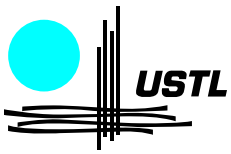


Étude de l'ordre des gènes : clusters de gènes et algorithmique des réarrangements

THÈSE

présentée et soutenue publiquement le 8 décembre 2004

pour l'obtention du

Doctorat de l'Université des Sciences et Technologies de Lille

(spécialité informatique)

par

Martin FIGEAC

Composition du jury

Président :	Jean-Marc GEIB, Professeur	Université de Lille I
Rapporteurs :	Mathieu BLANCHETTE, Assistant Professor	Mc Gill Canada
	Marie-France SAGOT, Directeur de recherche	INRIA Rhône-Alpes
Examinateurs :	Serge DULUCQ, Professeur	LABRI, Université de Bordeaux I
	Eric RIVALS, Chargé de recherche	LIRMM, Université de Montpellier II
Directeurs :	Jean-Paul DELAHAYE, Professeur	LIFL, Université de Lille I
	Jean-Stéphane VARRÉ, Maître de conférences	LIFL, Université de Lille I

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE
Laboratoire d'Informatique Fondamentale de Lille — UPRESA 8022
U.F.R. d'I.E.E.A. – Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX
Tél. : +33 (0)3 28 77 85 41 – Télécopie : +33 (0)3 28 77 85 37 – email : direction@lifl.fr

Mis en page et composé avec LaTeX et la classe thlifl.

Remerciements

Je tiens tout d'abord à remercier mes deux directeurs de thèse Jean Paul Delahaye et Jean-Stéphane Varré pour leur suivi tout au long de ces quatre années passées en leur compagnie. Ils m'ont d'abord initié à la bioinformatique en DEA et en début de Thèse puis m'ont laissé apporter ma propre expertise au domaine.

Je remercie les rapporteurs de ma thèse, Marie-France Sagot et Mathieu Blanchette pour avoir relu ma thèse, fait des remarques pertinentes et pour l'intérêt qu'ils y ont portés. Je les remercie aussi pour les conseils et perspectives enthousiastes qu'ils m'ont proposés à cette occasion.

Je remercie Serge Dulucq, examinateur de ma thèse pour avoir su se libérer de son emploi du temps très chargé afin d'assister à ma soutenance de thèse. Je remercie également Eric Rivals, examinateur de ma thèse, pour avoir lu avec autant d'intérêt mon travail.

Je remercie Jean-Marc Geib pour avoir accepter de présider ma soutenance de thèse.

Merci à toute l'équipe de bioinformatique de l'université de Lille qui ma accompagnée pendant ces quatre années. Merci aux thésards de l'équipe, Olivier, Laurent, Mathieu et Aude avec qui j'ai partagé plus que du travail. Merci aussi aux différentes personnes du LIFL avec lesquelles j'ai passé du temps et découvert la recherche universitaire.

Finalement, je voudrais remercier ma famille et plus particulièrement Émilie sans lesquelles ce travail n'aurait pu être mené à bien. Merci pour leur soutien et pour avoir cru en moi depuis bien longtemps.

Table des matières

Introduction

L'Homme a réalisé des progrès techniques extraordinaires au siècle dernier. Il est capable de voyager dans l'espace, d'envoyer des sondes et des robots sur d'autres planètes. Mais il ne connaît encore qu'une infime fraction des richesses de sa planète. Ce ne sont plus tant des territoires qu'il reste à découvrir, bien que la Terre soit en grande partie recouverte par les océans et que ceux-ci ne sont pas encore explorés, mais la richesse et la diversité des organismes vivants. Les estimations du nombre d'espèces sont impressionnantes. On peut citer par exemple les estimations du nombre d'espèces d'insectes allant de 5 millions à 30 millions (Nature du 25 avril 2002). Or chaque espèce représente un potentiel de nouvelles molécules et de biodiversité.

L'étude des espèces permet de comprendre le fonctionnement des organismes vivants. De cette compréhension et de la diversité des espèces, on peut tirer de nouveaux médicaments, découvrir de nouveaux modes de fonctionnement, inventer des nouvelles thérapies, et encore bien d'autres choses. Pour étudier et comprendre les mécanismes du vivant, on a recours à la comparaison des espèces. L'ADN (acide désoxyribonucléique) porteur de l'information génétique des espèces est au centre des mécanismes héréditaires. Toute l'information et la spécificité des organismes vivants sont codées dans leur ADN. D'une part les particularités des espèces et ce qu'elles ont en commun sont codées par l'ADN. D'autre part ce sont les mutations de l'ADN qui permettent l'évolution et la création des espèces. Pour comparer des organismes vivants, il est donc nécessaire de comparer leur ADN.

Parmi la multitude d'espèces, nous nous intéressons plus particulièrement à l'embranchement des procaryotes. Les procaryotes regroupent notamment les bactéries, avec par exemple *Yersinia pestis* qui est une des bactéries les plus pathogènes chez l'homme et est responsable de la peste. On recense à l'heure actuelle environ 5000 espèces de procaryotes et certains avancent le chiffre d'un million d'espèces.

Les génomes des procaryotes évoluent par leur taille, leur composition en nucléotides et par l'ordre de leurs gènes. La comparaison de génomes de procaryotes se fait usuellement au niveau de l'ADN : on aligne un gène d'une espèce avec ses orthologues (les gènes qui viennent d'une même origine) dans d'autres espèces. Avec le nombre croissant de génomes de procaryotes séquencés et annotés, il est théoriquement possible de comparer les génomes dans leur globalité. Cependant, dans la pratique, comparer deux génomes au niveau de leur séquence nucléique n'est pas facile. Des génomes, même évolutivement proches, peuvent être de taille et de composition différentes. La comparaison par alignement des génomes est alors très délicate.

Une méthode consiste à rechercher des gènes orthologues. Cette analyse ne prend en compte qu'une petite fraction des séquences génomiques des espèces comparées. Les gènes orthologues

évoluent dans leur composition mais aussi par leur position. La position des gènes n'est pas due au hasard, mais à des contraintes fonctionnelles. Par exemple, la régulation au niveau de la transcription est liée à la structure en opérons des génomes de procaryotes : des gènes côte à côte seront probablement transcrits ensemble et donc co-régulés. Lorsqu'au cours de l'évolution, des remaniements ont lieu et modifient la position des gènes, on parle de réarrangements génomiques. Ce mécanisme d'évolution ne modifie pas la composition des gènes mais leur nombre et leur ordre. Une des méthodes pour comparer des génomes dans leur globalité est l'étude de l'ordre de leurs gènes.

Une première approche consiste à rechercher des suites identiques de gènes dans plusieurs génomes. On suppose alors que ces gènes sont fonctionnellement liés. Cette approche permet de détecter des groupes de gènes contribuant à une fonction ou à une voie métabolique. Plus que des suites de gènes conservées dans le même ordre, les algorithmes évolués recherchent des ensembles de gènes conservés dans des intervalles de même taille. On ne regarde pas l'ordre des gènes à l'intérieur de ces intervalles. Parce que les méthodes développées reposent sur des critères mathématiques trop stricts, elles s'appliquent mal aux génomes de procaryotes [Uno and Yagiura, 2000, Heber and Stoye, 2001b, Bergeron et al., 2002b]. Ces méthodes ne prennent pas en compte les duplications et/ou ne permettent pas de comparer simultanément suffisamment de génomes. L'approche de [Rogozin et al., 2002] reprend le principe des gènes voisins dans plusieurs génomes. Elle consiste à appliquer une série de critères qui conduisent à retenir des associations de gènes. Malheureusement ces groupes de gènes sont mathématiquement mal caractérisés.

Notre approche, que nous exposons dans la première partie de ce document, donne lieu à une méthode de détection de groupe de gènes conservés. Notre méthode autorise les gènes dupliqués et la prise en compte simultanée d'un grand nombre de génomes. De plus, nous nous sommes efforcés de rendre cette méthode claire et d'offrir un modèle mathématique pour les objets détectés. La méthode présentée est spécifique aux procaryotes car elle se base directement sur les opérons. Cependant, nous proposons une extension permettant son utilisation pour toutes les espèces et notamment les eucaryotes. Ces deux travaux ont été présentés aux journées Index et Motifs [Figeac et al., 2003]. La méthode proposée a donné lieu au développement d'un logiciel nommé HUGO pour *Hierarchical Union of Genes from Operons* (union hiérarchique des gènes des opérons). Ce logiciel est disponible en téléchargement pour différentes plates-formes.

Parallèlement à l'étude des groupes de gènes conservés, une autre manière de comparer des génomes est d'étudier les réarrangements survenus entre eux. On essaye d'inférer une histoire évolutive entre ces deux génomes. La solution proposée est basée sur le principe de parcimonie. L'histoire évolutive inférée est donc celle qui suppose le moins de réarrangements. Les premiers algorithmes permettant le calcul d'un scénario évolutif minimal étaient basés sur un seul type de réarrangement et plus particulièrement les inversions de segments de gènes. Ces études ont commencé au début des années 90 et se sont surtout concentrées sur le problème algorithmique. On peut citer entre autres [Kececioglu and Sankoff, 1993, Kececioglu and Sankoff, 1995, Hannenhalli and Pevzner, 1995, Hannenhalli and Pevzner, 1996, Caprara, 1997, Bader et al., 2000, Berman et al., 2002]. Dans un deuxième temps, la communauté informatique s'est intéressée à affiner ces algorithmes pour que le modèle de départ soit plus proche des données réelles (insertion et délétion, taille des inversions, ...) [Walter et al., 1998, El-Mabrouk, 2000, Eriksen, 2001, Ajana et al., 2002,

Bergeron et al., 2002a, Dias and Meidanis, 2002]. L'échange entre la communauté informatique et la communauté biologique était relativement faible. Vers la fin des années 90, les bioinformaticiens ont commencé à regarder en détails l'ordre des gènes et les réarrangements génomiques. Il s'en est suivi une série d'articles décrivant des exemples et des statistiques sur les réarrangements [Tillier and Collins, 2000, Eisen et al., 2000, Mackiewicz et al., 2001]. Ces articles révèlent que les modèles algorithmiques proposés ne concordent pas parfaitement avec les observations. Ces dernieres laissent penser que toutes les inversions ne sont pas équiprobables, notamment que les petites inversions sont prépondérantes, ainsi que celles centrées sur l'origine de la réplication (voir figure 1).

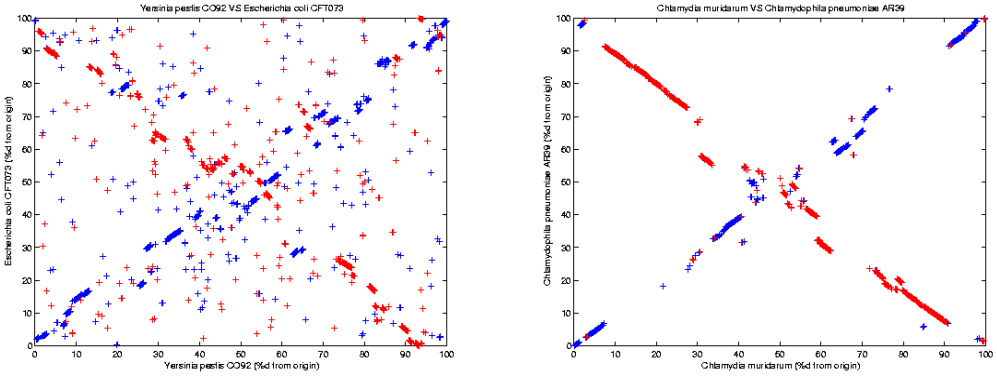


Figure 1: Représentation des positions relatives des gènes orthologues de *Yersinia pestis* contre ceux de *Escherichia coli* et ceux de *Chlamydia muridarum* contre ceux de *Chlamydophila pneumoniae*. On observe une forme de croix centrée sur l'origine de la réplication. Ce type de structure résulte d'inversions successives et centrées sur l'origine de la réplication.

Notre approche, présentée dans la deuxième partie de ce document, intègre au mieux les observations des biologistes faites sur les génomes de procaryotes dans le calcul d'un scénario évolutif minimal. Nous avons commencé par déceler des contradictions entre les scénarios évolutifs minimaux et les observations. Pour résoudre ces objections, nous contraignons les inversions autorisées pour qu'elles maintiennent toujours les groupes de gènes conservés. Notre approche semble naturelle : si des groupes de gènes sont conservés entre des espèces, un scénario évolutif passant de l'une à l'autre doit aussi les conserver. Cette approche est conforme à l'hypothèse des inversions centrées sur l'origine de la réplication. Dans cette partie nous étudions en détails ce nouveau problème algorithmique. Ce travail a fait l'objet d'une publication internationale dans les *proceedings* de la conférence WABI 2004, Lecture Notes in Computer Science, [Figeac and Varré, 2004].

Plan de lecture

Nous avons construit ce document en deux parties. La première partie présente nos travaux sur la détection de groupes de gènes conservés dans différents génomes. Dans la deuxième

partie, nous présentons nos travaux sur les réarrangements génomiques et nous montrons comment contraindre les calculs des distances évolutives basées sur les réarrangements en prenant en compte des groupes de gènes conservés.

Pour commencer, nous présentons dans **le chapitre 0** les données sur lesquelles nous travaillons. C'est-à-dire l'ADN, une macromolécule porteuse de l'information génétique des organismes vivants. Et nous détaillons plus particulièrement l'organisation des génomes de procaryotes.

Dans **la première partie** divisée en trois chapitres, nous commençons dans **le chapitre un**, par un état de l'art des méthodes de détection de groupes de gènes conservés dans plusieurs génomes. Nous avons découpé ces méthodes en deux classes : les méthodes définies formellement, trop strictes à notre sens, pour détecter des groupes de gènes dans des génomes distants ; et une méthode descriptive basée sur une suite de critères. Les méthodes formelles comprennent les Intervalles communs [Uno and Yagiura, 2000, Heber and Stoye, 2001a] et les Gene Teams [Bergeron et al., 2002b]. Ces derniers généralisent la notion de groupes de gènes mais ne permettent pas d'identifier des groupes gènes conservés à un sous ensemble des espèces. Cette méthode est donc difficile à mettre en place pour découvrir des nouveaux groupes de gènes fonctionnels. La méthode descriptive présentée est les Connected Gene Neighborhoods [Rogozin et al., 2002]. Cette méthode, bien que donnant des résultats intéressants, est tributaire de la suite de filtres pré paramétrés et limite ainsi les groupes de gènes identifiés à ces conditions expérimentales restrictives.

Dans **le chapitre deux**, nous présentons notre nouvelle méthode basée sur la définition des *über-operons* de [Lathe et al., 2000]. La méthode présentée repose sur la conservation de la régulation des groupes de gènes sous forme d'opérons entre différentes espèces. Nous exposons dans une première section le concept des über-operons pour mieux le formaliser. Cette nouvelle formalisation nous mène dans un second temps à la présentation d'un algorithme simple et efficace paru dans [Figeac et al., 2003]. Cet algorithme permet de détecter des groupes de gènes communs à un sous ensemble des espèces comparées et de prendre en compte les gènes dupliqués. Nous terminons cette première section en proposant un mode opératoire nécessaire à la bonne utilisation de l'algorithme. Nous exposons dans la deuxième section les outils nécessaires à l'expérimentation et à l'analyse des résultats. Les définitions, l'algorithmique et finalement le mode opératoire proposé permettent de modifier les conditions expérimentales et de trouver différents niveaux de groupes de gènes conservés. Finalement, dans la troisième section qui clôt ce chapitre, nous utilisons notre méthode et les outils développés pour détecter des groupes de gènes fonctionnels : détection de groupes de gènes déjà connus qui valident la méthode ; utilisation de la méthode pour détecter des nouveaux groupes de gènes qui ne peuvent être détectés par les méthodes existantes.

