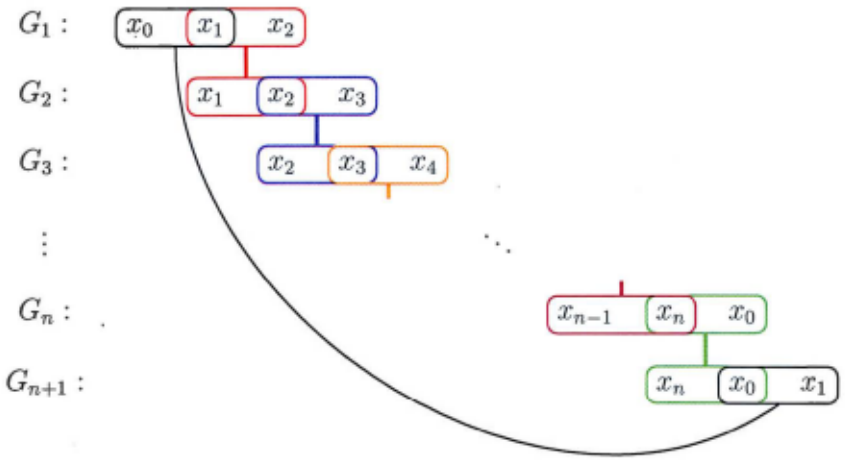

Figure 3.4 Ensemble de n génomes non fonctionnellement colinéaires

Comment éviter que de tels phénomènes se produisent ? Il est tentant de penser que la présence d'un ordre total sur les appariements implique la colinéarité fonctionnelle. Ce qu'on entend par *ordre total* ici est un ordre $\leq$ sur les appariements, défini ainsi : $A_i \leq A_j$ si, dans tout génome impliqué dans les appariements A_i et A_j, la première position de A_i n'est pas située après la première position de A_j. Autrement dit : si on parcourt chacun des génomes de $\mathcal{G}$ et qu'on note les appariements à mesure qu'on les rencontre, il n'arrivera jamais qu'on rencontre un appariement A_i avant un autre appariement A_j dans un certain génome, et qu'on les rencontre dans l'ordre inverse dans un autre génome.

L'existence d'un tel ordre total est certainement quelque chose de souhaitable. Malheureusement, la présence d'un ordre total n'implique pas la colinarité fonctionnelle, même avec seulement trois génomes, comme le montre le contre-exemple de la figure 3.5.

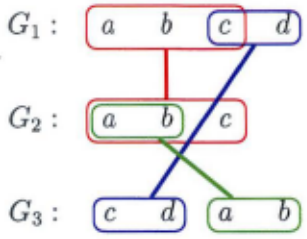

Figure 3.5 Trois génomes non fonctionnellement colinéaires mais avec un ordre total sur les appariements

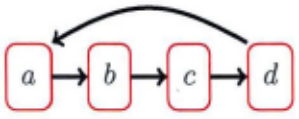

Figure 3.6 Graphe contracté correspondant aux appariements de la figure 3.5

L'ensemble d'appariements est $\mathcal{A} = \{A_1, A_2, A_3\}$ et il possède l'ordre total $A_1 \leq A_2 \leq A_3$. Pourtant, cet ensemble de génomes n'est pas fonctionnellement colinéaire puisque son graphe contracté, montré à la figure 3.6, contient un cycle.

Dans l'exemple ci-dessus, on peut noter que la séquence *ab* qui se retrouve dans le troisième génome n'a pas été mise en appariement avec la séquence *ab* du premier génome. Rien, dans la définition des appariement, n'empêche cela, mais c'est quand même un peu bizarre que *ab* forme un appariement entre les génomes G_2 et G_3, mais pas entre G_1 et G_3 On dira que cet ensemble d'appariements n'est pas complet. Serait-ce pour cette raison que nous n'avions pas de colinéarité fonctionnelle dans l'exemple précédent ? Pour tenter une réponse, il convient de réfléchir à différentes propriétés souhaitables que pourrait avoir un ensemble d'appariements.

Définition 6. *Un ensemble d'appariements est ...*

— unique *si une séquence qui fait partie d'un appariement ne se trouve pas ailleurs sur le même génome.*

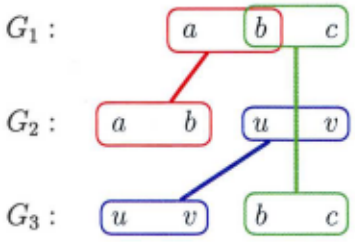

Figure 3.7 Trois génomes non fonctionnellement colinéaires mais dont les appariements sont uniques, maximaux et complets

- maximal *si on ne peut pas allonger les appariements vers la gauche ou vers la droite. Autrement dit, lorsque* $G_1[s_1..t_1], G_2[s_2..t_2] \in \mathcal{A}$, *alors* $G_1[s_1-1] \neq G_2[s_2-1]$ *et* $G_1[s_1+1] \neq G_2[s_2+1]$.
- complet *si tous les appariements qu'on peut déduire par transitivité sont présents. Autrement dit, lorsque* $G_1[s_1..t_1], G_2[s_2..t_2] \in \mathcal{A}$ *et que* $G_1[s_1..t_1] = G_3[s_3..t_3]$, *alors* $\{G_1[s_1..t_1], G_3[s_3..t_3]\} \in \mathcal{A}$ *et* $\{G_2[s_2..t_2], G_3[s_3..t_3]\} \in \mathcal{A}$.

Avec cette définition, on peut dire que l'exemple de la figure 3.5 ne faisait pas intervenir un ensemble d'appariements complet. Il était par contre unique et maximal.

Si les trois conditions d'unicité, de maximalité et complétude sont satisfaites pour un ensemble d'appariements qui possède un ordre total, peut-on en conclure qu'alors on a forcément colinéarité fonctionnelle ? Eh bien non. La figure 3.7, encore sur trois génomes seulement, montre un ensemble d'appariements unique, maximal et complet, possédant un ordre total, mais tel que le graphe des colonnes est cyclique.

L'ensemble d'appariements est $\mathcal{A} = \{A_1, A_2, A_3\}$ et il possède l'ordre total $A_1 \leq A_2 \leq A_3$. On voit aisément qu'il est unique, maximal et complet. Pourtant, son graphe contracté est cyclique comme le montre la figure 3.8.

Tous les exemples de cette section peuvent sembler décourageants et faire croire

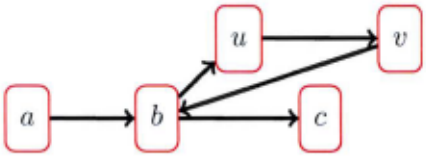

Figure 3.8 Graphe contracté correspondant aux appariements de la figure 3.7

que la colinéarité fonctionnelle est une propriété très rare. Dans les faits, il n'en est rien, du moins quand on travaille avec des génomes de bactériophages. Les exemples ci-dessus présentent un intérêt théorique mais ils ne sont pas rencontrés dans la réalité des phages que nous avons étudiés.

*
* *

Nous avons identifié deux conditions nécessaires pour avoir la colinéarité fonctionnelle : l'absence de duplication dans les appariements et l'absence de transposition entre les appariements. Nous avons vu qu'à partir d'un ensemble de trois génomes, ces conditions nécessaires ne sont pas suffisantes. Le problème d'identifier des conditions suffisantes pour garantir la colinéarité fonctionnelle dans le cas général est un problème ouvert.

CHAPITRE IV

ALGORITHMES POUR LA CRÉATION DU GRAPHE D'ALIGNEMENT

Maintenant que nous avons vu les idées théoriques principales qui sous-tendent l'utilisation de graphes basés sur les appariements pour comparer des génomes de bactériophages, nous allons discuter de l'implantation de ces idées et décrire un outil informatique appelé Alpha — *al*ignement de génomes de *pha*ges — qui est le premier aligneur conçu spécifiquement pour l'alignement et la comparaison de génomes entiers de bactériophages. En mettant à profit les idées décrites dans les trois chapitres précédents, Alpha permet de révéler la structure en mosaïque des génomes de bactériophages.

En plus de générer un graphe d'alignement, Alpha fournit une interface graphique interactive qui permet de manipuler celui-ci avec un niveau de détail variable. Il permet donc de comparer des structures à grande échelle en se concentrant sur les ancres, mais aussi d'effectuer des zooms avant et arrière, de cacher ou d'afficher les mutations ponctuelles, etc.

Dans ce chapitre, nous verrons les algorithmes qui permettent de construire les structures vues aux chapitres précédents, c'est-à-dire le graphe des colonnes, le graphe d'alignements et le graphe contracté.

4.1 Des génomes aux appariements

Étant donné un ensemble $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ de n séquences génomiques et un entier m, la première étape est de trouver tous les appariements de longueur au moins m dans $\mathcal{G}$. Il s'agit du problème classique d'identifier des répétitions maximales dans un ensemble de chaînes. Nous utilisons la fonction *repfind* de l'outil GenomeTools (Gremme *et al.*, 2013) pour le faire.

GenomeTools est un programme qui utilise un tableau suffixe augmenté d'autres tableaux auxiliaires pour déterminer toutes les répétitions maximales en espace $\mathcal{O}(N)$ et en temps $\mathcal{O}(N + k)$, où N est la somme des longueurs des génomes et k est le nombre d'appariements trouvés, ce qui est asymptotiquement optimal. L'algorithme qui identifie les répétitions maximales est bien connu mais il est assez long et complexe. Une explicitation de l'algorithme nous entraînerait trop loin du sujet principal de ce mémoire. Le lecteur intéressé peut consulter (Gusfield, 1997) où l'algorithme est décrit en détail en termes d'un arbre suffixe. Dans (Abouelhoda *et al.*, 2004), on explique comment modifier cet algorithme — et en général n'importe quel algorithme utilisant un arbre suffixe — pour utiliser un tableau suffixe augmenté au lieu d'un arbre suffixe, sans perte d'efficacité.

4.2 Des appariements aux graphes

L'algorithme 4.1 présente une vue d'ensemble de la façon un ensemble d'appariements permet d'obtenir le graphe des colonnes et le graphe d'alignement. Cet algorithme prend en entrée un ensemble $\mathcal{G}$ de séquences génomiques et un ensemble $\mathcal{A}$ d'appariements de longueur minimale m, et il construit le graphe des colonnes décrit à la section 2.1. Si le graphe des colonnes ne contient pas de cycle, il fusionne certaines colonnes pour former le graphe d'alignement, tel qu'indiqué à la section 2.2.

Algorithme 4.1 Construction du graphe des colonnes et du graphe d'alignement à partir de la liste des appariements

fonction CONSTRUCTION-DE-GRAPHE($\mathcal{G}$: génomes, $\mathcal{A}$: appariements)
  ▷ Étape 1. Construction des colonnes
  pour chaque génome G de $\mathcal{G}$ **faire**
    pour chaque position p de G **faire**
      $[\![p_{ij}]\!] \leftarrow \{p\}$
  pour chaque appariement A de $\mathcal{A}$ **faire**
    pour chaque égalité $G_i[j] = G_k[l]$ de A **faire**
      Fusionner les classes $[\![p_{ij}]\!]$ et $[\![p_{kl}]\!]$

  ▷ Étape 2. Dissociation des colonnes invalides
  pour chaque classe $[\![p_{ij}]\!]$ **faire**
    si $|[\![p_{ij}]\!]| \neq |\mathrm{sp}\,[\![p_{ij}]\!]|$ **alors**
      Séparer la classe $[\![p_{ij}]\!]$ en singletons

  ▷ Étape 3. Construction du graphe des colonnes
  Chaque classe d'équivalence devient un sommet du graphe C
  pour chaque génome G de $\mathcal{G}$ **faire**
    pour chaque paire de positions consécutives p_j et p_{j+1} de G **faire**
      Ajouter à C un arc de $[\![p_j]\!]$ à $[\![p_{j+1}]\!]$

  ▷ Étape 4. Si C est acyclique, le transformer en graphe d'alignement
  si le graphe C est acyclique **alors**
    Fusionner les paires de colonnes consécutives qui ont le même support
  retourner C

Dans l'étape 1, l'algorithme commence par placer chaque position de chaque génome dans sa propre colonne, ce qui prend un temps et un espace dans $\mathcal{O}(N)$, où N est la somme des longueurs des génomes. Il utilise ensuite une structure d'*ensembles disjoints* (Galler et Fisher, 1964) pour construire les colonnes en fusionnant deux colonnes lorsqu'un appariement nous permet de le faire. Cette partie prend un temps dans $\mathcal{O}(M\alpha(M))$ où M est la somme des longueurs des appariements, et α est la réciproque de la fonction d'Ackermann. On a $\alpha(M) \leq 5$ pour toute valeur raisonnable de M, donc le temps est, à toutes fins pratiques, essentiellement linéaire en M (Tarjan, 1975).

Dans l'étape 2, les colonnes (c'est-à-dire les classes d'équivalence) sont examinées pour voir si elles sont invalides. Une colonne invalide est une colonne qui contient plus d'une position appartenant à un même génome. Cela revient à dire qu'une colonne est invalide si sa taille diffère de la taille de son support, ce qui explique la ligne 11 de l'algorithme 4.1. Les colonnes invalides sont dissociées en singletons, c'est-à-dire que chaque position devient une nouvelle classe d'équivalence. Cela revient à éliminer tous les appariements qui impliquent cette colonne, que ce soit directement ou par transitivité de la relation.

Dans l'étape 3, le graphe des colonnes est fabriqué en insérant des arcs entre toutes les colonnes qui incluent des positions successives dans un même génome.

Finalement, à l'étape 4, un test d'acyclicité est fait sur le graphe des colonnes. Si le graphe est cyclique, le traitement s'arrête là et un message d'avertissement est émis à l'usager. Si le graphe est acyclique, les colonnes successives qui impliquent les même génomes sont fusionnées de manière à former le graphe d'alignement.

4.3 Structure d'ensembles disjoints

Le programme Alpha est écrit en Python, un langage qui comporte de nombreux avantages mais qui n'est pas particulièrement connu pour la rapidité d'exécution. Un profilage à l'aide du programme RunSnakeRun (Fletcher, 2013) a montré que la construction du graphe d'alignement est la partie du programme où l'efficacité est importante. C'est pourquoi, après avoir implanté la structure d'ensembles disjoints en Python pour confirmer son bon fonctionnement, cette partie a été récrite en langage C en tentant d'en optimiser la vitesse.

Les opérations que nous devons implanter sont les suivantes :

— INITIALISER : construit les classes d'équivalence initiales, contenant chacune une position.

— RECHERCHER : retourne, pour une position, le représentant de sa classe d'équivalence.

— FUSIONNER : fusionne les classes d'équivalences de deux positions.

— DISSOCIER : scinde une classe d'équivalence en singletons.

— COLLER : fusionne deux classes d'équivalences consécutives dans le graphe des colonnes.

Les opérations INITIALISER, RECHERCHER et FUSIONNER sont habituelles dans une structure d'ensembles disjoints, et l'implantation standard, qui utilise des tableaux, est donnée dans l'algorithme 4.2.