Le **troisième chapitre** introduit une généralisation des über-operons en *méta-clusters*. Les méta-clusters ne se basent plus sur les opérons des procaryotes mais sur les adjacences. De ce fait cette méthode s'apparente plus aux Gene Teams et possède l'avantage de s'appliquer aux eucaryotes. Nous définissons les méta-clusters et proposons un algorithme pour les détecter.

Dans **la deuxième partie**, nous nous intéressons à l'utilisation des groupes de gènes conservés pour contraindre des scénarios évolutifs. Cette partie se décompose en trois chapitres. Le **chapitre quatre** présente les travaux existants liés au calcul de scénarios évolutifs

[Kececioglu and Sankoff, 1993, Hannenhalli and Pevzner, 1995, Caprara, 1997, Christie, 1998, Bader et al., 2000, Bergeron, 2001]. Nous exposons dans une première section la problématique. Puis dans une deuxième section les travaux existants, basés sur le principe de parcimonie. Dans une troisième section, nous abordons les quelques travaux étudiant les caractéristiques et par conséquent les limites, de ces scénarios évolutifs [Ajana et al., 2002, El-Mabrouk, 2000, Cosner et al., 2000, Moret et al., 2001]. Les principaux problèmes sont d'une part que les scénarios évolutifs parcimonieux sont très nombreux et différents, et d'autre part que les distances évolutives calculées sous certaines conditions sous-estiment de beaucoup les vraies distances.

Le **chapitre cinq** présente nos travaux basés sur la constatation que les scénarios évolutifs parcimonieux cassent des groupes de gènes pour les reformer par la suite. Une partie de ces travaux sont parus dans [Figeac and Varré, 2004]. Nous contraignons les scénarios évolutifs pour qu'ils respectent les groupes de gènes conservés. La première section expose les contraintes appliquées au scénarios évolutifs. La deuxième section pose le problème et propose un algorithme le résolvant. La troisième section montre que ce problème est NP-complet. Et finalement la quatrième section étudie le diamètre de ce problème.

Dans le dernier chapitre, **chapitre six**, nous utilisons ces algorithmes pour inférer des distance évolutives. Nous utilisons pour cela des données générées par différents modèles d'évolutions que nous proposons. Une première section utilise des données générées suivant des modèles d'évolution basiques alors que la deuxième et dernière section présente des modèles d'évolution basés sur les observations des génomes de procaryotes.

Nous proposons dans ce document deux approches pour comparer l'ordre des gènes des génomes de procaryotes. La première permet de détecter des groupes de gènes conservés dans plusieurs génomes. Plus particulièrement, nous proposons une méthode permettant de comparer ensemble un grand nombre de génomes évolutivement distants. La deuxième approche infère le nombre de réarrangements génomiques séparant deux génomes. Nous utilisons le principe des groupes de gènes conservés à deux génomes pour contraindre le nombre et l'historique des réarrangements possibles. Nous nous sommes attachés dans ces deux approches à donner plus d'importance aux contraintes biologiques qu'aux contraintes algorithmiques. Nous nous sommes d'ailleurs spécialisés dans les génomes de procaryotes ce qui nous a permis d'utiliser leurs spécificités pour détecter et utiliser des informations plus rigoureuses sur le plan biologique.

chapitre 0

Evolution de l'ordre des gènes chez les procaryotes

L'ADN (acide désoxyribonucléique) est présent dans toutes les cellules de tous les organismes vivants et détermine les caractéristiques des espèces. L'ADN est le plan qui permet la fabrication des protéines dont la cellule a besoin pour se développer, et accomplir son rôle dans l'organisme. L'ADN est transmis de génération en génération lors de la reproduction ce qui le place au centre des mécanismes héréditaires. Il est le garant des particularités des espèces. Mais paradoxalement c'est son évolutivité qui permet l'évolution et la création de nouvelles espèces.

Au cours de l'évolution, des changements ponctuels comme les mutations, ou des changement globaux comme les réarrangements génomiques modifient l'ADN. La sélection naturelle permet aux espèces d'adopter ou non ces modifications. C'est pourquoi, en comparant les patrimoines génétiques d'espèces diverses, on peut voir comment les gènes ont été modifiés au cours du temps, comment se sont différenciées les espèces et à quel degré elles sont apparentées.

Nous nous intéressons à la totalité de l'ADN des espèces en considérant non pas un morceau d'ADN spécifique mais les génomes complets représentant les espèces.

Nous présentons dans ce chapitre les données sur lesquelles nous travaillons, c'est-à-dire les génomes de procaryotes et leur organisation. Les notions que nous développons dans ce chapitre sont volontairement simplifiées. Nous tentons de donner une vue globale et synthétisée des données auxquelles nous sommes confrontées. Certaines particularités ont été intentionnellement omises pour ne pas perdre le lecteur dans les détails. Dans un premier temps nous introduisons les procaryotes et leur génome. Puis nous exposons subrepticement les mécanismes de modification ponctuels de l'ADN. Et finalement, nous présentons la dynamique des génomes de procaryotes.

0.1 Organisation des génomes de procaryotes

Les organismes vivants sont classés en catégories suivant le type de leur(s) cellule(s). Les organismes constitués d'une cellule étant faite d'un seul compartiment sont des procaryotes. Ce compartiment est protégé de l'extérieur par une ou des membranes. Comme il n'y a qu'un seul compartiment, l'ADN est libre à l'intérieur de la cellule. Les procaryotes sont des organismes unicellulaires. Les procaryotes sont opposés aux eucaryotes, des organismes multi-cellulaires (Les eucaryotes comprennent notamment les mammifères). L'ADN des procaryotes est dans la majorité des cas constitué d'un seul chromosome soit linéaire, soit circulaire (c'est à dire qu'il forme un cercle). L'embranchement des procaryotes comprend entre autre les bactéries. Une des bactérie la plus étudiée est *Escherichia coli*. Certaines souches pathogènes provoquent des affections intestinales et des maladies graves chez les nourrissons. Mais d'autres souches et d'autres bactéries sont bénéfiques. Elles synthétisent certaines vitamines et par exemple aident à digérer certaines substances. Les bactéries sont les hôtes d'autres organismes. Les deux génomes sont alors présents dans la cellule.

Une cellule de procaryotes peut contenir un deuxième génome, celui d'un phage. Les phages sont des virus qui infectent les bactéries. Comme les virus, ils se reproduisent en utilisant le métabolisme de leur hôte. Le processus de vie du phage comprend deux étapes cruciales : faire éclater la cellule hôte puis aller infecter d'autres bactéries. Les phages sont spécifiques à leur hôte et n'attaquent que les bactéries. Ils sont utilisés pour détruire les bactéries infectieuses chez l'humain. Leurs utilisations ont fortement diminué avec les antibiotiques. Récemment leur utilisation est de nouveau au goût du jour étant donné la résistance croissante des bactéries aux antibiotiques.

Le fonctionnement des organismes vivants est resumé par le dogme central de la biologie moléculaire. Le dogme central, dans sa version simplifiée, repose sur trois types de molécules et sur deux actions permettant de passer d'une molécule à une autre (voir figure 0.1). Le premier type de molécules mis en jeu est l'ADN. Celle-ci se réplique et indépendamment permet la production d'un deuxième type de molécule : les ARN messager. Les ARN messager à leur tour permettent la synthèse des protéines. Ce sont principalement les protéines qui permettent le bon fonctionnement des organismes vivants.

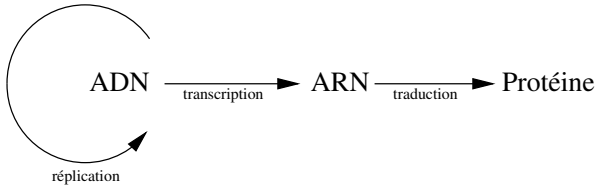


Figure 0.1: Le dogme central de la biologie moléculaire.

Premier niveau : l'ADN

L'ADN est une macromolécule formée de deux brins organisés en double hélice. Les deux brins forment des chaînes complémentaires l'une de l'autre. Les brins sont constitués d'un squelette formé alternativement d'un groupe phosphate et d'un sucre. Le sucre est lié à une

base azotée. Il existe quatre sortes de bases azotées : *l'adénine* (symbole A), la *guanine* (symbole G), la *thymine* (symbole T) et la *cytosine* (symbole C). L'ensemble formé d'une base azotée, d'un sucre et d'un groupement phosphate est appelé *nucléotide*.

Une base azotée est liée au sucre qui est lui même lié au phosphate, lesquels sont liés entre eux. Cette structure forme un des brins d'ADN. Mais ce ne sont pas les seules liaisons entres les composées chimiques formant l'ADN. Les bases azotées A et T peuvent s'appareiller, ainsi que les bases C et G, formant deux brins structurés en double hélice. Les appariements des bases azotées sont toujours les mêmes et les brins sont donc complémentaires l'un de l'autre. La structure en double brin de l'ADN est représentée figure 0.2.

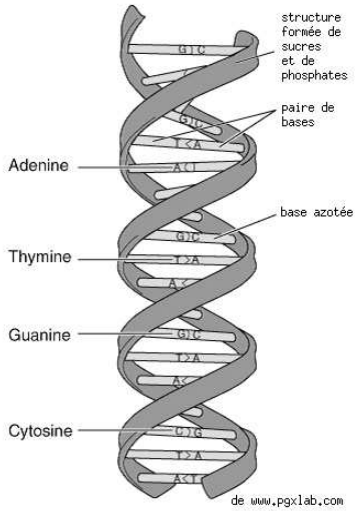


Figure 0.2: La structure en hélice des brins d'ADN

Les deux brins d'ADN sont orientés et seront lus dans un sens précis. On dit que les brins d'ADN sont lus dans le sens 5'-3'. Les extrémités des brins d'ADN n'ont pas la même composition et sont nommées 5' ou 3'. Ce sont ces deux extrémités qui donne le sens de lecture de l'ADN : de la partie 5' vers la 3'. Pour le brin complémentaire, les zones 5' et 3' sont inversées. Le brin complémentaire est lui aussi lu dans le sens 5'-3' ce qui correspond au sens inverse du premier brin.

De part la complémentarité des deux brins d'ADN et les liaisons entres les nucléotides par les phosphates, on représente souvent l'ADN comme une suite de bases azotées. L'ADN devient alors un mot sur un alphabet de quatre lettres {A,T,C,G}. Par exemple le mot ATTCGGGCCTAATA peut représenter une séquence d'ADN.

Transcription de l'ADN

L'ADN est le réservoir d'information génétique mais il n'agit pas tel quel dans la cellule. Il permet, entre autre, de synthétiser des enzymes ; mais pour ce faire, il doit d'abord être transcrit. La transcription est la copie d'une molécule d'ADN en une molécule d'ARN, l'ARN messager. L'enzyme responsable de cette transcription est l'ARN polymérase. Ce sont les gènes qui sont transcrits en ARN messager. L'ARN polymérase vient se fixer sur la région régulatrice

du gène, le promoteur, pour initier la transcription. Quand un gène est transcrit, on dit qu'il s'exprime. L'ARN polymérase, une fois fixée, se déplace dans le sens 5'-3' et synthétise un ARN à partir de la séquence d'ADN en aval de son site de fixation.

Deuxième niveau : les gènes

Alors que le terme gène est couramment utilisé, il n'existe pas de consensus pour sa définition. Le mot gène a été introduit par W. Johannsen en 1910 et désigne une hypothétique unité d'information qui contrôle la transmission héréditaire d'un trait distinctif chez un organisme particulier. La définition plus concrète de gènes que nous utiliserons est liée à l'ADN. L'ADN a été découvert dès 1869 par Friedrich Miescher mais il a fallu attendre les années 1950 pour que sa structure en double hélice soit proposée par Watson et Crick ; ce qui a permis de placer l'ADN au centre des mécanismes héréditaires. C'est Oswald Avery qui a le premier découvert que seul l'ADN était porteur de l'information génétique. Ce qui a permis de faire le lien entre les gènes et l'ADN.

Dès 1940, George Beadle et Edward Tatum ont proposé l'aphorisme "une enzyme, un gène". Ils ont ainsi associé à la définition des gènes les enzymes qu'ils produisent. Plus tard, l'aphorisme a été affiné en proposant : "un polypeptide[1], un gène". Nous savons maintenant que celui ci n'est pas plus exact que le premier car certains gènes servent au contrôle de la production de molécules d'ARN requises pour synthétiser les protéines.

Nous utiliserons la définition suivante : un gène est un segment d'ADN codant pour un (des) ARN(s) ou une (des) protéine(s). La projection d'un gène sur l'ADN est donnée figure 0.3.

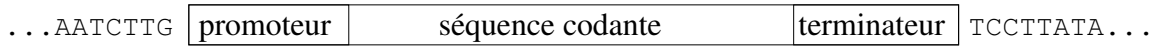


Figure 0.3: La structure du gène. Un gène est un segment d'ADN codant pour un (des) ARN(s) ou une (des) protéine(s). Il est délimité par la séquence promotrice et le terminateur.

Troisième niveau : les protéines

Les protéines servent à élaborer des structures comme la peau et les cheveux, à envoyer et recevoir des messages, à protéger l'organisme contre les corps étrangers, à transcrire l'ADN, etc. Les protéines sont au cœur de tout organisme vivant.

Les protéines sont construites à partir des ARN messagers : c'est la traduction. Au cours de la synthèse protéique, l'appareil de traduction se déplace dans le sens 5'-3' le long d'une molécule d'ARN messager. La séquence d'ARN messager est alors lue par groupes de trois nucléotides à la fois, un codon. Un ARN dit de transfert reconnaît les différents codons. Pour chaque codon, il existe un ARN de transfert différent le reconnaissant. L'ARN de transfert spécifique détermine alors l'acide aminé correspondant au codon. Il existe $4^3 = 64$ codons différents codant avec redondance pour 20 acides aminés différents. Certains codons cor-

[1] Un polypeptide est une séquence de peptides correspondant à une protéine lorsque celle-ci est représentée linéairement.

respondent au signal de début ou de fin de la traduction. Pour ces raisons, la plupart des acides aminés sont représentés par plus d'un codon, et l'on dit que le code génétique est dégénéré. Deux acides aminés, la méthionine et le tryptophane, ont un seul codon chacun, et ce sont les acides aminés les moins abondants dans les protéines. Au fur et à mesure que l'appareil de traduction se déplace le long de la molécule d'ARN messager, les différents acides aminés sont associés les uns avec les autres et forment un chaîne polypeptidique appelée protéine.

Pour agir, les protéines se lient de façon spécifique à d'autres molécules et forment ainsi des complexes. Pour les protéines intervenant dans les structures des organismes, les liaisons se font souvent entre molécules identiques. Ces protéines en s'assemblant forment un édifice volumineux comme une fibre ou la peau. Les autres protéines peuvent se lier avec des molécules de natures différentes. Prenons l'exemple de l'anticorps qui se lie avec des antigènes spécifiques afin de détecter les organismes étrangers. Un autre type de fixation jouant un rôle important pour les organismes vivants à lieu dans les mécanismes de régulation. Les protéines liées à la régulation se fixent sur des séquences d'ADN spécifiques et peuvent, par exemple, empêcher la fixation de l'ARN polymérase. Ces fixations ne sont pas stables mais évoluent au cours du temps en fonction de l'environnement des protéines. Les protéines/enzymes peuvent aussi entraîner des réactions chimiques. Une suite de réactions entre enzymes dans un but spécifique est appelée une voie métabolique.