Algorithme 4.2 Structure classique d'ensembles disjoints avec compression des chemins et union selon le rang

```
    fonction INITIALISER(N : nombre d'éléments)
        pour k de 0 à N faire
            parent[k] ← k
            rang[k] ← 0
 5: fonction RECHERCHER(x)
        si x ≠ parent[x] alors
            parent[x] ← RECHERCHER(parent[x])
        retourner parent[x]
    fonction FUSIONNER(x, y)
10:     x_r ← RECHERCHER(x)
        y_r ← RECHERCHER(y)
        si x_r = y_r alors
            retourner
        si rang[x_r] < rang[y_r] alors
15:         échanger x_r et y_r
        si rang[x_r] = rang[y_r] alors
            rang[x_r] ← rang[x_r] + 1
        parent[y_r] ← x_r
```

L'opération DISSOCIER, quand à elle, nécessite une manière efficace de lister tous les éléments d'une classe d'équivalence. Elle est implantée au moyen d'une liste circulaire simplement chaînée qui permet de parcourir tous les éléments d'une classe d'équivalence donnée.

Finalement, l'opération COLLER permet, lorsque le graphe des colonnes est acyclique, de former le graphe d'alignement. Elle est implantée en ajoutant un attribut *largeur* à chaque classe d'équivalence. Cet attribut, initialement 1, indique le nombre de positions successives couvertes par la classe d'équivalence. Une liste circulaire doublement chaînée qui parcourt les représentants de chaque classe d'équivalence permet de lister toutes les classes rapidement.

Pour des raisons d'efficacité, les attributs listés ci-dessus sont implantés au moyen de tableaux, comme les attributs *parent* et *rang* de la structure classique. On imagine les n génomes G_1 à G_n, de longueur totale, $N = |G_1| + |G_2| + \cdots + |G_n|$, concaténés les uns aux autres. Chaque entier de 0 à $N - 1$ représente donc une position dans un génome. Des tableaux $genome[0..N]$ et $pos[0..N]$ permettent de savoir en temps constant le génome et la position représentée par un entier $0 \leq i < N$. Nous avons les tableaux supplémentaires suivants, tous de taille N.

- *taille* donne le nombre de nucléotides inclus dans chaque classe d'équivalence.
- *frere* implante une liste chaînée qui garde trace des positions comprises dans les classes d'équivalence.
- *largeur* indique le nombre de nucléotides successifs inclus dans un sommet du graphe d'alignement
- *precedent* et *suivant* implantent une liste doublement chaînée qui parcourt les représentants de chaque classe d'équivalence.

L'algorithme 4.3 détaille les modifications à apporter aux trois fonctions de base afin de maintenir les nouveaux tableaux. Aucun changement n'est nécessaire à la fonction RECHERCHER. Par contre, lorsqu'on fusionne deux classes d'équivalence, il faut additionner la taille des deux classes, joindre les listes des positions et retirer le représentant d'une des deux classes de la liste des représentants.

La figure 4.1 montre comment, à la ligne 27 de l'algorithme 4.3, l'échange des va-

Algorithme 4.3 Structure d'ensembles disjoints enrichie

```
fonction INITIALISER(N : somme des longueurs des génomes)
    pour k de 0 à N faire
        parent[k] ← k
        rang[k] ← 0
        taille[k] ← 1
        frere[k] ← k
        largeur[k] ← 1
        precedent[k] ← k − 1
        suivant[k] ← k + 1
    precedent[0] ← N − 1                          ▷ Circulariser la liste
    suivant[N − 1] ← 0
fonction RECHERCHER(x)                            ▷ Aucun changement
    si x ≠ parent[x] alors                        ▷ pour cette fonction
        parent[x] ← RECHERCHER(parent[x])
    retourner parent[x]
fonction FUSIONNER(x, y)
    x_r ← RECHERCHER(x)
    y_r ← RECHERCHER(y)
    si x_r = y_r alors
        retourner
    si rang[x_r] < rang[y_r] alors
        échanger x_r et y_r
    si rang[x_r] = rang[y_r] alors
        rang[x_r] ← rang[x_r] + 1
    parent[y_r] ← x_r
    taille[x_r] ← taille[x_r] + taille[y_r]
    échanger frere[x_r] et frere[y_r]             ▷ Joindre les deux listes
    suivant[precedent[y_r]] ← suivant[y_r]
    precedent[suivant[y_r]] ← precedent[y_r]      ▷ y_r n'est plus un représentant
```

leurs $frere[x_r]$ et $frere[y_r]$ représentées par les deux flèches colorées suffit permet de former une liste chaînée circulaire qui parcourt les deux listes précédentes, de sorte que la nouvelle classe d'équivalence contient bien l'union des deux classes précédentes.

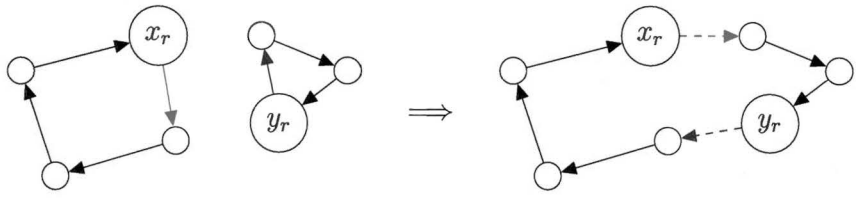

Figure 4.1 Fusion de deux classes d'équivalence dans une structure d'ensembles disjoints enrichie

L'opération DISSOCIER est détaillée à l'algorithme 4.4. Pour séparer une classe d'équivalence en singletons, on parcourt chaque position incluse dans la classe et on en fait une classe d'équivalence. Le parcours de la liste chaînée doit se faire au moyen de deux pointeurs qui se suivent, de manière à pouvoir insérer les nouvelles classes dans la liste des représentants des classes.

Pour former le graphe des colonnes, on doit examiner chaque classe pour voir si elle est valide et, si elle ne l'est pas, la dissocier. Comme indiqué dans l'algorithme 4.5, cela se fait en parcourant les classes d'équivalence. Pour chaque classe, on utilise un tableau de booléens pour garder trace des génomes impliqués dans cette classe ; s'il y a un génome qui est vu deux fois, cela veut dire que la classe est invalide et on appelle alors DISSOCIER pour la séparer en singletons.

Finalement, l'opération COLLER permet de former le graphe d'alignement en fusionnant les sommets du graphe des colonnes qui ont le même support. Les détails sont présentés à l'algorithme 4.6. On parcourt toutes les classes d'équivalence. Lorsqu'une classe x a un seul successeur y et que ces deux classes ont le même support, on les colle en ajoutant la largeur de y à la largeur de x et en retirant y

Algorithme 4.4 Séparation d'une classe d'équivalence en singletons

```
fonction DISSOCIER(x)
    x ← RECHERCHER(x)
    y ← x                                          ▷ y est la position précédente
    z ← frere[x]                                   ▷ z est la position courante
    tant que z ≠ x faire
        zsuivant ← frere[z]
        ▷ Parcourir la liste chaînée et singletonniser
        (parent[z], taille[z], frere[z], rang[z]) ← (z, 1, z, 0)
        suivant[z] ← suivant[y]                    ▷ Insérer le singleton
        precedent[z] ← y                           ▷ dans la liste des classes
        precedent[suivant[y]] ← z
        suivant[y] ← z
        y ← z                                      ▷ Passer à la position suivante
        z ← zsuivant
    (parent[x], taille[x], frere[x], rang[x]) ← (x, 1, x, 0)   ▷ Singletonniser x
```

Algorithme 4.5 Validation du graphe des colonnes en séparant les classes d'équivalence invalides en singletons

```
fonction VALIDER(n : le nombre de génomes)
    pour chaque classe d'équivalence x faire
        vu[0..n − 1] ← [faux, faux, ..., faux]
        pour chaque position p de x faire
            si vu[genome[p]] alors
                DISSOCIER(x)
                Fin boucle interne
            sinon
            vu[genome[p]] ← vrai
```

de la liste des représentants.