Regroupement des gènes en opérons

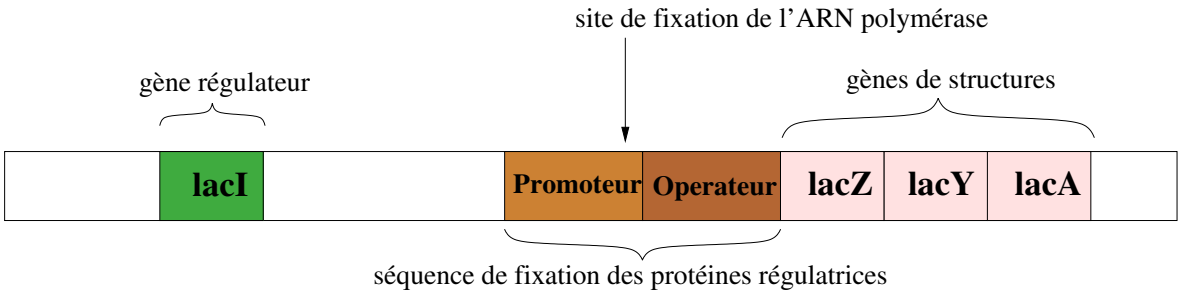
Les procaryotes, contrairement aux Eucaryotes, ont besoin de s'adapter rapidement aux conditions environnementales. Celles-ci évoluent promptement et les aliments disponibles peuvent varier d'un instant à un autre. La flexibilité est une des caractéristiques des génomes de procaryotes. La flexibilité si elle est mal gérée peut entraîner une consommation importante d'énergie des organismes. Répondre aux variations de l'environnement en utilisant des voies énergétiques coûteuses est en opposition avec la flexibilité. Le compromis bactérien est d'éviter la synthèse des enzymes d'une voie métabolique en l'absence de leur substrat. Par contre, la synthèse des enzymes doit être préparée pour qu'elle puisse commencer dès que le substrat sera présent.

On observe chez *Escherichia coli K12* que les gènes codant pour des protéines impliquées dans une même voie métabolique sont souvent régulés de manière coordonnée. On remarque la même organisation pour des gènes qui spécifient des fonctions étroitement apparentées [Singer and Berg, 1991]. L'expression de ces gènes est conjointement modulée par le même mécanisme de régulation. Ces gènes sont adjacents sur les génomes ce qui permet au groupe de gènes en entier d'être transcrit en un seul ARN messager. Cet ARN messager sera traduit de manière séquentielle en plusieurs protéines. Ces groupes de gènes co-transcrits forment une unité de transcription. Chez *Escherichia coli K12*, très peu de gènes sont individuellement transcrits [Lewin, 1987].

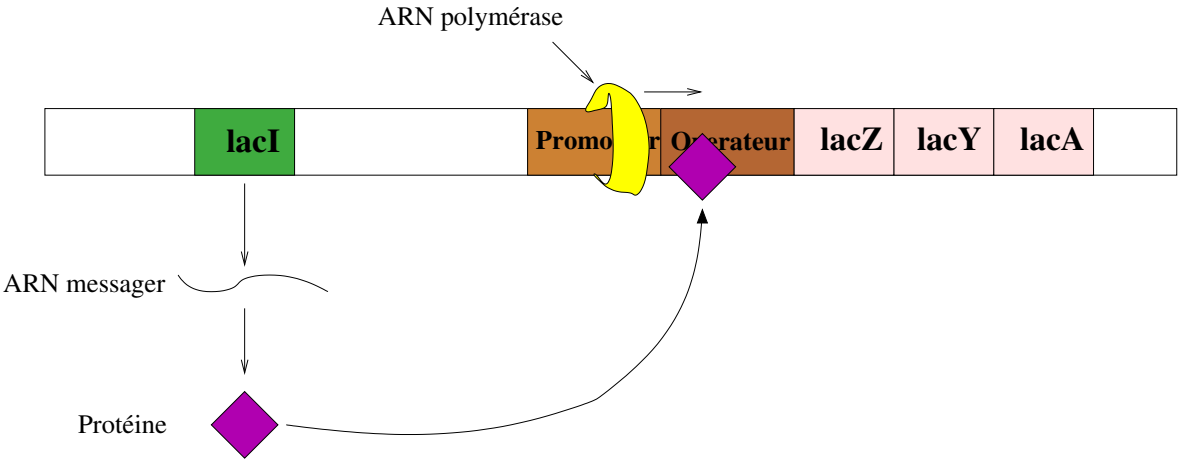
Les gènes qui codent pour des protéines requises par la cellule pour des activités enzymatiques ou des fonctions structurales sont appelés des gènes de structure. En opposition aux gènes de structure, on définit les gènes régulateurs. Les gènes codant pour des protéines qui ont pour fonction le contrôle de l'expression des gènes structuraux sont les gènes régulateurs. Les gènes régulateurs sont soit répresseurs, si la fonction du gène est d'empêcher l'expression des gènes structuraux ; soit inducteurs si leur fonction est d'activer l'expression des gènes

structuraux. En 1961, deux français, Francis Jacob et Jacques Monod, ont identifié cette caractérisation des gènes en deux groupes, ce qui les a amené à proposer une nouvelle structure : les opérons. Un opéron est une unité d'expression génétique qui inclut des gènes structuraux et les éléments contrôlant leur expression (les gènes régulateurs).

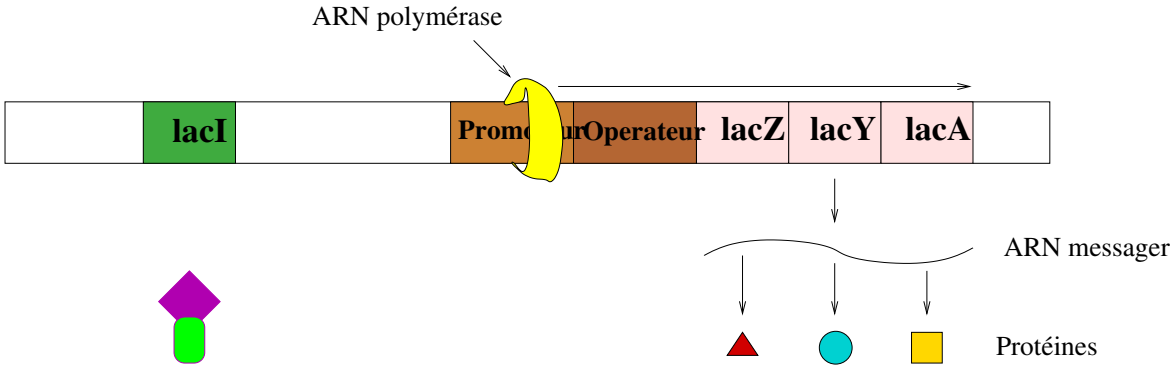
L'exemple usuel pour expliquer la structure en opéron est l'opéron lactose. La régulation de l'opéron lactose permet en cas de pénurie de glucose d'exploiter un autre sucre, le lactose. L'opéron lactose contient trois gènes voisins – lacZ, lacY, et lacA – qui servent à la dégradation du lactose. Ce sont trois gènes de structure. En amont de ces trois gènes, au niveau de l'ADN, se trouve la séquence de fixation des protéines régulatrices de leur expression. Cette région régulatrice comprend le promoteur et l'opérateur. En amont de cette zone régulatrice, se trouve le gène régulateur, lacI, qui code pour une protéine régulatrice. C'est un répresseur.



En absence de lactose, le répresseur est sous sa forme active et se lie spécifiquement au niveau de l'opérateur de l'opéron lactose en bloquant l'initiation de la transcription.



En présence de lactose, l'allolactose, un isomère du lactose, se lie au répresseur et l'inactive. Le répresseur ne peut plus se fixer sur l'opérateur et bloquer l'initiation de la transcription. La transcription a lieu normalement et l'ARN polymérase synthétise l'ARN messager qui servira à produire les protéines des gènes lacZ, lacY et lacA.



Lorsque la quantité de glucose diminue, un complexe formé de la protéine Cap et AMPc se fixe en amont du site de fixation de l'ARN polymérase et permet d'augmenter l'affinité de l'ARN polymérase pour le promoteur. La présence de ce complexe permet d'augmenter d'un facteur 50 la transcription de l'opéron lactose.

Il existe une autre notion voisine aux opérons, les régulons. Un régulon est l'ensemble des gènes régulés par un gène régulateur spécifique. Ainsi un régulon peut correspondre à plusieurs opérons mais pas l'inverse.

0.2 Mutation et réplication de l'ADN

La *réplication* permet de transmettre de génération en génération l'ADN des êtres vivants. Les brins d'ADN sont recopiés et quelques modifications peuvent apparaître. La réplication de l'ADN exige la rupture de la structure en double hélice de l'ADN via la création d'une fourche de réplication. La réplication commence à un site spécifique, l'origine de la réplication, et se poursuit soit dans une seule direction, soit dans les deux jusqu'à ce que l'ADN soit complètement dupliqué. Une fois la rupture de la double hélice commencée, chaque brin d'ADN est copié indépendamment en créant un nouveau brin complémentaire. Au final, on obtient quatre brins d'ADN imbriqués deux par deux en double hélice. Chaque brin complémentaire nouvellement synthétisé est associé avec un des anciens brins : la réplication est semi-conservative. Chez quelques organismes et notamment chez les Eucaryotes il existe plusieurs origines de réplication. La réplication se fait alors en parallèle jusqu'à ce que les fourches de duplications se rencontrent. La figure 0.4 schématise la réplication d'un génome de bactérie.

L'ADN code non seulement des fonctions enzymatiques, les mécanismes liés à sa propre réplication, mais aussi des mécanismes de réparation. La réplication de l'ADN (par exemple 4.10^3 paires de bases sont répliquées chez *Escherichia coli K12*) crée occasionnellement des erreurs dans le code génétique. La réplication n'est pas le seul phénomène altérant le code génétique, l'environnement joue aussi un rôle important. On peut citer les ultraviolets ou certaines maladies qui modifient l'ADN.

Les modifications de l'ADN sont appelées des mutations. Si des nucléotides sont omis lors de la réplication on parlera de délétions, s'ils sont insérés on parlera d'insertion, sinon ils peuvent êtres différents et on parlera de substitution. Lorsque les événements sont liés à un nucléotide spécifique, on parlera de mutation ponctuelle. Les mutations n'ayant effet que sur un des brins d'ADN entraînent des anomalies dans la structure en double hélice de l'ADN. Les sites où se produisent des mutations sont reconnus spécifiquement par des nucléases. Celles-ci

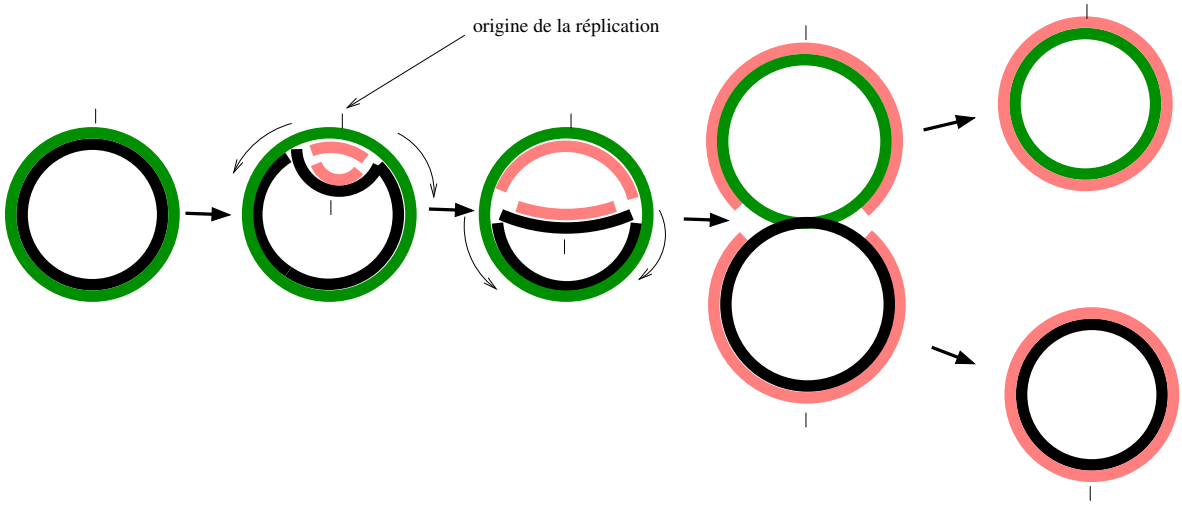


Figure 0.4: La réplication de l'ADN chez les bactéries. Une fourche de réplication débute au niveau de l'origine de la réplication : les deux brins d'ADN en double hélice se séparent. Chacun des doubles brins et dupliqué en son complémentaire. Ces nouveaux brins synthétisés reforment alors la structure en double hélice de l'ADN. Une fois la fourche de réplication arrivée à la fin de la réplication (diamétralement opposée a l'origine de la réplication), les deux brins d'ADN et leur nouveau complémentaires donne deux génomes identiques.

excisent la région endommagée de l'ADN. Puis d'autres enzymes se chargent de synthétiser la séquence manquante à l'aide du brin complémentaire.

Gènes orthologues et paralogues

Chaque organisme vivant est constitué de son propre génome, lequel est composé de gènes. Les organismes vivants sont différents les uns des autres et bien souvent deux organismes vivants d'apparence très similaire diffèrent au niveau de leur ADN. La théorie de l'évolution (Darwin, On the Origin of Species, 1859) suppose que deux espèces prises au hasard sont toujours issues d'une même espèce ancestrale. À un moment donné de leur évolution, ces deux espèces n'en ont formé qu'une seule et leur ADN a donc un passé commun. Lorsque deux séquences d'ADN sont issues d'une même origine on parlera de séquences homologues.

Une mutation modifie un gène d'un organisme et cette modification du matériel génétique peut entraîner l'apparition d'une nouvelle fonction pour ce gène. Lorsque cette modification est transmise aux descendants on dira qu'elle est fixée.

Il n'est pas possible de connaître quelles séquences sont homologues entre elles. Pour inférer des séquences homologues, on a recours à la comparaison de ces séquences en utilisant la similarité. Des séquences similaires sont probablement homologues mais pas forcément car la notion d'homologie n'a rien à voir avec la similarité, elle reflète uniquement une notion d'ancêtre commun. La similarité est une mesure expérimentale et l'hypothèse d'homologie qui en découle est une interprétation. D'ailleurs toute relation d'homologie est une interprétation.

Comparer deux génomes en comparant leurs gènes, demande de connaître quels gènes sont homologues et lesquels ne le sont pas afin de pouvoir les comparer entre eux. La notion

d'homologie entre gènes peut-être vague : la création de nouveau matériel génétique se fait notamment par la duplication des gènes, suivie de leur évolution indépendante. Après des duplications, suivies de mutations et de la fixation des nouveaux gènes, puis de nouvelles duplications de ceux-ci, les gènes issus de cette évolution sont tous homologues entre eux car ils sont issus d'un même gène ancestral. Mais l'homologie entre tous ces gènes doit être interprétée à différents niveaux.

Pour différencier les évolutions et le type d'homologie, il existe des notions plus précises : on parlera de gènes orthologues ou de gènes paralogues. Ces deux notions sont utilisées lorsqu'on s'intéresse à la comparaison de gènes et à leur histoire évolutive. Quand un gène se duplique au sein d'une espèce, il donne lieu à deux gènes qui sont paralogues : généralement ils vont donner lieu à deux fonctions distinctes. Quand un organisme ancestral se divise en deux lignées (deux espèces distinctes), chacune des lignées hérite du gène ancestral, ces deux gènes sont dits orthologues et généralement ils correspondent à la même fonction dans les deux organismes.