Si, dans un parcours des classes d'équivalences, il y a eu des classes qui ont été collées, alors on reprend le parcours une nouvelle fois, et on répète ce processus jusqu'à ce qu'il y ait un parcours n'ayant causé aucune fusion des classes, ce qui arrête le processus.

Algorithme 4.6 Formation du graphe d'alignement à partir d'un graphe des colonnes acyclique

```
    fonction COLLER
        modifie ← vrai
        tant que modifie faire
            modifie ← faux
5:          pour chaque classe d'equivalence x faire
                si TENTATIVECOLLER(x) alors
                    modifie ← vrai
    fonction TENTATIVECOLLER(x)
        y ← RECHERCHER(x + largeur[x])
10:     si taille[x] ≠ taille[y] alors
            retourner faux
        pour chaque position p de x faire
            si RECHERCHER(p + largeur[x]) ≠ y alors
                retourner faux
15:     precedent[suivant[y]] ← precedent[y]
        suivant[prececent[y]] ← suivant[y]
        largeur[x] ← largeur[x] + largeur[y]
        retourner vrai
```

Notez que le nombre de fois qu'on devra recommencer le parcours de toutes les classes d'équivalence est moins grand qu'il n'y paraît. En effet, considérons le cas extrême d'un graphe linéaire dont toutes les colonnes ont le même support. Tous les sommets de ce graphe devront être fusionnés. La première boucle va fusionner le sommet 1 avec le sommet 2, le sommet 3 avec le sommet 4, et ainsi de suite, de sorte que le nouveau graphe aura la moitié des sommets du graphe précédent. Une deuxième boucle divisera à nouveau le nombre de sommets par 2, et ainsi de suite tel qu'illustré à la figure 4.2. Le nombre de parcours est donc essentiellement

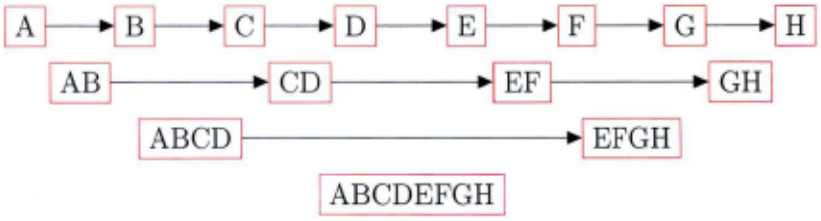

Figure 4.2 Fusions successives de sommets de même support dans un graphe des colonnes linéaire

le logarithme en base 2 du nombre de sommets.

Grâce à l'utilisation d'une structure d'ensembles disjoints enrichie, l'aligneur Alpha est en mesure de traiter des ensembles de plusieurs génomes entiers de bactériophages dans un temps raisonnable, même sur un appareil de puissance moyenne. Par exemple, il est lui possible de générer le graphe d'alignement pour un ensemble de 29 génomes de mycobactériophages, chacun d'une longueur de l'ordre de 30 000 nucléotides, en quelques minutes, ce qui rend possible son utilisation pour comparer de grands ensembles de phages.

CHAPITRE V

RÉSULTATS

Dans ce chapitre, nous montrerons des exemples d'alignements multiples générés par Alpha et nous verrons diverses façons dont Alpha peut être utile pour la compréhension des liens qui existent dans un ensemble de génomes de bactériophages.

5.1 Étude de six génomes de bactériophages

À la section 2.4 de ce mémoire, nous avons présenté un ensemble de six bactériophages dont une partie du génome a été séparée en variantes. On rappelle ici la séquence de variantes au tableau 5.1 et le graphe à la figure 5.1. Notez que Alpha utilise, dans le libellé des sommets, des lettres majuscules pour identifier les génomes. Nous avons ajouté ces lettres dans la première colonne du tableau 5.1, et pour éviter la confusion, nous utilisons des lettres minuscules pour identifier les variantes. Dans ce qui suit, une lettre majuscule représente un génome, et une lettre minuscule représente une variante d'un module.

Voyons comment Alpha peut nous aider à analyser ces six génomes et retrouver le graphe d'alignement de manière automatique.

Lorsqu'on donne à Alpha le fichier des six séquences génomiques, il commence par construire le *graphe des ancres* montré à la figure 5.2. Dans ce graphe, chaque sommet représente une séquence génomique qui est présente dans tous les génomes.

Tableau 5.1 Séquences de variantes extraites de bactériophages de *S. aureus*

Lettre	Nom	Séquence de variantes
A	Bactériophage 85	abcdefgn
B	Bactériophage 92	abdefgn
C	Bactériophage 53	ahdmfgn
D	Bactériophage 29	abclfjn
E	Bactériophage 88	abifjn
F	Bactériophage 187	akfjn

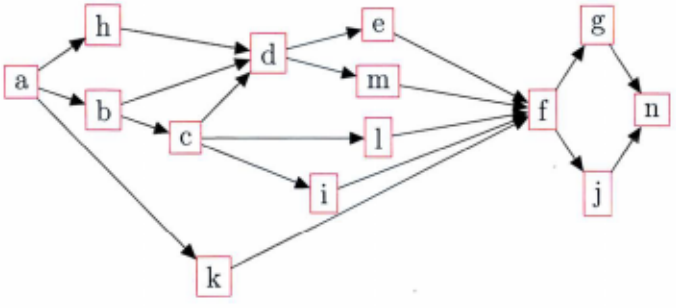

Figure 5.1 Le graphe d'alignement de six séquences extraites de bactériophages de *S. aureus*

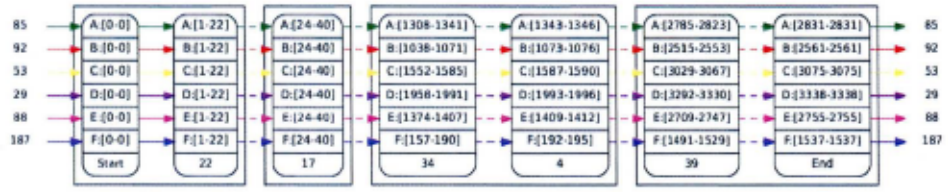

Figure 5.2 Le graphe des ancres généré par Alpha à partir d'une section de six génomes de bactériophages de *S. aureus*

À l'intérieur des sommets, les intervalles comme $A : [1 - 22]$ indiquent que la séquence est présente dans la séquence A — c'est-à-dire le génome du bactériophage 85, comme indiqué dans le tableau 5.1 —, de la position 1 à la position 22. La dernière ligne dans chaque sommet indique sa largeur, c'est-à-dire le nombre de positions qu'elle retrouve. Il y a aussi toujours deux ancres appelées *Start* et *End* à chaque extrémité, mais celles-ci ne correspondent pas à des positions dans les génomes.

Dans la figure 5.2, on peut distinguer une première ancre qui occupe les positions 1 à 40 dans les six génomes. C'est l'ancre que nous avons appelée a dans le tableau 5.1. Plus loin, une deuxième ancre couvre les positions 1308 à 1346 dans le bactériophage 85. C'est l'ancre f. Finalement, les séquences se terminent par une troisième ancre qui occupe les positions 2785 à 2831 dans le bactériophage 85. C'est l'ancre n. On voit qu'Alpha confirme l'existence des trois ancres que nous avions trouvées à la section 2.4.

Les boîtes rectangulaires autour des sommets regroupent des ancres qui sont séparées par des séquences non identiques mais qui s'alignent sans insertions ni délétions. Ici, on peut considérer que deux ancres dans une même boîte forment en réalité une seule ancre comportant, dans un ou plusieurs génomes, une mutation ponctuelle. C'est pourquoi nous avons considéré la séquence 1308–1346 dans

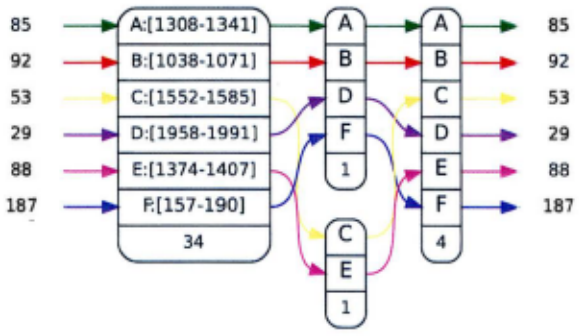

Figure 5.3 Polymorphisme détecté par Alpha dans l'ancre f

le bactériophage 85 comme une seule ancre, même si le graphe des ancres la sépare en deux séquences : 1308–1341 et 1342–1346. La raison de cette séparation est claire si on utilise Alpha pour visualiser le graphe complet correspondant aux positions 1308–1346 dans le bactériophage 85, donné à la figure 5.3. On voit que les bactériophages 53 et 88 ont un nucléotide différent des quatre autres, à la cinquième position à partir de la droite.

Comme on l'a fait pour l'ancre f, on peut sélectionner une portion du graphe des ancres et demander à Alpha de tracer le graphe d'alignement entre ces deux ancres. La figure 5.4 montre le graphe d'alignement contracté tracé par Alpha entre les ancres a et f. Ce graphe est fort intéressant puisqu'on y reconnaît exactement les variantes a à f ainsi que l et m de la figure 5.1. Les lettres correspondant aux variantes ont été ajoutées à la main pour montrer l'isomorphisme.

Dans la figure 5.4, le premier sommet à gauche représente un alignement sans insertion ni délétion. Il couvre les 53 premiers nucléotides des six bactériophages et possède un pourcentage d'identité de 96 %. En cliquant sur ce sommet, on peut le développer de manière à voir l'alignement exact, comme montré à la figure 5.5. Cela nous permet de trouver les deux positions (23 et 41) où se manifeste du

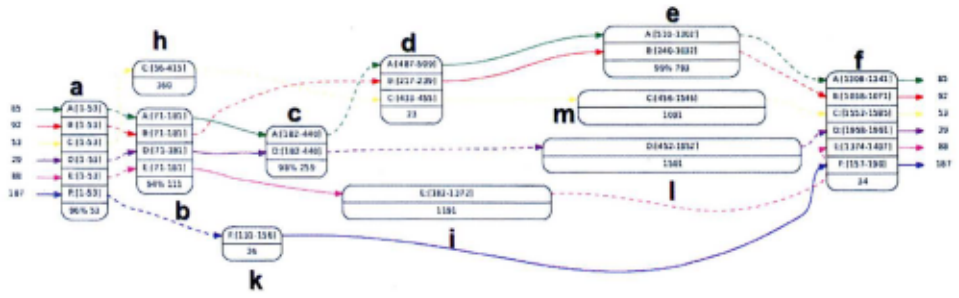

Figure 5.4 Un alignement généré par Alpha à partir d'une section de six génomes de bactériophages de *S. aureus*

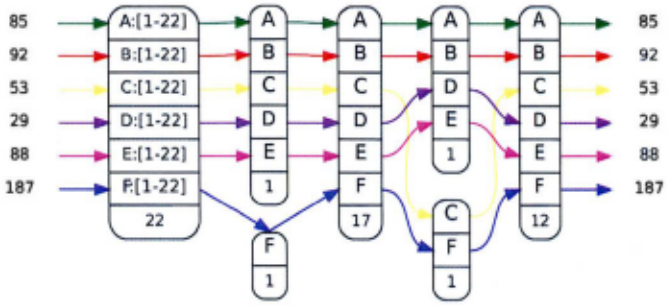

Figure 5.5 Détail des 53 premières position de l'alignement de six génomes de bactériophages de *S. aureus*

polymorphisme et de voir que les bactériophages 187 et 53 sont impliqués. Comme un tel niveau de détail est souvent excessif quand on analyse des sections plus longues que quelques dizaines de nucléotides, Alpha ne l'affiche pas par défaut.

Toujours dans la figure 5.4, on remarque que certains sommets sont reliés par des flèches pleines, d'autres par des flèches discontinues. Il s'agit là d'une autre manière de cacher certains détails moins pertinents, de manière à afficher par défaut un graphe aussi utile que possible. Une flèche discontinue indique que certains sommets de petite taille ont été omis du graphique. Il y a moyen d'afficher

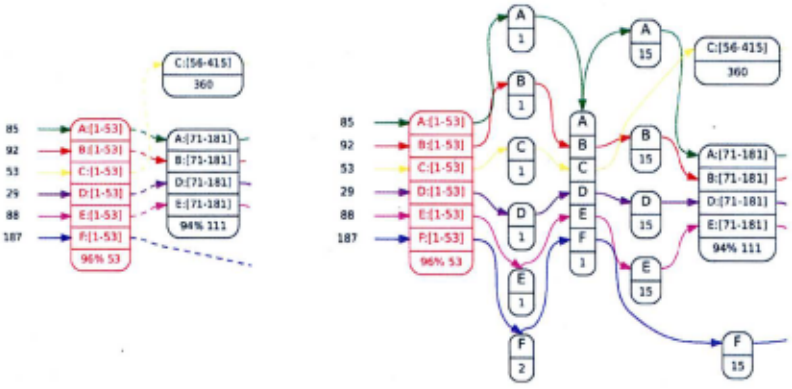

Figure 5.6 À gauche, une portion du graphe d'alignement avec les petits sommets omis. À droite, la même portion mais en incluant les petits sommets

ces sommets mais le graphe ainsi généré a souvent une allure de spaghetti et n'est pas très lisible, sauf sur de très courtes séquences, comme le montre la figure 5.6 qui contraste la partie extrême gauche du graphe contracté et sa version où tous les sommets sont inclus.

5.2 Mise en évidence de la structure en mosaïque des génomes de bactériophages

Les graphes générés par Alpha sont utiles pour illustrer la structure en mosaïque des génomes de bactériophages. La figure 5.7 montre un groupe de quatre bactériophages de *S. aureus* qui ont été analysés par Alpha. Remarquez que les positions indiquées dans les sommets du graphe montrent que le génome du bactériophage *PhiNMI* est décalé de plus de 16 000 positions par rapport aux autres, puisqu'il n'existe pas de consensus dans les banques de données quant à la position de départ de la séquence génomique. Alpha s'occupe de normaliser automatiquement les séquences qui lui sont fournies, en utilisant les ancres communes à tous les génomes.

Dans la figure 5.7, les sommets du graphe d'alignement sont très larges et correspondent à cinq modules que nous avons numérotés de 1 à 5. Les modules 1, 3 et 5 montrent des séquences de plusieurs centaines de nucléotides qui s'alignent sans insertion ni délétion avec un fort pourcentage d'identité, de 84 % à 94 %, entre les quatre génomes. Entre ces ancres, le module 2 montre que les génomes A et D partagent une variante, tandis que les génomes B et C en partage une autre. Un peu plus loin, dans le module 4, ce sont A et B qui partagent une variante et C et D qui en partagent une autre. Dans les modules pairs, les pourcentages d'identité à l'intérieur d'une même variante varient de 95 % à 100 %, tandis qu'il n'y a aucune homologie détectable entre les séquences qui sont placées dans des variantes différents. On distingue clairement la forme de transfert horizontal particulier aux génomes de bactériophages.