L'utilisation de ces deux terminologies s'explique par le type d'évolution subie par les gènes dans les différents organismes vivants : soit une évolution parallèle pour les gènes paralogues, soit une évolution orthogonale pour les gènes orthologues. Les notions d'orthologie et de paralogie sont illustrées figure 0.5.

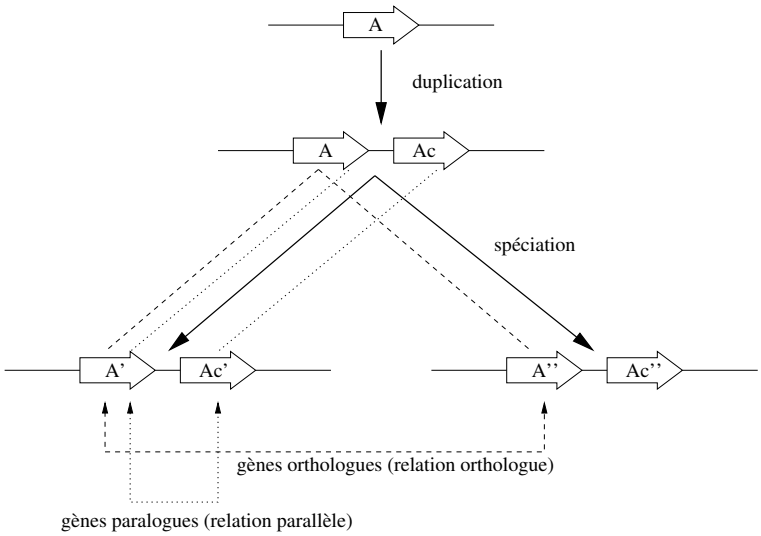


Figure 0.5: Gènes orthologues et paralogues. Deux gènes orthologues sont issus d'une évolution orthogonale ; comme les gènes A' et A". Deux gènes paralogues sont issus d'une évolution parallèle et sont le produit d'une duplication. Le gène A a été dupliqué et a donné le gène A et Ac. Puis ces deux gènes ont évolués de manière parallèle dans le génome de gauche. Le gène A à évolué en A' alors que le gène Ac a évolué en Ac' pour le génome de gauche. A' et Ac' sont des gènes paralogues.

Les familles de gènes

Il existe une autre caractérisation des relations entre les gènes : la notion de *famille de gènes*. Une famille de gènes est un ensemble de gènes dont les protéines résultantes de leur

traduction codent pour une même fonction. Une famille de gènes est donc constituée de gènes homologues ayant gardés la même fonction au cours de l'évolution. Deux gènes paralogues dont l'un a évolué pour acquérir une nouvelle fonction, ne sera pas dans la même famille de gènes.

0.3 La dynamique des génomes

L'évolution du matériel génétique des génomes n'est pas uniquement due aux mutations ponctuelles mais aussi aux recombinaisons et aux réarrangements. L'organisation des génomes n'est pas fixée : un morceau de génome peut être déplacé ou copié vers un autre endroit du même chromosome ou d'un autre. Parmi ces réarrangements, comme nous nous sommes uniquement intéressé aux génomes unichromosomal, nous exposons principalement les réarrangements traitant d'un seul chromosome. Ces réarrangements sont deux de types : ceux qui modifient le contenu des génomes et/ou ceux qui modifient l'organisation des génomes. Des événements comme les duplications, ou les transferts horizontaux permettent aux génomes d'acquérir du nouveau matériel génétique. Les inversions ou les transpositions, quand elles ne sont pas conservatives, modifient exclusivement l'organisation des génomes. Elles modifient la position des gènes les uns par rapport aux autres.

Nous présentons quelques unes des façons de modifier l'organisation des génomes. Cette présentation des remaniements génomiques n'a pas pour but d'être exhaustive et est une version simplifiée.

Modification de l'organisation des gènes

Les modifications de l'organisation des génomes sont basées sur la recombinaison. La recombinaison permet l'échange de segments d'ADN. La recombinaison implique un accolement de deux ADN double brins identiques. Puis dans un second temps les doubles brins d'ADN sont cassés et s'appareillent avec l'autre double brin d'ADN en se croisant. Ce croisement des deux brins d'ADN est appelé un *cross-over*. Le point de croisement se déplace le long des séquences identiques. Et finalement des coupures sont faites sur les brins d'ADN afin de les libérer. Ces étapes sont expliquées figure 0.6.

Insertion

Deux génomes circulaires peuvent n'en former qu'un seul par recombinaison réciproque entre deux ADN identiques. Le processus est expliqué figure 0.7. La présence d'ADN identique entre les deux génomes va permettre la recombinaison. Une fois celle-ci terminée, les deux brins d'ADN restent associés ensemble.

Délétion

Une séquence d'ADN comprise entre deux régions répétées peut être excisée par recombinaison réciproque. Le segment d'ADN excisé est alors perdu par la cellule. Le processus est expliqué figure 0.8.

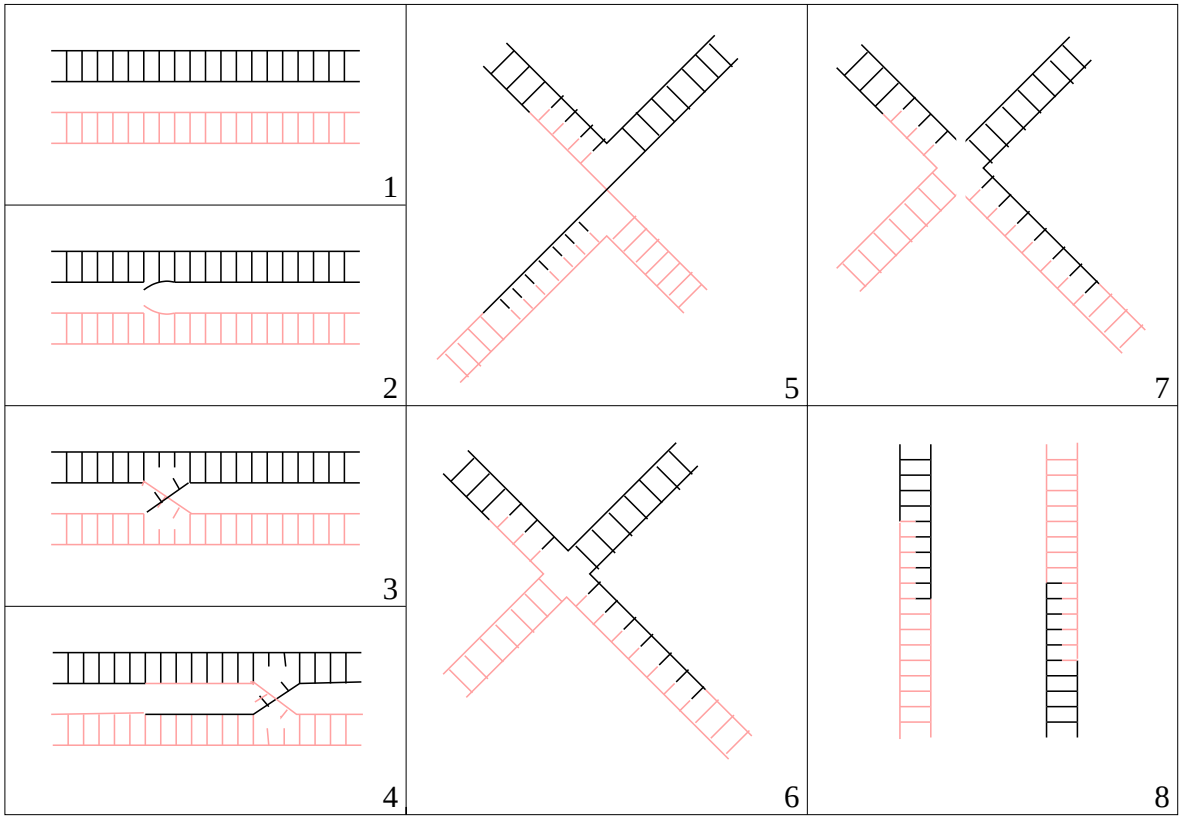


Figure 0.6: Recombinaison entre ADN identiques. Deux ADN avec chacun un segment identique (ces segments peuvent être homologues mais pas forcément). Les deux segments identiques vont se mélanger lors d'un cross-over (étapes 2 et 3). Puis, cette échange continue le long des deux brins d'ADN qui peuvent s'appareiller car ils sont identiques (étape 4). L'étape 5 représente les deux ADN avec leurs parties identiques complètement appareillées. L'étape 6 représente la même configuration que l'étape 5 mais avec une rotation de l'ADN du bas. Finalement, les brins d'ADN sont à nouveaux cassés (étape 7) pour donner naissance à deux doubles brins d'ADN mélangées.

Inversion

Une séquence d'ADN comprise entre deux régions répétées et inversées peut être retournée par recombinaison réciproque. Cette opération est nommée une inversion. Le processus est expliqué figure 0.9.

La figure 0.10 montre l'organisation de génomes de procaryotes les uns par rapport aux autres. On observe une organisation des génomes en X. Une telle structure peut être obtenue par des inversions successives autour de l'origine de la réplication. La figure 0.11 explique la construction d'une telle structure.

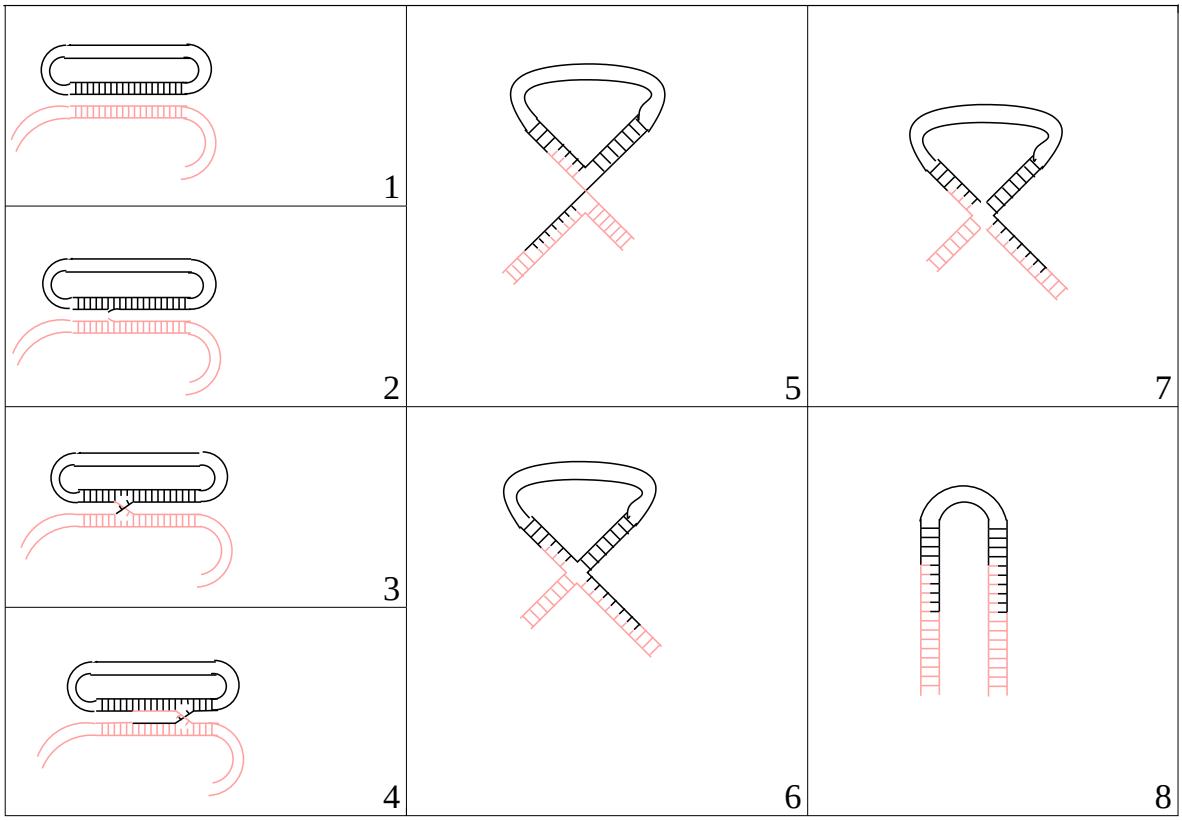


Figure 0.7: Insertion d'un segment d'ADN par recombinaison réciproque. Deux séquences d'ADN répétées et de même sens dans deux séquences peuvent s'associer entre elles ; puis par un mécanisme de recombinaison, les deux séquences d'ADN n'en forment plus qu'une.

Duplication

Une duplication correspond à la copie d'un segment d'ADN. Généralement les duplications sont en tandem, c'est-à-dire que le segment d'ADN copié est positionné juste en amont du nouveau segment d'ADN inséré. On peut ainsi observer dans les génomes des suites de segments répétés. Les duplications sont la source principale de nouvelles séquences génétiques. Une copie peut garder sa fonction tandis que l'autre évolue et produit une nouvelle fonction.

Transposition

La transposition fait référence au déplacement avec ou sans conservation d'une séquence d'ADN appelée transposons. Lorsqu'il y a conservation de la séquence d'ADN le mécanisme de transposition est équivalent à une duplication. Les transpositions permettent de dupliquer des segments d'ADN. Ces segments dupliqués peuvent permettre par la suite la recombinaison de l'ADN et favoriser les réarrangements comme l'inversion ou la délétion. Les événements évolutifs précédents n'ont pas besoin des transpositions pour avoir lieu, mais celles-ci sous certaines conditions peuvent les favoriser.

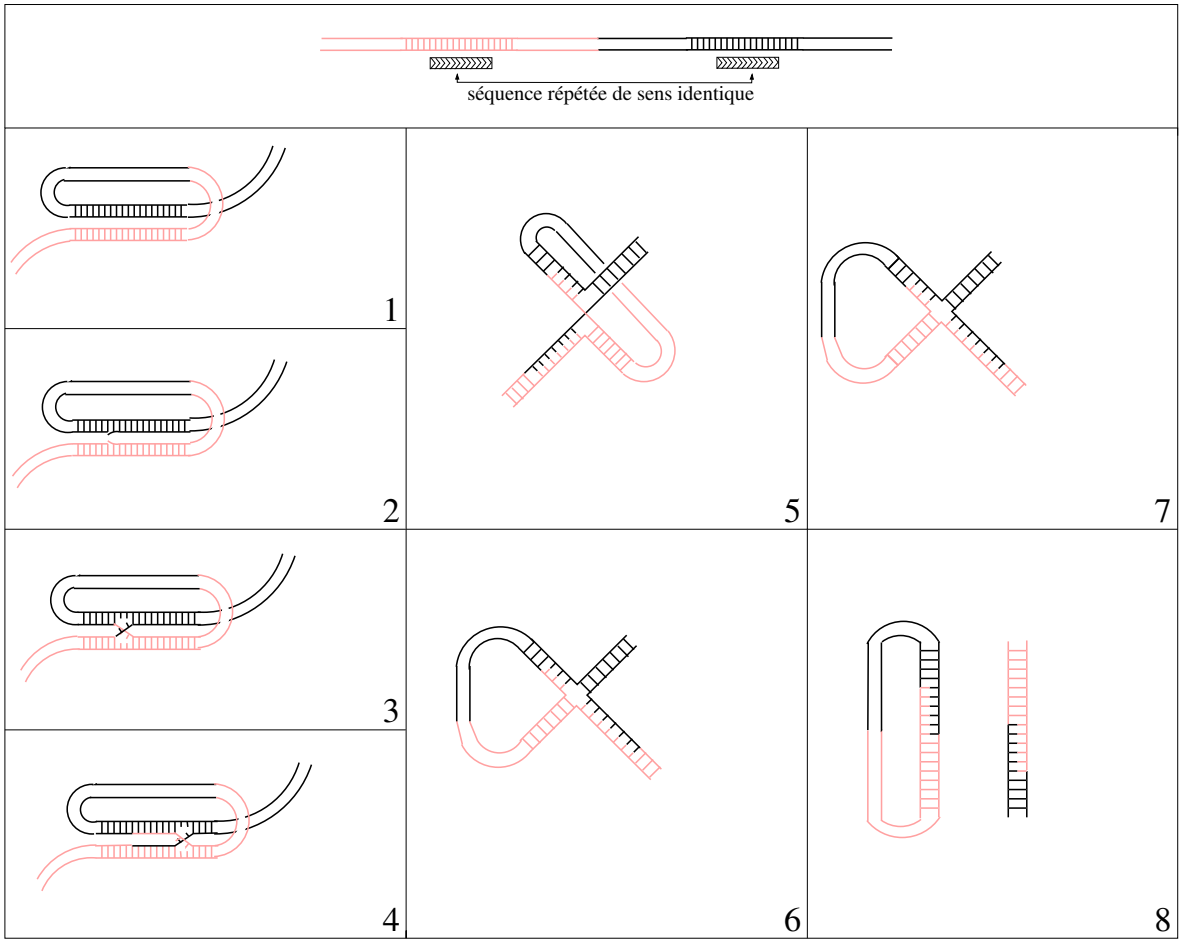


Figure 0.8: Délétion d'un segment d'ADN par recombinaison réciproque. Deux séquences d'ADN répétées et de même sens dans une même séquence peuvent s'associer entre elles ; puis par un mécanisme de recombinaison, le morceau compris entre les répétitions devient indépendant du reste de l'ADN. Cette séquence indépendante ne formant pas un génome est alors perdue par la cellule.