5.3 Le graphe des ancres

Les ancres sont des alignements qui recouvrent l'ensemble des n génomes étudiés. Ce sont des alignements exacts maximaux (Höhl *et al.*, 2002) mais ils peuvent être très courts. Même lorsqu'elles ne mesurent que quelques bases, leur construction à partir du graphe d'alignement montre que les ancres sont biologiquement signifiantes, puisque chaque ancre résulte de l'intersection d'au moins $n - 1$ alignements deux à deux de longueur m — les appariements — donc leur crédibilité augmente à la fois avec le nombre de génomes étudiés et avec le paramètre m qui

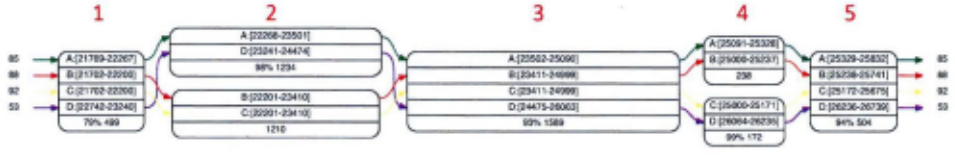

Figure 5.7 Graphe d'alignement de quatre bactériophages montrant clairement les modules et les variantes

est la longueur minimale d'un appariement. Une ancre qui contiendrait ne fût-ce qu'un seul nucléotide peut ainsi avoir une crédibilité à toute épreuve. Le graphe des ancres associé à un groupe de génomes de bactériophages forme l'épine dorsale de l'alignement de ce groupe.

Lorsqu'on veut étudier un groupe plus nombreux, il convient de commencer par le graphe des ancres, et de spécifier une valeur de m suffisamment élevée, de manière à avoir un petit nombre d'ancres dans lesquelles on peut avoir une confiance absolue. Une fois ces ancres établies, on peut ensuite étudier indépendamment la portion du génome comprise entre deux ancres consécutives, puisque les ancres forment des points d'articulation du graphe d'alignement. On peut alors diminuer la valeur de m, ou demander à Alpha de déterminer lui-même le m minimal qui garde le graphe cyclique.

À titre d'exemple, considérons un ensemble de 29 génomes de mycobactériophages dont on a gardé les 33 000 premières positions après normalisation. En utilisant Alpha avec une longueur d'appariement minimale de $m = 175$, on obtient le graphe des ancres à la figure 5.8. Étant donnée la valeur très élevée de m, il n'y a qu'un petit nombre d'ancres pour une longueur de génome de 33 000 nucléotides, mais chacune de ces ancres est justifiée par au moins 29 appariements de longueur au moins 175. Elles sont donc indéniables. En un seul graphique tenant sur un écran, on a un point de départ pour explorer un très grand ensemble de données.

À partir du graphe des ancres, on peut sélectionner deux ancres et demander à Alpha d'afficher le graphe d'alignement entre celles-ci. Si le graphe est trop large, on peut répéter l'opération en cliquant sur deux sommets du graphe, de manière à n'afficher que le sous-graphe compris entre ceux-ci. Plus la longueur diminue, plus la valeur de m peut être choisie petite. La figure 5.9 montre le résultat d'une telle opération, où on se concentre sur une région d'environ 1000 nucléotides. Dans ce

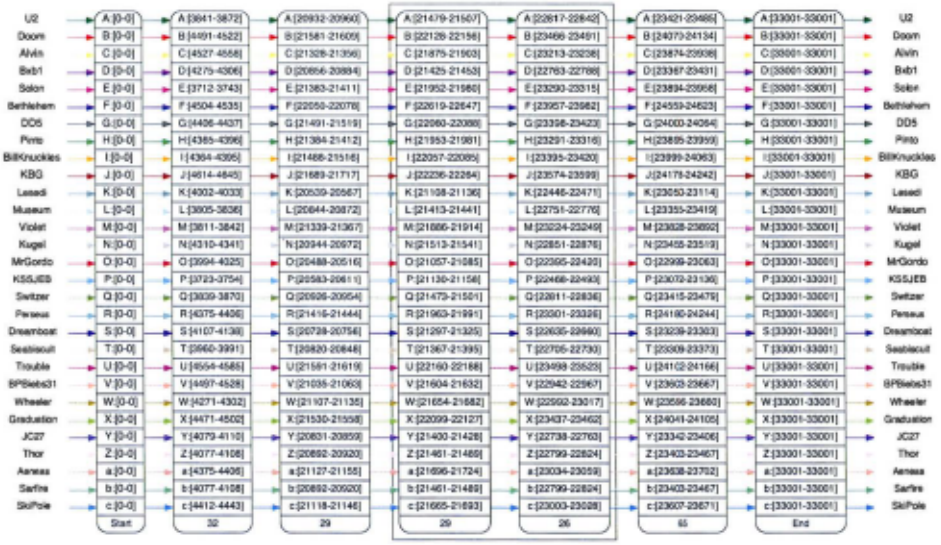

Figure 5.8 Graphe des ancres d'un ensemble de 29 mycobactériophages

graphe, la valeur de m est 15. C'est la longueur minimale qui permet de tracer le graphe, tel que déterminé par Alpha. On voit une organisation en cinq modules, avec deux variantes pour chacun des modules 2 et 4.

Une des utilités d'un tel graphe est de pouvoir facilement identifier un sous-ensemble de phages qui ont des comportements distincts, ce qui permet de diminuer le nombre de génomes à étudier simultanément. Dans l'exemple de la figure 5.9, on voit que les phages se divisent en trois catégories : ceux qui, comme A : U2, possèdent la variante du bas dans les modules 2 et 4 ; ceux qui, comme E : Solon, possèdent la variante du haut dans les modules 2 et 4, et ceux qui, comme C : Alvin, possèdent la variante du bas dans le module 2 et la variante du haut dans le module 4. Aucun phage ne possède la variante du bas dans les deux modules.

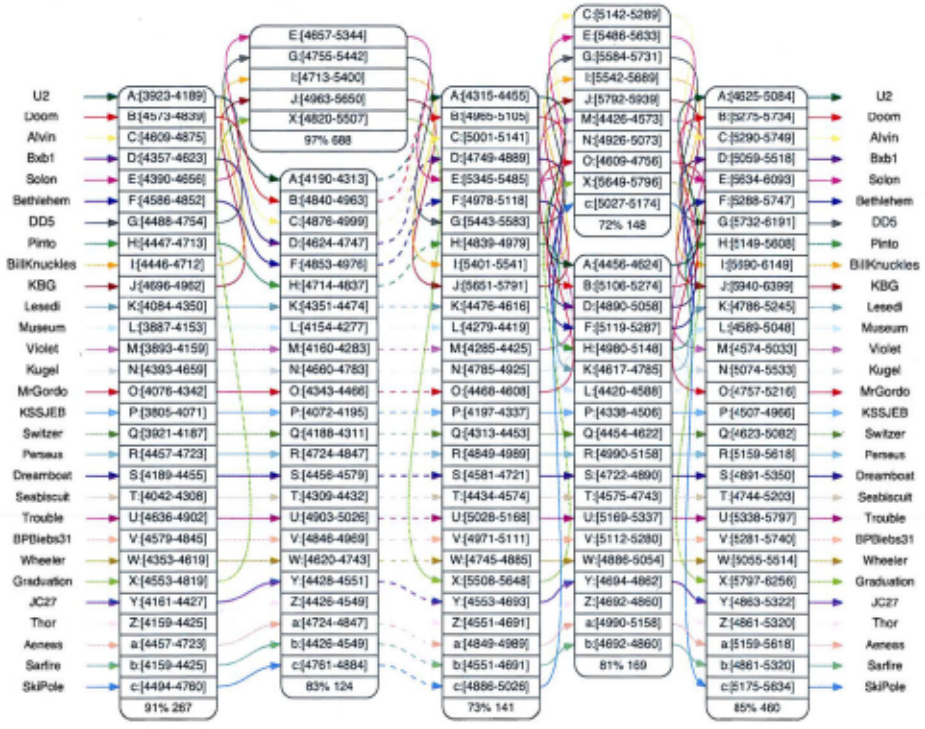

Figure 5.9 Graphe d'alignement d'une partie du génome de 29 mycobactériophages

5.4 Évaluation de la qualité des alignements

L'algorithme utilisé par Alpha est particulièrement simple. Il ne contient qu'un seul paramètre — la longueur minimale d'un appariement — et ne requiert pas de période de calibrage ou d'entraînement. Contrairement à plusieurs autres, il ne tente pas de maximiser un score mais utilise l'identité de séquence. Malgré cette simplicité, Alpha fait bonne figure lorsqu'on le compare à des outils d'alignement bien établis.

Pour évaluer la qualité des alignements générés par Alpha, trois ensembles de 4, 6 et 29 génomes de bactériophages provenant de la banque de données PhagesDB (Russell et Hatfull, 2016) ont fait l'objet d'un alignement multiple par Alpha, et les alignements ainsi générés ont été soumis à trois aligneurs : Clustal Omega (Sievers *et al.*, 2011), T-Coffee (Notredame *et al.*, 2000) et Muscle (Edgar, 2004).

Sur un total de 491 alignements sans insertion ni délétion de deux séquences ou plus trouvés par Alpha, 479 d'entre eux sont confirmés par les trois autres aligneurs. Les 12 autres ont été contestés par au moins un aligneur, qui leur a préféré des alignements contenant des insertions ou délétions. Une étude des 12 alignements contestés montre que neuf d'entre eux sont dans des régions codantes dont les protéines s'alignent sans insertion ni délétion. On peut donc rejeter les alignements contenant des insertions ou des délétions qui introduiraient des changements de cadre de lecture. Pour les trois derniers alignements contestés, il est plus difficile de trancher étant donné que souvent, les aligneurs proposent des alignements tous différents entre eux. On peut tout de même conclure que seulement 0,6 % des alignements trouvés par Alpha dans ces trois tests ne sont pas immédiatement validés.

Au-delà de l'exactitude des alignements trouvés par Alpha, on peut aussi comparer la façon dont ceux-ci sont présentés. Visuellement, les graphes interactifs générés

par Alpha sont plus faciles à interpréter que les alignements linéaires produits par d'autres aligneurs. Cela est d'autant plus vrai que le nombre de génomes augmente. On peut le constater à la figure 5.10, qui contraste la représentation générée par Alpha d'un alignement impliquant quatre génomes et celle générée par Mauve (Darling *et al.*, 2010). Dans la figure, les chiffres 1, 2, 3, 4, 5 ont été ajoutés à la main pour montrer la correspondance entre les alignements d'Alpha et ceux de Mauve. La lettre x correspond à un alignement trouvé par Mauve seulement, mais qui est à rejeter car cet alignement impliquerait de nombreux changements de cadre de lecture dans une région codante.

Le lecteur intéressé par les détails de la comparaison entre Alpha et les autres aligneurs est invité à consulter (Bérard *et al.*, 2016).

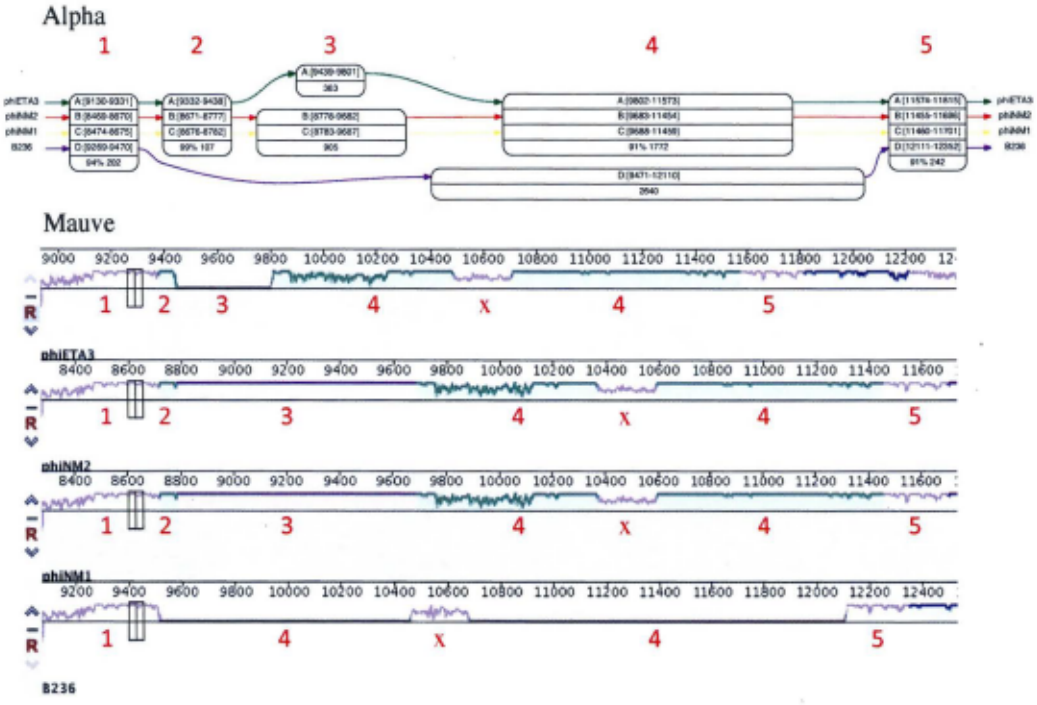

Figure 5.10 Comparaison entre les alignements de quatre génomes de bactériophages obtenus par Alpha et par Mauve pour une même région

CONCLUSION

La structure en mosaïque des génomes des bactériophages fait que la similarité fonctionnelle ne correspond pas forcément à la similarité de séquence, ce qui rend caduques les techniques traditionnelles d'alignement qui utilisent la similarité de séquence. Cependant, des caractéristiques propres aux bactériophages suggèrent d'autres techniques pour aligner leurs génomes. Ces caractéristiques sont la colinarité fonctionnelle chez les bactériophages d'une même famille, un faible taux de duplication causé par la pression d'efficacité, et la présence de longues séquences partagées par plusieurs membres d'une même famille.

Nous avons montré comment ces caractéristiques peuvent être utilisées pour obtenir un alignement multiple en utilisant une structure d'ordre partiel et des algorithmes sur les graphes qui représentent celui-ci. Ce modèle utilise l'égalité de séquence comme outil de base et il ne comporte pas de paramètre arbitraire, ce qui évite une longue étape de calibration sur les données spécifiques. Le modèle a été implémenté sous la forme d'un aligneur capable de traiter, en temps réel, des ensembles comportant des dizaines de génomes complets de bactériophages. L'aligneur permet à l'usager de visualiser l'alignement complet ou une partie de celui-ci, sous la forme de graphes interactifs faciles à manipuler à l'écran.

Nous avons pu comparer les alignements ainsi produits avec ceux donnés par d'autres aligneurs et constater que les algorithmes ainsi produits — par des algorithmes qui ne savent rien de la biologie sous-jacente — sont biologiquement significatifs.

Une application que nous n'avons pas eu le temps de traiter mais qui apparaît

naturelle est l'utilisation d'alignements multiples pour le transfert d'annotations. Certains génomes de bactériophages sont bien annotés dans les banques de données, et d'autres non. Une fois un alignement multiple complété, les variantes non annotées qui se retrouvent dans un même module que des variantes annotées peuvent être présumées correspondre à la même fonction biologique. On peut envisager un programme entièrement automatisé qui réalise les alignements et ajoute les annotations ainsi transférées aux banques de données.