Transferts horizontaux

Une autre façon d'évoluer pour les procaryotes est liée aux transferts horizontaux : du matériel génétique est transféré d'un génome à un autre. On observe chez les bactéries des transferts d'information génétique entre leur génome et les génomes des plasmides.

Fission et fusion

La fission correspond à la séparation d'un chromosome en deux chromosomes distincts. La fusion est l'opération inverse et correspond à la fusion de deux chromosomes en un seul.

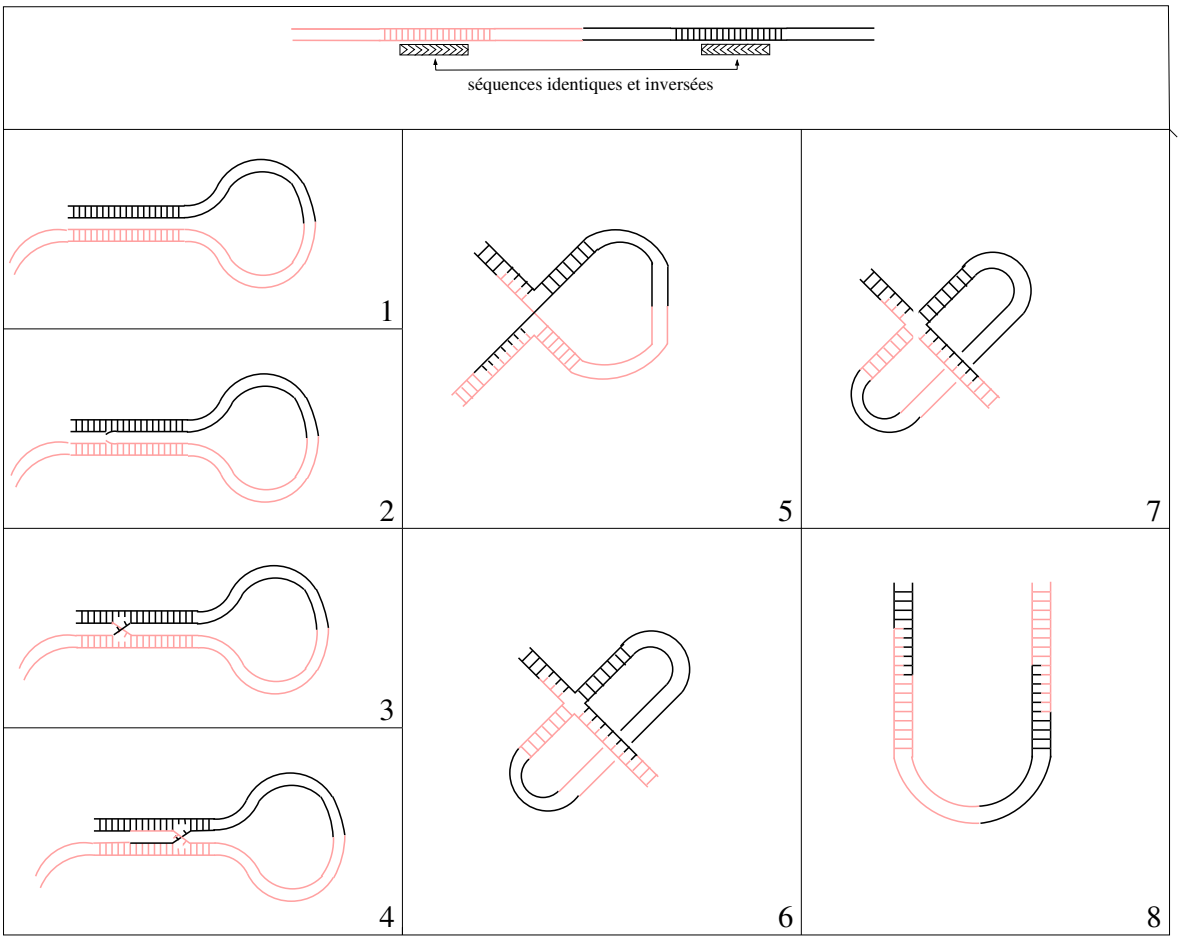


Figure 0.9: Inversion d'un segment d'ADN par recombinaison réciproque. Deux séquences d'ADN répétées mais de sens contraire dans une même séquence peuvent s'associer entre elles ; puis par un mécanisme de recombinaison, le morceau compris entre les répétitions est retourné.

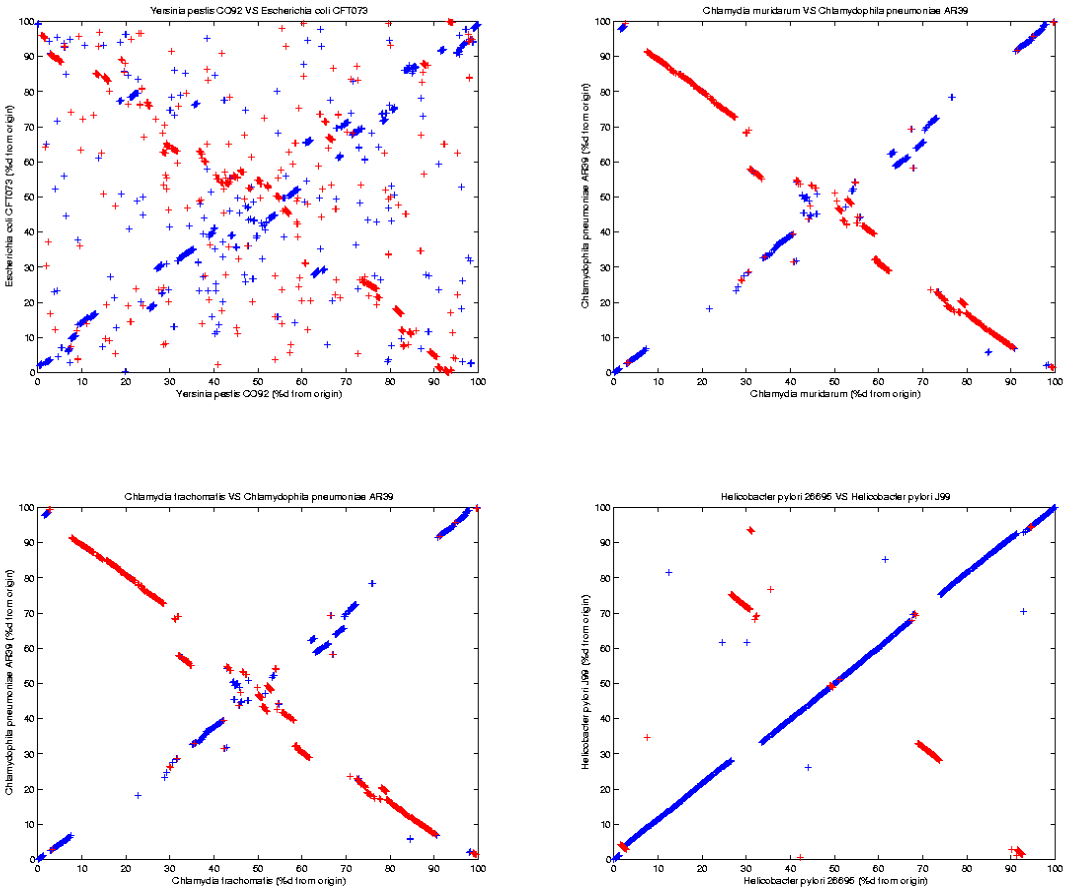


Figure 0.10: Représentation des positions relatives des gènes orthologues de *Yersinia pestis* contre ceux de *Escherichia coli*, ceux de *Chlamydia muridarum* contre ceux de *Chlamydophila pneumoniae*, ceux de *Chlamydia trachomatis* contre ceux de *Chlamydophila pneumoniae* et ceux de *Helicobacter pylori 26695* contre ceux de *Helicobacter pylori J99*. On observe une forme de croix centrée sur l'origine de la réplication. Ce type de structure résulte d'inversions successives et centrées sur l'origine de la réplication et/ou de la terminaison de la réplication.

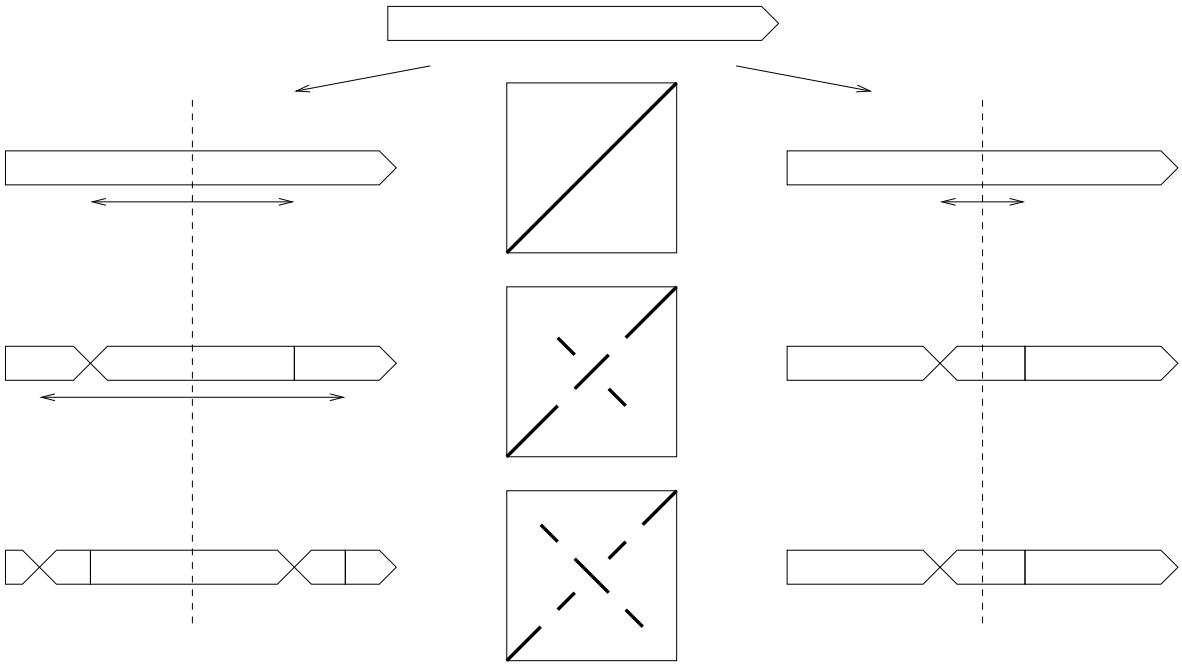


Figure 0.11: Création d'une structure en X. En haut un génome ancestral. A gauche et à droite, deux évolutions possibles de ce génome. La ligne en pointillé dénote l'origine de la réplication. À gauche le génome subit deux inversions et à droite il n'en subit qu'une seule. Les inversions sont représentées par les doubles flèches. Au milieu, on peut observer au cours de l'évolution, la représentation des positions relatives des gènes orthologues des deux génomes issus du génome ancestral. Les inversions successives centrées sur l'origine de la réplication font apparaître une structure en X.

Première Partie

Algorithmes de recherche de clusters de gènes

chapitre 1

État de l'art sur les clusters de gènes

Les réarrangements génomiques modifient le contenu et l'organisation des génomes. Ces modifications entraînent des changements fonctionnels et contribuent à la spéciation des organismes vivants. Certaines modifications sont fixées par l'évolution alors que d'autres disparaissent. Les évolutions fixées sont transmises de génération en génération et deviennent une des caractéristiques des espèces. Au cours du temps, ces modifications s'accumulent et l'organisation des gènes d'une espèce à une autre diffère d'autant plus que ces espèces sont évolutivement distantes. Deux génomes, provenant d'espèces différentes, ayant la même organisation de leurs gènes, s'ils sont suffisamment divergents, ont certainement une même contrainte fonctionnelle fixant cette organisation. Partant de ce principe, rechercher des groupes de gènes de même organisation dans différents génomes permet de prédire des contraintes fonctionnelles ou plus simplement des liens entre certains gènes. Ce sont ces groupes de gènes partageant une fonction et par conséquent souvent une même organisation que nous nommons des *clusters de gènes*.

Nous distinguons deux approches pour la détection de clusters de gènes : l'approche formelle, basée sur un modèle mathématique fort lié à une considération biologique et l'approche informelle basée sur un ensemble d'observations. La première approche définit un modèle mathématique et l'applique aux séquences génomiques. Elle donne des algorithmes très efficaces mais les contraintes rigoureuses liées à l'algorithmique et au modèle ne permettent de détecter que peu de structures conservées. La deuxième approche découvre beaucoup de clusters de gènes mais les contraintes lâches ne permettent pas d'être catégorique sur la conservation des structures décelées. Le problème de ces méthodes est la difficulté de mesurer leur fiabilité.

Dans un premier temps, nous introduisons la notion de cluster de gènes et l'utilité de les détecter. Dans un deuxième temps, nous présentons trois méthodes de détection de clusters de gènes. Nous avons classé ces méthodes en deux catégories. Les Intervalles communs et les Gene Teams pour les méthodes formelles lesquelles sont en opposition avec les Connected Gene Neighborhoods représentant les méthodes informelles (ou descriptives). Dans un troisième temps, nous présentons les inconvénients et les problèmes inhérents à ces méthodes.

1.1 La problématique de la prédiction de clusters de gènes

À chaque protéine, résultat de la traduction d'un ARN messager issu de la transcription d'un gène, est associée au moins une fonction biologique. Les fonctions des protéines ne sont pas toutes indépendantes et les protéines forment souvent des complexes ou interagissent entre elles pour composer des fonctions plus vastes et plus complètes.