Un article récent (Mai *et al.*, 2018) présente des algorithmes efficaces pour réaliser des comparaisons d'ensembles de *chaînes dégénérées* qui peuvent représenter des alignements sans insertion ni délétion. Ces algorithmes pourraient peut-être être utilisés pour augmenter l'efficacité de l'outil décrit dans ce mémoire. Dans un contexte de métagénomique où l'on ne se contente plus de comparer des génomes de phages entre eux mais où on désire comparer un ensembles de génomes à un autre ensemble de génomes, les chaînes dégénérées pourraient former la base d'une mesure de distance entre des ensembles de génomes de phages.

RÉFÉRENCES

Abouelhoda, M. I., Kurtz, S. et Ohlebusch, E. (2004). Replacing suffix trees with enhanced suffix arrays. *Journal of discrete algorithms*, *2*(1), 53–86.

Ackermann, H. W. (2011). Bacteriophage taxonomy. *Microbiology Australia*, *32*(2), 90–94.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W. et Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, *215*(3), 403–410.

Bérard, S., Chateau, A., Pompidor, N., **Guertin, P.**, Bergeron, A. et Swenson, K. M. (2016). Aligning the unalignable : bacteriophage whole genome alignments. *BMC bioinformatics*, *17*(1), 30.

Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D. *et al.* (2004). Aligning multiple genomic sequences with the threaded blockset aligner. *Genome research*, *14*(4), 708–715.

Botstein, D. (1980). A theory of modular evolution for bacteriophages. *Annals of the New York Academy of Sciences*, *354*(1), 484–491.

Cox, D. B. T., Platt, R. J. et Zhang, F. (2015). Therapeutic genome editing : prospects and challenges. *Nature medicine*, *21*(2), 121.

Darling, A. E., Mau, B. et Perna, N. T. (2010). progressivemauve : multiple genome alignment with gene gain, loss and rearrangement. *PloS one*, *5*(6), e11147.

Diallo, A. B., Makarenkov, V. et Blanchette, M. (2009). Ancestors 1.0 : a web server for ancestral sequence reconstruction. *Bioinformatics*, *26*(1), 130–131.

d'Herelle, F. *et al.* (1917). Sur un microbe invisible antagoniste des bacilles dysentériques. *CR Acad. Sci. Paris*, *165*, 373–375.

Edgar, R. C. (2004). Muscle : a multiple sequence alignment method with reduced time and space complexity. *BMC bioinformatics*, *5*(1), 113.

Fletcher, M. (2013). *RunSnakeRun*. Récupéré de `http://www.vrplumber.com/programming/runsnakerun/`

Galler, B. A. et Fisher, M. J. (1964). An improved equivalence algorithm. *Communications of the ACM*, *7*(5), 301–303.

Gremme, G., Steinbiss, S. et Kurtz, S. (2013). Genometools : a comprehensive software library for efficient processing of structured genome annotations. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *10*(3), 645–656.

Gusfield, D. (1997). *Algorithms on strings, trees and sequences : computer science and computational biology*, 143–146. Cambridge university press.

Hershey, A. D., Burgi, E. et Ingraham, L. (1963). Cohesion of dna molecules isolated from phage lambda. *Proceedings of the National Academy of Sciences*, *49*(5), 748–755.

Höhl, M., Kurtz, S. et Ohlebusch, E. (2002). Efficient multiple genome alignment. *Bioinformatics*, *18*(suppl_1), S312–S320.

Kahánková, J., Pantůček, R., Goerke, C., Růžičková, V., Holochová, P. et Doškař, J. (2010). Multilocus pcr typing strategy for differentiation of staphylococcus aureus siphoviruses reflecting their modular genome structure. *Environmental microbiology*, *12*(9), 2527–2538.

Kaiser, A. D. et Wu, R. (1968). Structure and function of dna cohesive ends. Dans *Cold Spring Harbor symposia on quantitative biology*, volume 33, 729–734. Cold Spring Harbor Laboratory Press.

Kececioglu, J. et Gusfield, D. (1998). Reconstructing a history of recombinations from a set of sequences. *Discrete Applied Mathematics*, *88*(1-3), 239–260.

Kehr, B., Trappe, K., Holtgrewe, M. et Reinert, K. (2014). Genome alignment with graph data structures : a comparison. *BMC bioinformatics*, *15*(1), 99.

Krupovic, M., Prangishvili, D., Hendrix, R. W. et Bamford, D. H. (2011). Genomics of bacterial and archaeal viruses : dynamics within the prokaryotic virosphere. *Microbiology and Molecular Biology Reviews*, *75*(4), 610–635.

Kurtz, S. et Schleiermacher, C. (1999). Reputer : fast computation of maximal repeats in complete genomes. *Bioinformatics (Oxford, England)*, *15*(5), 426–427.

Kutter, E., De Vos, D., Gvasalia, G., Alavidze, Z., Gogokhia, L., Kuhl, S. et Abedon, S. T. (2010). Phage therapy in clinical practice : treatment of human

infections. *Current pharmaceutical biotechnology*, *11*(1), 69–86.

Lee, C., Grasso, C. et Sharlow, M. F. (2002). Multiple sequence alignment using partial order graphs. *Bioinformatics*, *18*(3), 452–464.

Mai, A., Lorraine, A., Giulia, B., Grossi, R., Iliopoulos, C. S., Pisanti, N., Pissis, S. P. et Rosone, G. (2018). Degenerate string comparison and applications. Dans *18th Conference on Algorithms in Bioinformatics (WABI), 2018*. DEU.

Martinsohn, J. T., Radman, M. et Petit, M.-A. (2008). The λ red proteins promote efficient recombination between diverged sequences : implications for bacteriophage genome mosaicism. *PLoS genetics*, *4*(5), e1000065.

Mirsky, L. (1971). A dual of dilworth's decomposition theorem. *The American Mathematical Monthly*, *78*(8), 876–877.

Notredame, C., Higgins, D. G. et Heringa, J. (2000). T-coffee : a novel method for fast and accurate multiple sequence alignment1. *Journal of molecular biology*, *302*(1), 205–217.

Russell, D. A. et Hatfull, G. F. (2016). Phagesdb : the actinobacteriophage database. *Bioinformatics*, *33*(5), 784–786.

Sanger, F., Air, G. M., Barrell, B. G., Brown, N. L., Coulson, A. R., Fiddes, J. C., Hutchison III, C. A., Slocombe, P. M. et Smith, M. (1977). Nucleotide sequence of bacteriophage φx174 dna. *nature*, *265*(5596), 687.

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J. *et al.* (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, *7*(1), 539.

Swenson, K. M., **Guertin, P.**, Deschênes, H. et Bergeron, A. (2013). Reconstructing the modular recombination history of staphylococcus aureus phages. *BMC bioinformatics*, *14*(15), S17.

Tarjan, R. E. (1975). Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)*, *22*(2), 215–225.

Twort, F. W. (1915). An investigation on the nature of ultra-microscopic viruses. *The Lancet*, *186*(4814), 1241–1243.

Weinbauer, M. G. (2004). Ecology of prokaryotic viruses. *FEMS microbiology reviews*, *28*(2), 127–181.

Weisberg, R. A. et Adhya, S. (1977). Illegitimate recombination in bacteria and bacteriophage. *Annual review of genetics*, *11*(1), 451–473.

Whitman, W. B., Coleman, D. C. et Wiebe, W. J. (1998). Prokaryotes : the unseen majority. *Proceedings of the National Academy of Sciences*, *95*(12), 6578–6583.