On peut citer par exemple le ribosome, composé d'un assemblage complexe de protéines et d'ARN, permettant la traduction de l'ARN messager en séquences peptidiques. Les ribosomes dits 70S sont formés de deux sous-unités, une grande, la 50S, et une petite, nommée 30S. Chacune de ces sous-unités comporte des protéines et des ARN. La sous-unité 50S est constituée de deux ARN et d'au moins 34 protéines, la deuxième sous unité quant à elle contient un ARN et au moins 21 protéines.

Nous basons notre définition de clusters de gènes sur ce qu'ils représentent, c'est-à-dire un ensemble de gènes avec une ou plusieurs fonctions apparentées. Ce sont les liens entre les protéines et entre les gènes qui nous permettent de définir de la manière la plus générale les clusters de gènes. Détecter des clusters de gènes permet de prédire les fonctions des gènes, de comprendre des mécanismes d'interactions entre les protéines ou encore de prédire des mécanismes de régulation communs ou dépendants les uns des autres. Il n'est donc pas question de détecter des clusters de gènes en utilisant les fonctions associées aux gènes mais de les révéler par un autre moyen afin de prédire leurs interactions et donc leurs fonctions. Lorsqu'une nouvelle méthode de détection est proposée, elle est souvent proposée comme une nouvelle définition de clusters de gènes. Nous généralisons la notion de clusters de gènes en utilisant une et une seule définition : un ensemble de gènes avec une fonction apparentée. Il existe, par contre, plusieurs méthodes et différentes approches permettant de découvrir des clusters de gènes.

Parmi les différentes approches de détection de clusters de gènes, une méthode simple et intuitive repose sur ce qu'on appelle les profils phylogénétiques (*phylogenetic profiles*, [Pellegrini et al., 1999]) : on détecte les ensembles de gènes conjointement présents et/ou conjointement absents dans les mêmes génomes. Si lorsqu'un gène est présent, un autre l'est aussi dans les mêmes génomes, et réciproquement, alors ces deux gènes ont a priori une fonction apparentée ou contribuent au même mécanisme. Une autre méthode consiste à regarder les suites de gènes qui sont présentes dans le même ordre dans plusieurs génomes. Si aucun réarrangement génomique n'a séparé ces gènes, c'est qu'il y a probablement un bénéfice à ce qu'ils restent côte à côte. On suppose alors que ces gènes contribuent au même mécanisme. Ces gènes forment donc un cluster de gènes car ils ont une relation fonctionnelle entre eux.

Nous nous intéresserons exclusivement à la deuxième catégorie des clusters de gènes, basée sur les réarrangements génomiques. Bien que, dans la plupart des cas, l'approche fondée sur les profils phylogénétiques est implicitement inclue dans les méthodes basées sur les réarrangements.

1.2 Les algorithmes proposés

Nous présentons trois caractérisations de clusters de gènes et les algorithmes permettant la détection de telles structures. Nous avons regroupés les Intervalles communs

[Yagiura et al., 1994] et les Gene Teams [Bergeron et al., 2002b] dans une première partie décrivant les méthodes formelles. Dans un deuxième temps, nous présentons une méthode descriptive : les Connected Gene Neighborhoods [Rogozin et al., 2002].

1.2.1 La première définition formelle : les Intervalles communs

Les intervalles communs reposent sur la représentation des génomes sous forme de permutations. L'efficacité des algorithmes proposés par Yagiura, Nagamochi et Ibaraki [Yagiura et al., 1994], Uno et Yagiura [Uno and Yagiura, 2000], puis par Heber et Stoye [Heber and Stoye, 2001a, Heber and Stoye, 2001b] est due à la simplification des génomes à travers ce modèle mathématique. Cette représentation n'autorise ni les gènes dupliqués, ni les gènes insérés, ni les gènes délétés ; mais permet d'obtenir des algorithmes linéaires en fonction du nombre de gènes – communs – des génomes comparés.

Ce sont d'abord Yagiura, Nagamochi et Ibaraki qui ont les premiers étudié le problème des Intervalles communs. Dans leur étude, il n'est pas question de génomes mais d'optimiser un crossover dans un algorithme génétique dédié au problème du voyageur de commerce. Le lien avec l'étude de l'ordre des gènes est dû à Heber et Stoye un peu plus tard.

Permutations et génomes

Une *substitution* s est une application bijective d'un ensemble fini E sur lui-même. On représente communément une substitution par une correspondance de deux lignes :

$$s = \begin{pmatrix} a & b & c & d & e \\ c & a & b & e & d \end{pmatrix}$$

La première ligne est appelée ligne de référence. L'ordre des colonnes n'a pas d'importance et on peut donc réordonner la substitution pour que l'ordre de la ligne de référence soit les éléments de E suivant une relation d'ordre stricte dont E est muni. On se restreint alors à la deuxième ligne, dite ligne image, que l'on nommera une *permutation* des éléments de E. Substitution et permutation sont deux concepts différents mais on fera l'abus de langage en ne parlant que de permutations, la ligne de référence étant toujours la même.

Par exemple, en supposant l'ordre suivant sur les éléments de E : $a < b < c < d < e$, on parlera de la permutation $\pi = c \quad a \quad b \quad e \quad d$. Le i-ème élément de π sera nommé $\pi(i)$ ou π_i alors que π^i représentera une i-ème permutation. Lorsque nous discutons de plusieurs permutations, le i-ème élément de la j-ème permutation sera nommé $\pi^j(i) = \pi^j_i$.

Il existe $n!$ permutations de n éléments et ces $n!$ permutations relatives aux éléments de l'ensemble $E = \{\pi_1, \ldots, \pi_n\}$ forment le groupe symétrique S_n.

Un génome d'organisme simple peut être représenté par une liste de gènes. En effet nous avons vu que les génomes de procaryotes sont constitués d'un seul chromosome et que leurs gènes sont les uns à la suite des autres. Indépendamment du brin à partir duquel sera transcrit le génome, les gènes sont présents sur les deux brins d'ADN, l'un étant le complémentaire de l'autre. En lisant un des brins d'ADN dans le sens 5'-3' nous obtenons une liste de gènes.

Si nous comparons deux génomes, nous nous intéressons aux gènes communs à ces deux génomes, donc à la comparaison de deux listes de gènes. Si on suppose que les génomes n'ont pas de duplication, l'ordre des gènes d'un génome par rapport à l'autre forme une substitution sur E l'ensemble des n gènes communs à ces deux génomes. On travaille alors sur le groupe symétrique S_n et l'on nommera Π l'ensemble des génomes/permutations sur lesquelles on travaille. Le premier génome est symbolisé par la ligne de référence et le deuxième génome est représenté par la ligne image. De même que précédemment, l'ordre de la ligne de référence n'a pas d'importance et elle peut être remplacée par l'énumération des éléments de E étant donné un ordre sur ses éléments. Dans ce cas, la ligne de réference sera appelée *permutation identité*.

Prenons par exemple les deux listes de gènes suivantes : $l_1 = (b, a, c, f, h, g, e, d)$ et $l_2 = (g, b, h, a, d, f, e, c)$. La substitution les représentant est :

$$s = \begin{pmatrix} b & a & c & f & h & g & e & d \\ g & b & h & a & d & f & e & c \end{pmatrix} = \begin{pmatrix} a & b & c & d & e & f & g & h \\ b & g & h & c & e & a & f & d \end{pmatrix}$$

La permutation représentant ces deux listes de gènes est alors $\pi^2 = b \quad g \quad h \quad c \quad e \quad a \quad f \quad d$ et implicitement la première permutation est $\pi^1 = a \quad b \quad c \quad d \quad e \quad f \quad g \quad h$.

Définitions et algorithmes pour deux génomes

Nous nous plaçons dans le contexte où nous avons deux listes de gènes, restreintes aux gènes communs, et nous voulons identifier des groupes de gènes conservés. Nous avons vu précédemment un type de cluster : des suites de gènes conservées dans le même ordre. Nous retirons maintenant la contrainte liée à l'ordre des gènes et nous voulons identifier des groupes de gènes conservés à l'ordre près. Nous commençons par définir ces groupes de gènes nommés Intervalles communs.

Définition 1.1 *Étant donné une permutation π sur l'ensemble $E = \{1, ..., n\}$, pour tout $1 \leq x \leq y \leq n$, on note $[x, y] = \{x, x + 1, ..., y\}$ un intervalle de π.*

On notera $\pi([x, y]) = \{\pi(i) | i \in [x, y]\}$ la projection de cet intervalle sur la permutation π.

Définition 1.2 *[Yagiura et al., 1994] Pour $E = \{1, ..., n\}$, deux permutations π et σ ont un Intervalle commun nommé $c \subseteq E$ si et seulement s'il existe x, y, l et u avec $[x, y]$ et $[l, u]$ deux intervalles de π tels que*

$$c = \pi([x, y]) = \sigma([l, u])$$

Autrement dit, deux permutations ont un Intervalle commun si elles ont chacune un intervalle contenant les même éléments. D'après la définition, les intervalles de taille un peuvent être commun ; dans la plupart des cas ceux-ci seront exclus car ils portent peu d'intérêt.

Nous supposerons toujours dans la suite que la ligne de référence est donnée par la permutation identité et que toute permutation est définie par rapport à celle-ci. Cela nous permet

de spécifier un Intervalle commun en ne précisant qu'un seul intervalle. Celui-ci est définit sur la permutation courante, le deuxième intervalle lié à la permutation identité en est automatiquement déduit.

Propriété 1.1 *[Uno and Yagiura, 2000] Pour deux permutations π et σ, σ étant la permutation identité. $[x, y]$, un intervalle de π, est un Intervalle commun de π et σ si et seulement si*

$$\max(\pi([x,y])) - \min(\pi([x,y])) = y - x$$

Preuve : Nous notons $l = \min \pi([x,y])$ et $u = \max \pi([x,y])$. Montrons que

$$u - l = y - x \Longleftrightarrow \pi([x,y]) = \sigma([l,u])$$

- *Si on suppose que $\pi([x,y]) = \sigma([l,u])$, alors $\pi([x,y]) = [l,u]$ avec π une application bijective et donc $y - x = u - l$.*
- *Si on suppose que $y - x = u - l$, alors $|[x,y]| = |[l,u]|$ or $\pi([x,y]) \subseteq [l,u]$ donc $\pi([x,y]) \subseteq \sigma([l,u])$, et comme π et σ n'ont pas d'élément dupliqué, on a $\pi([x,y]) = \sigma([l,u])$*

◄

Un algorithme quadratique pour les Intervalles communs.

La propriété précédente précise qu'un couple d'indices x, y forme un Intervalle commun, $c = [x,y]$, si et seulement si on peut vérifier que $\max(\pi([x,y])) - \min(\pi([x,y])) = y - x$. Pour obtenir tous les Intervalles communs à deux séquences, il suffit de tester cette propriété pour tous les couples i, j d'éléments de E. L'algorithme BSC, de complexité $\theta(n^2)$, proposé par [Yagiura et al., 1994] en découle :

entrée : $\pi, E = \{1, ..., n\}$
début
1. **Pour chaque** $x \in \{1, ..., n-1\}$ **Faire**
2. $l := \pi_x$
3. $u := \pi_x$
4. **Pour chaque** $y \in \{x+1, ..., n\}$ **Faire**
5. $l := \min(l, \pi_y)$
6. $u := \max(l, \pi_y)$
7. **Si** $u - l = y - x$ **Alors**
8. afficher ("$[x,y]$")
9. **FinSi**
10. **Fait**
11. **Fait**

Algorithme 1: Recherche d'Intervalles communs : BSC, la méthode naïve.

Un algorithme linéaire pour les Intervalles communs.

Nous présentons maintenant un algorithme en $O(n + K)$ proposé par [Uno and Yagiura, 2000]. K est le nombre d'Intervalles communs, qui dans le pire des cas est en $O(n^2)$ mais est en $O(1)$ en pratique pour des permutations aléatoires. L'algorithme repose sur deux lemmes permettant d'éviter, quand c'est possible, la deuxième boucle de l'algorithme 1. Pour cela il détermine les intervalles $[x, y]$ dits *inutiles*, qui n'ont pas besoin d'être testés.

Lemme 1.1 *Soit $y > x > 1$, s'il existe $y' > y$ tel que $\max(\pi([x, y])) < \max(\pi([x, y']))$ et $\max(\pi([x-1, y])) = \max(\pi([x-1, y']))$, alors pour tout $x' < x$, $[x', y]$ n'est pas un Intervalle commun.*

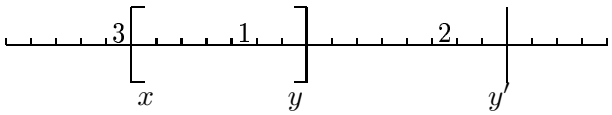


Figure 1.1: Illustration du lemme 1.1 : il n'existe pas d'Intervalle commun avec pour borne droite y et une borne gauche à gauche de x s'il existe les éléments $1 < 2 < 3$ tels que positionnés sur la figure. On ne peut pas avoir un Intervalle commun avec les éléments 1 et 3 sans l'élément 2.

Dans la suite on représentera l'équation $y - x = \max(\pi([x, y])) - \min(\pi([x, y]))$ par la fonction $f(x, y) = \max(\pi([x, y])) - \min(\pi([x, y])) - y + x$ dont on testera la nullité. Donc d'après la propriété 1.1 $[x, y]$ et $[\min(\pi([x, y])), \max(\pi([x, y]))]$ forment un Intervalle commun si et seulement si $f(x, y) = 0$.

Lemme 1.2 *S'il existe $f(x, y) > f(x, y')$ pour $y' > y > x > 1$ alors pour tout $x' < x$, $[x', y]$ n'est pas un Intervalle commun.*

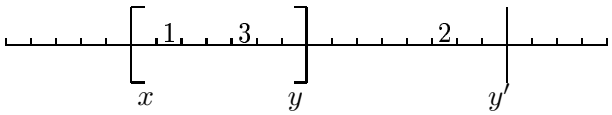


Figure 1.2: Illustration du lemme 1.2 : il n'existe pas d'Intervalle commun avec pour borne droite y et une borne gauche à gauche de x s'il existe les éléments $1 < 2 < 3$ tels que positionnés sur la figure. On ne peut pas avoir un Intervalle commun avec les éléments 1 et 3 sans l'élément 2.

Les figures 1.1 et 1.2 illustrent respectivement les lemmes 1.1 et 1.2. L'algorithme proposé ne teste pas ces Intervalles communs et permet ainsi de réduire la complexité du premier algorithme présenté. La difficulté restante est de les identifier en un temps raisonnable. L'algorithme part d'une liste de bornes droite y à tester et, à chaque étape, cette liste est réduite par l'élimination des y *inutiles* pour les étapes suivantes. L'ensemble des y inutiles éliminés est nommé I. On nommera cette liste, la liste des Y. L'algorithme présenté, proposé par Uno et Yagiura, se nomme RC pour *Reduce Candidate*. Il est détaillé par l'algorithme 2.

entrée : $\pi, E = \{1, ..., n\}$
début
1. $Y := E$
2. **Pour chaque** $x \in \{1, ..., n-1\}$ en ordre inversé **Faire**
3. **Pour chaque** $\{y \in Y | y > x\}$ **Faire**
4. **Si** $f(x, y) = 0$ **Alors**
5. afficher ("$[x, y]$")
6. **Sinon Si** $f(x, y) > 0$ **Alors**
7. break
8. **FinSi**
9. **Fait**
10. $Y := Y \setminus I$ /* $I \subseteq \{y \in E | y \geq x$ et $f(x', y) > 0$ pour tout $x' < x\}$ */
11. **Fait**

Algorithme 2: Recherche d'Intervalles communs : RC, algorithme "Reduce Candidate".

Le nombre total d'étapes nécessaires pour mettre à jour la liste des Y est linéaire en fonction du nombre d'éléments retirés. La mise à jour de la liste des Y est donc en $O(Y)$. En supposant la liste sur les Y triée, l'énumération des éléments de la liste des Y peut être stoppée dès qu'un des y vérifie $f(x, y) > 0$ car pour tout $y' > y$ on obtient $f(x, y') > 0$. Le nombre d'itération du "si" (ligne 4) est donc en $O(K)$. L'algorithme RC est donc de complexité $O(n + K)$ en temps.

Algorithmes pour une famille de génomes

Les algorithmes proposés précédemment cherchent des Intervalles communs à deux permutations. Leur origine est due à l'optimisation d'un algorithme de résolution du problème du voyageur de commerce. L'adaptation à la biologie moléculaire demande quelques raffinements et a été réalisé par Heber et Stoye [Heber and Stoye, 2001a].

Dans un premier temps nous présentons comment étendre l'algorithme RC pour chercher des clusters de gènes communs à k permutations. Dans un second temps nous exposons comment prendre en compte les génomes à plusieurs chromosomes. Puis dans un troisième temps nous verrons l'extension de l'algorithme aux permutations signées. Et finalement nous présentons les modifications nécessaires pour que les Intervalles communs prennent en compte les génomes circulaires.

Extension de deux permutations vers k permutations.

La modification la plus importante due à Heber et Stoye est la généralisation du problème de la détection d'Intervalles communs à deux permutations à la détection d'Intervalles communs à k permutations étant donné ces k permutations. Nous faisons remarquer ici que le problème de la recherche de clusters de gènes communs à k permutations étant donné n permutations, $n > k$, est différent du problème de la recherche de clusters de gènes communs à ces n permutations.

Définition 1.3 *k permutations $\pi^1, ..., \pi^k$ ont un* Intervalle commun *nommé $c \subseteq E = \{1, ..., n\}$ si et seulement s'il existe l^i et u^i pour tout i tels que $1 \leq i \leq k$ et $1 \leq l^i < u^i \leq n$ avec $c = \pi^i([l^i, u^i])$.*

Définition 1.4 *C_Π est l'ensemble des Intervalles communs de la famille de k permutations $\Pi = (\pi^1, ..., \pi^k)$. C_{Π^i} est l'ensemble des Intervalles communs de la sous famille de i permutations $(\pi^1, ..., \pi^i)$.*

L'algorithme construit les Intervalles communs à k permutations à partir des intervalles communs à $k-1$ permutations. En effet, un Intervalle est commun à k permutations s'il est commun à la première et à la k-ième permutation et qu'il appartient à $C_{\Pi^{k-1}}$, l'ensemble des Intervalles communs aux $k-1$ premières permutations.

Si on procède de cette manière, à la k-ième étape, il faut tester l'appartenance des Intervalles commun de π^k à $C_{\Pi^{k-1}}$. Or $C_{\Pi^{k-1}}$ a un nombre d'éléments en $O(n^2)$. La comparaison des Intervalles communs de π^k avec les $O(n^2)$ déjà calculé de $C_{\Pi^{k-1}}$ est trop coûteuse en temps.

L'algorithmique, proposé par Heber et Stoye, repose sur RC, l'algorithme de Uno et Yagiura, et introduit la notion d'Intervalles irréductibles. Pour limiter le nombre d'Intervalles communs à comparer, on va se limiter à un sous ensemble de ceux-ci permettant de générer les autres.

Définition 1.5 *Deux Intervalles communs c_1 et c_2 forment un* chevauchement non trivial *si $c_1 \cap c_2 \neq \emptyset$ et qu'aucun des deux n'inclut l'autre.*

Par exemple, pour la permutation $\pi = 5\ \ 6\ \ 3\ \ 4\ \ 1\ \ 2$ et la permutation identité, les deux Intervalles communs $c_1 = \pi([1,4]) = \{3,4,5,6\}$ et $c_2 = \pi([3,6]) = \{1,2,3,4\}$ forment un chevauchement non trivial. Par contre, les Intervalles communs c_1 et $c_3 = \pi([1,6]) = \{1,2,3,4,5,6\}$ se chevauchent mais ne forment pas un chevauchement non trivial.

Définition 1.6 *Une liste l d'Intervalles communs forme une* chaîne *si chaque couple successif d'intervalles communs de l forme un chevauchement non trivial.*

Par exemple, pour la permutation $\pi = 7\ \ 8\ \ 5\ \ 6\ \ 3\ \ 4\ \ 1\ \ 2$ et la permutation identité, les Intervalles communs $c_1 = \pi([1,4])$, $c_2 = \pi([3,6])$ et $c_3 = \pi([5,8])$ forment une chaîne.

Définition 1.7 *Un Intervalle commun c est dit* réductible *s'il existe une chaîne de plus d'un élément qui le génère. Sinon, il est dit* irréductible.

Par exemple, pour les trois permutations suivante : $\pi_1 = 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9$, $\pi_2 = 3\ \ 2\ \ 1\ \ 9\ \ 7\ \ 8\ \ 6\ \ 5\ \ 4$ et $\pi_3 = 4\ \ 5\ \ 6\ \ 8\ \ 7\ \ 1\ \ 2\ \ 3\ \ 9$. Les Intervalles communs et les Intervalles communs irréductibles à ces trois permutations sont :

$\pi_1 = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

$\pi_2 = 3\ 2\ 1\ 9\ 7\ 8\ 6\ 5\ 4$

$\pi_3 = 4\ 5\ 6\ 8\ 7\ 1\ 2\ 3\ 9$

Intervalles communs irréductibles $/\pi_1$		Intervalles communs réductibles $/\pi_1$
$[1,2][2,3]$	$\Rightarrow$	$[1,3]$
$[4,5][5,6]$	$\Rightarrow$	$[4,6]$
$[4,5][5,6][6,8]$	$\Rightarrow$	$[4,8]$
$[5,6][6,8]$	$\Rightarrow$	$[5,8]$

Définition 1.8 *I_Π est l'ensemble des Intervalles communs irréductibles de la famille de k permutations $\Pi = (\pi^1, ..., \pi^k)$. I_{Π^i} est l'ensemble des Intervalles communs irréductibles de la sous famille de i permutations $(\pi^1, ..., \pi^i)$.*

L'algorithme va successivement construire $I_{\Pi^{i+1}}$ à partir de I_{Π^i}. En supposant que π^1 est la permutation identité, on obtient trivialement $I_{\Pi^1} = \{[j, j+1] | 1 \leq j \leq n-1\}$. Pour chaque élément c de I_{Π^i} il faut déterminer s'il appartient à $I_{\Pi^{i+1}}$ ou s'il doit être supprimé. Étant donné I_{Π^i}, c un intervalle commun irréductible de $I_{\Pi^{i+1}}$ est tel que $c \in C_{\Pi^{i+1}}$ et c est le plus petit intervalle contenant un intervalle de I_{Π^i}.

Une fois les Intervalles communs irréductibles identifiés, on peut générer tous les Intervalles communs en temps linéaire par rapport au nombre d'Intervalles communs irréductibles, lequel est en $O(n)$. Finalement, nous obtenons l'algorithme proposé par [Heber and Stoye, 2001a], basé sur l'algorithme RC, nommé ERC pour Extended Reduce candidate. Un pseudo algorithme le décrivant est exposé par algorithme 3.

entrée : $\Pi = (\pi^1, ..., \pi^k)$
début
1. I_{Π^1} := $\{[1,2], [2,3], ..., [n-1, n]\}$
2. **Pour chaque** $i \in [2, k]$ **Faire**
3. **Pour chaque** $[x, y]$ non *inutile* **Faire**
4. **Si** $f(x, y) = 0$ **et** $\nexists [x', y'] \in I_{\Pi^{i-1}}$ tel que $[x', y'] \subseteq [x, y]$**et** $f(x', y') = 0$ **Alors**
5. I_{Π^i} := $I_{\Pi^i} \cup \{[x, y]\}$
6. **FinSi**
7. **Fait**
8. **Fait**
9. C_Π := génère (I_{Π^k})
10. afficher (C_Π)

Algorithme 3: Recherche d'Intervalles communs à k permutations, pseudo algorithme ERC, Extended Reduce Candidate

La complexité de l'algorithme ERC, qui permet d'identifier tous les Intervalles Communs à k permutations, est en $O(kn + K)$, avec K le nombre d'Intervalles communs.

Extension aux génomes à plusieurs chromosomes.

Le génome d'une espèce peut être constitué de plusieurs chromosomes. On améliore la définition des Intervalles communs en ajoutant la contrainte naturelle qu'un Intervalle commun ne peut appartenir à plusieurs chromosomes pour un même génome. La représentation des génomes sous forme de permutations ne change pas, par contre on définit la fonction CHROMOSOME qui associe à un élément, pour une permutation donnée, le chromosome sur lequel le gène correspondant est positionné :

$$\text{CHROMOSOME}_\pi : E \longrightarrow I\!N$$

Définition 1.9 *k permutations $\pi^1, ..., \pi^k$ ont un Intervalle commun nommé $c \subseteq E = \{1, ..., n\}$ si et seulement s'il existe l^i et u^i pour tout i tels que $1 \leq i \leq k$, $1 \leq l^i < u^i \leq n$ avec $c = \pi^i([l^i, u^i])$ et pour tout j tel que $l^i < j \leq u^i$ on a* $\text{CHROMOSOME}(\pi^i_{l^i}) = \text{CHROMOSOME}(\pi^i_j)$.

Soit $E = \{1, 2, 3, 4, 5, 6\}$, alors $\pi^1 = (1\ \ 2\ \ 3)(4\ \ 5\ \ 6)$ et $\pi^2 = (5\ \ 6\ \ 4\ \ 1)(3\ \ 2)$ sont des permutations représentant des génomes à chromosomes multiples et $\pi^3 = 6\ \ 4\ \ 5\ \ 3\ \ 1\ \ 2$ est une permutation représentant un génome composé d'un seul chromosome. Les Intervalles communs de π^1, π^2 et π^3 sont $c_1 = \{2, 3\}$ et $c_2 = \{4, 5, 6\}$.

Les algorithmes précédents peuvent être étendus aux permutations à chromosomes multiples en ajoutant simplement le test d'appartenance des gènes à un même chromosome. Cette modification ne change pas la complexité des algorithmes présentés [Heber and Stoye, 2001a].

Extension aux permutations signées.

Une permutation telle que spécifiée jusqu'à maintenant représente la suite de gènes d'un chromosome (ou de plusieurs dans le cas de génomes à plusieurs chromosomes). Un génome est constitué de deux brins d'ADN, le brin dit principal et son complémentaire. Les gènes sont positionnés sur l'un ou l'autre brin, et ce positionnement induit le sens de transcription des gènes (i.e. leur sens de lecture). Le brin complémentaire est une copie du brin principal à ceci près que les Amines sont remplacées par des Thymines (et réciproquement) et que les Cytosines sont remplacées par des Guanines (et réciproquement). Les gènes sont donc présents sur les deux brins d'ADN mais sous un codage différent, par contre chaque gène sera transcrit à partir d'un brin à l'exclusion de l'autre.

On représente donc un génome par une permutation signée où les éléments de la permutation représentent les gènes et où le signe d'un élément représente le brin à partir duquel il est transcrit. Nous définissons la fonction SIGNE qui pour un gène et une permutation signée, retourne le signe de celui-ci.

$$\text{SIGNE}_\pi : E \longrightarrow \{\text{-1}, +1\}$$

Nous avons vu dans le chapitre 0 page 11 que les gènes sont transcrits sous forme d'opérons, c'est à dire les uns à la suite des autres tant qu'ils sont sur le même brin d'ADN. On suppose qu'un groupe de gènes forme cluster de gènes si ses gènes sont transcrits ensemble ; il faut donc qu'ils soient au moins de même signe. Heber et Stoye proposent d'ajouter à la définition des

Intervalles commun, une contrainte liée à la transcription des gènes. Les Intervalles communs sur des permutations signées sont des intervalles contenant les mêmes gènes dans toutes les permutations et tels que, sur chaque permutation, les signes des gènes de l'intervalle soient les mêmes.

Définition 1.10 *k permutations signées $\pi^1, ..., \pi^k$ ont un intervalle commun signé nommé $c \subseteq E = \{1, ..., n\}$ si et seulement s'il existe k couples (l^1, u^i) tels que $1 \leq l^i < u^i \leq n$ avec $c = \pi^1([l^1, u^1]) = ... = \pi^k([l^k, u^k])$ et pour tout j tel que $l^i < j \leq u^i$ on a* $\text{SIGNE}(\pi^i_{l^i}) = \text{SIGNE}(\pi^i_j)$.

Pour appliquer cette nouvelle définition à l'algorithme de détection des intervalles communs, il suffit de considérer que deux gènes côte à côte, s'ils sont de signe différent, sont sur des chromosomes différents. Cette transformation peut être effectuée en temps linéaire et la complexité finale de l'algorithme ne change pas.

Extension aux permutations circulaires.

Comme nous l'avons vu dans le chapitre 0, la plupart des génomes de procaryotes sont des génomes circulaires. La représentation linéaire, sous forme de permutations, utilisée jusqu'à maintenant n'est donc pas optimale et ne permet pas toujours d'obtenir tous les Intervalles communs des génomes.

L'algorithme de détection des intervalles communs sur des permutations circulaires repose une nouvelle fois sur les résultats précédents mais nécessite l'introduction d'une nouvelle propriété :

Propriété 1.2 *Pour tout Intervalle commun c d'une famille de permutations circulaires, $\bar{c} = E \setminus c$, son complémentaire, est aussi un Intervalle commun.*

Les algorithmes précédents peuvent être modifiés de la façon suivante : on recherchera les Intervalles communs de taille au plus $\llcorner \frac{n}{2} \lrcorner$, et on déduira les autres en prenant leur complémentaire. Le test d'existence d'un Intervalle commun pour la i-ème permutation est réalisé quatre fois au lieu d'une :

- une fois avec $\pi^1([1, n])$ et $\pi^i([1, n])$;
- une fois avec $\pi^1([1, n])$ et $\pi^i([\llcorner \frac{n}{2} \lrcorner + 1, \llcorner \frac{n}{2} \lrcorner])$
- une fois avec $\pi^1([\llcorner \frac{n}{2} \lrcorner + 1, \llcorner \frac{n}{2} \lrcorner])$ et $\pi^i([1, n])$
- et une dernière fois avec $\pi^1([\llcorner \frac{n}{2} \lrcorner + 1, \llcorner \frac{n}{2} \lrcorner])$ et $\pi^i([\llcorner \frac{n}{2} \lrcorner + 1, \llcorner \frac{n}{2} \lrcorner])$

La propriété citée ci-dessus est vraie tant que l'on a pas de permutations à chromosomes multiples. Dans ce cas, le complémentaire d'un Intervalle commun n'est pas toujours lui même un Intervalle commun. En effet, tous les gènes issus du complémentaire d'un Intervalle commun ne sont pas forcément sur le même chromosome. Le problème a lieu pour les chromosomes circulaires, l'idée est donc d'ajouter des cassures artificiels permettant de linéariser les chromosomes circulaires.

On ajoute des cassures entre deux gènes adjacents de la première permutation dès que ces gènes ne sont pas dans un même chromosome pour toutes les autres permutations. On applique la même modification en comparant les adjacences des autres permutations à celles

de la première. On sait qu'il ne peut y avoir d'Intervalle commun avec une cassure ; donc en coupant un chromosome circulaire au niveau de la cassure, on obtient un chromosome linéaire sans avoir perdu d'Intervalles communs. Cette modification permet de linéariser la plupart des chromosomes circulaires. S'il ne reste que des chromosomes linéaires, le problème est résolu. Sinon, on choisit un chromosome linéaire et on note p et d ses premiers et derniers éléments. On sait alors qu'aucun Intervalle commun autre que la permutation elle même ne contient p et d. Les autres chromosomes linéaires sont traitées sans autre modification. Si on rencontre un chromosome circulaire, alors si p et d sont adjacents dans ce chromosome, on peut le linéariser en coupant entre p et d. Le problème est alors résolu et il restera à traiter séparément le cas de l'Intervalle commun comprenant tous les éléments. Par contre, si p et d ne sont pas adjacents, on traite le chromosome circulaire de la même manière que dans le cas général des chromosomes circulaires en testant les quatre coupures possibles (à gauche de p, à gauche d, à droite de p et à droite de d). On obtient alors tous les Intervalles communs sauf ceux contenant les éléments p et d. Comme p et d sont le premier et le dernier élément d'un chromosome linéaire, le seul Intervalle commun les contenant tous les deux est la permutation elle-même. Cet Intervalle commun est testé séparément.

1.2.2 Extension des Intervalles Communs : les Gene Teams

Anne Bergeron, Sylvie Corteel et Mathieu Raffinot ont travaillé sur une autre définition des groupes de gènes conservés que l'on peut voir comme une extension des Intervalles communs [Bergeron et al., 2002b, Luc et al., 2003]. La principale différence est la prise en compte des gènes non communs à tous les génomes dans la représentation des permutations. Cette différence a priori minime nécessite pourtant la modification complète de l'algorithmique.

Les Gene Teams sont plus proches des données biologiques et généralisent un peu plus la caractérisation des groupes de gènes conservés. Leur définition repose sur la notion de distance entre gènes : soit $E = \{1, ..., n\}$ un ensemble d'éléments représentant les gènes. Soit un chromosome C et la fonction POS_C qui associe à un gène donné une et une seule position dans C.

$$\text{POS}_C : E \longrightarrow \mathbb{R}$$

Cette position induit un ordre et permet de représenter un chromosome comme une liste ordonnée de gènes. Lors de la comparaison de deux chromosomes, s'ils contiennent le même ensemble de gènes alors les listes ordonnées de gènes forment une substitution et peuvent par conséquent être représentées par des permutations. Les gènes qui ne sont pas présents dans tous les chromosomes comparés seront identifiés par l'élément ★ (c'est le seul élément auquel est associé plusieurs positions dans un unique chromosome, mais comme il ne fait pas partie des gènes de E, la fonction POS ne lui est pas appliquée).

Définition 1.11 *La distance Δ_C entre deux gènes g_1 et g_2 de E sur un chromosome C est* $\Delta_C(g_1, g_2) = ||\text{POS}_C(g_2) - \text{POS}_C(g_1)||$.

Cette définition d'une distance entre deux gènes permet aux Gene Teams de généraliser la définition des groupes de gènes initiée par les Intervalles communs. La fonction position peut prendre en compte la distance réelle sur les chromosomes ou le nombre de nucléotides entre le gène d'intérêt et l'origine de la réplication, ou plus simplement le nombre de gènes

δ	Gene Teams
1	$\emptyset$
2	$\{1,2,3\},\{4,5\}$
3	$\{1,2,3,4,5\}$
4	$\{1,2,3,4,5,6\}$

Table 1.1: Les Gene Teams pour différentes valeurs de δ et avec pour définition de la position d'un gène le nombre de gènes à sa gauche. Les deux chromosomes comparés sont : $C^1 =$ 1 2 ★ 3 ★ ★ 4 5 ★ ★ 6 et $C^2 =$ 3 4 1 5 2 ★ ★ ★ 6

entre celui-ci et l'origine de la réplication. Le choix de la mesure permet d'être plus proche des contraintes biologiques. La deuxième particularité des Gene Teams est la prise en compte des gènes non communs aux chromosomes comparés, les ★.

Définition 1.12 *Pour S un sous ensemble de E, et $(g_1, \ldots, g_k)$ une liste ordonnée des éléments de S par rapport à un chromosome C. Pour $\delta > 0$, l'ensemble S est appelé une δ-chaîne de C si pour tout $1 \leq i < k$ on a $\Delta_C(g_i, g_{i+1}) \leq \delta$.*

Définition 1.13 *Un sous ensemble S de E est un δ-ensemble des chromosomes $C^1, \ldots, C^m$ si S est une δ-chaîne de C^i pour $1 \leq i \leq m$.*

Définition 1.14 *Une δ-team (ou plus généralement un Gene Team) de $C^1, \ldots, C^m$ est un δ-ensemble de $C^1, \ldots, C^m$ maximal pour l'inclusion et de taille au moins deux.*

Les δ-ensembles, maximaux pour l'inclusion, de taille 1 sont appelés des gènes orphelins.

Exemple : soit deux chromosomes C^1 et C^2.

$$C^1 = 1\ 2\ \bigstar\ 3\ \bigstar\ \bigstar\ 4\ 5\ \bigstar\ \bigstar\ 6$$

$$C^2 = 3\ 4\ 1\ 5\ 2\ \bigstar\ \bigstar\ \bigstar\ 6$$

en prenant pour définition de la position d'un gène, le nombre de gènes à sa gauche, on obtient $\text{POS}_{C^2}(5) = 3$ et $\text{POS}_{C^2}(4) = 1$. La distance entre les gènes 4 et 5 sur le chromosome C^2 est $\Delta_{C^2}(4,5) = 2$. Par exemple $\{1,3,4\}$ et $\{1,3,4,5\}$ forment des δ-chaînes de C^2 pour $\delta = 1$, $\{4,5\}$ forme une δ-chaîne de C^1 pour $\delta = 1$ et une δ-chaîne de C^2 pour $\delta = 2$. $\{1,2,3\}$ est une δ-chaîne de C^1 et de C^2 pour $\delta = 2$. $\{4,5\}$ et $\{1,2,3\}$ sont des δ-ensembles (maximales) de C^1 et de C^2 pour $\delta = 2$ et par voie de conséquence, des Gene Teams. Le tableau 1.1 donne tous les Gene Teams pour différentes valeurs de δ.

Remarque 1.1 *les chromosomes circulaires sont traités en définissant la position des gènes sur un intervalle fini et avec pour définition de la distance entre deux gènes :*

$$\Delta_C(g_1, g_2) = \min \left\{ \begin{array}{c} ||\text{POS}_C(g_2) - \text{POS}_C(g_1)|| \\ |C| - ||\text{POS}_C(g_2) - \text{POS}_C(g_1)|| \end{array} \right.$$

Contrairement aux Intervalles communs, lesquels étaient construits à partir de plus petits intervalles (les Intervalles communs irréductibles) l'algorithme des Gene Teams repose sur le fonctionnement inverse. L'algorithme est basé sur la construction des Gene Teams en partant de tous les gènes regroupés puis en scindant itérativement le ou les groupes de gènes pour arriver soit à des gènes orphelins, soit à des Gene Teams. L'ensemble de gènes qui est scindé au cours des itérations est nommé une ligue. On considère le paramètre δ fixé et on cherche tous les Gene Teams pour ce δ.

Définition 1.15 *Une ligue des chromosomes $C^1, ..., C^m$ est une union des Gene Teams des chromosomes $C^1, ..., C^m$.*

Comparaison de deux chromosomes C^1 et C^2.
L'algorithme repose sur une approche diviser pour régner. La première étape est la sélection d'une ligue S de C^1. Puis il faut vérifier sa présence sur C^2 : soit elle existe sur C^2 et forme ainsi un Gene Team, soit elle n'est pas présente sur C^2 et doit être scindée en deux sous problèmes indépendants. On divise le problème en deux sous problèmes : l'un avec les gènes de S, l'autre sans. La deuxième étape est l'appel récursif sur les deux sous problèmes indépendants.

La sélection d'une ligue S est basée sur les deux propriétés suivantes :

Propriété 1.3 *[Bergeron et al., 2002b] L'ensemble des δ-chaînes maximales forment une partition de E.*

Propriété 1.4 *[Bergeron et al., 2002b] Toute δ-chaîne maximale d'un chromosome est une ligue.*

L'efficacité de l'algorithme repose évidemment sur la taille maximum des sous problèmes. Nous montrons tout d'abord qu'il existe toujours une ligue S, si elle n'est pas un Gene Team, de taille au plus $|S|/2$ gènes.

Lemme 1.3 *[Bergeron et al., 2002b] Si S est une ligue et S n'est pas un Gene Team, il existe une sous ligue de S avec au plus $|S|/2$ gènes.*

Preuve : Soit S une ligue, n'étant pas un Gene Team. Si S n'est pas un Gene Team, alors S est au moins l'union de plusieurs d'entre eux. Et il ne peut y avoir plus d'un Gene Team de plus $|S/2|$ gènes car les δ-chaînes forment une partition de E. S a toujours une sous ligue avec au plus $|S/2|$ gènes. ◄

Pour une ligue S, le premier sous problème est donc de taille au plus $|S|/2$ et le deuxième de taille $n - |S|$ avec $|S| \leq n/2$.

La séparation en deux sous problèmes demande pour le premier sous problème d'extraire une ligue des chromosomes. L'extraction dans le premier chromosome ne pose pas de problème car on vient de parcourir la ligue. Pour l'extraire du deuxième chromosome, il suffit de garder à jour les positions relatives des gènes que l'on vient d'extraire du premier chromosome dans le deuxième chromosome. Si on tri ces positions, on obtient l'extraction de la ligue dans le

deuxième chromosome. Si la ligue a au plus p gènes, l'extraction est effectuée en $O(p \log p)$. Dans un deuxième temps, pour l'extraction du deuxième sous problème, il est nécessaire d'obtenir le complémentaire des extractions déjà effectuées. Cela peut être effectué en temps linéaire par rapport à la taille de la ligue.

Comme on ne connaît pas dans le pire des cas la taille de la ligue à extraire, on suppose que la séparation en deux sous problèmes est effectuée en $O(\alpha p \log p)$ pour $p = |S|$, la taille de la ligue courante. L'équation de récurrence exprimant la complexité de l'algorithme est :

$$T(n) = \begin{cases} 1 \text{ pour } n = 1 \\ \max\limits_{1 \leq p \leq \llcorner n/2 \lrcorner} \alpha p \log p + T(p) + T(n-p) \text{ sinon} \end{cases}$$

Ce qui donne finalement une complexité dans le pire des cas en $O(n \log^2 n)$ pour l'algorithme comparant deux chromosomes [Bergeron et al., 2002b]. Nous nommerons FFT (pour *Fast Find Teams*) cette méthode de détection des Gene Teams, décrite par l'algorithme 4.

Comparaison de m chromosomes $C^1, ..., C^m$.

L'algorithmique reste la même sauf que la ligue de départ est recherchée sur tous les chromosomes car elle doit être de taille p inférieure à $n/2$ sur au moins un des chromosomes. La division du problème s'effectue maintenant en extrayant les m sous ligues de taille p des m chromosomes. La méthode de détection des Gene Teams est décrite par l'algorithme 5.

La complexité de cet algorithme est en $O(mn \log^2 n)$ pour m chromosomes de taille au plus n gènes.

1.2.3 Une approche informelle : les Connected Gene Neighborhoods

Les Connected Gene Neighborhoods (CGN) [Rogozin et al., 2002] sont la seule alternative connue aux définitions précédentes. Alors que les modèles comme les Intervalles Communs et même les Gene Teams sont très strictes quant à la définition des objets comparés, les CGN relâchent ces contraintes en autorisant les gènes paralogues, les gènes non communs à tous les génomes et l'insertion/délétion de gènes. Les reproches qu'on peut leur faire est qu'ils sont très peu précis mais surtout qu'il font appel à un problème NP-complet : la recherche de chemins maximaux dans un graphe orienté.

Les CGN ne recherchent pas des clusters de gènes existants dans tous les génomes comparés mais des clusters dont des sous ensembles chevauchants existent dans différents génomes. Par exemple, si les gènes $g_1 g_2 g_3 g_4$ peuvent être identifiés sous cette forme dans trois génomes, et si dans trois autres génomes, les gènes $g_2 g_3 g_4 g_5$ peuvent être identifiés sous cette forme, alors les gènes $g_1 g_2 g_3 g_4 g_5$ forment un CGN.

Pour justifier cette propriété, Rogozin *et al.* se basent sur la définition des über-operons introduite par [Lathe et al., 2000]. Succinctement, un opéron est une suite de gènes présente dans un génome et transcrite en un seul brin d'ARN messager (voir section 0.1 page 11) ; un über-operon est une structure représentant un ensemble de gènes provenant de divers opérons présents dans plusieurs génomes. La contrainte étant que le contenu en gènes des über-operons

entrée : $C^1 = g^1_1...g^1_n, C^2 = g^2_1...g^2_n$ et $E = \{1, ..., n\}$
début
1. `SousLigue` ≔ **Faux**
2. `p` ≔ 0
3. **Tant Que** ¬`SousLigue` **et** `p` $\leq \llcorner n/2 \lrcorner$ **Faire**
4. **Si** $\Delta_{C^1}(g^1_p, g^1_{p+1}) > \delta$ **ou** $\Delta_{C^1}(g^1_{n-p}, g^1_{n-p+1}) > \delta$ **ou** $\Delta_{C^1}(g^2_p, g^2_{p+1}) > \delta$ **ou** $\Delta_{C^1}(g^2_{n-p}, g^2_{n-p+1}) > \delta$ **Alors**
5. `SousLigue` ≔ **Vrai**
6. **Sinon**
7. `p` ≔ `p` + 1
8. **FinSi**
9. **Fait**
10. **Si** `SousLigue` **Alors**
11. **Si** $\Delta_{C^1}(g^1_p, g^1_{p+1}) > \delta$ **Alors**
12. `FFT` $\left(g^1_1...g^1_p, \texttt{extract}\left(g^1_1...g^1_p, C^2\right)\right)$
13. `FFT` $\left(g^1_{p+1}...g^1_n, C^2 \setminus \texttt{extract}\left(g^1_1...g^1_p, C^2\right)\right)$
14. **Sinon Si** $\Delta_{C^1}(g^1_{n-p}, g^1_{n-p+1}) > \delta$ **Alors**
15. `FFT` $\left(g^1_1...g^1_{n-p}, \texttt{extract}\left(g^1_1...g^1_{n-p}, C^2\right)\right)$
16. `FFT` $\left(g^1_{n-p+1}...g^1_n, C^2 \setminus \texttt{extract}\left(g^1_1...g^1_{n-p}, C^2\right)\right)$
17. **Sinon Si**
18. ...
19. **Sinon Si**
20. ...
21. **FinSi**
22. **Sinon**
23. `affiche` $\left(g^1_1...g^1_n\right)$
24. **FinSi**

Algorithme 4: Recherche de Gene Teams : FFT, la méthode diviser pour régner. La fonction extract$(g_1...g_k, C)$ transforme les gènes autres que $g_1...g_k$ de C en ★. On extrait $g_1...g_k$ de C.

est en partie partagé entre plusieurs opérons provenant d'espèces différentes. Nous ne nous attardons pas pour l'instant sur le concept des über-operons, sur lequel nous reviendrons plus tard, car nous considérons que les CGN s'éloignent fortement de ce concept dans leur mise en œuvre. Pour une explication détaillée du concept des über-operons, le lecteur est prié de se référer à la section 2.1 page 48.

La définition des Connected Gene Neighborhoods est induite par leur processus de détection et ne peut être simplement décrite. Nous présentons donc d'abord les différentes étapes de la méthode permettant l'identification des Connected Gene Neighborhoods. Puis nous discuterons des objets détectés par cette méthode.

La première étape construit un graphe orienté $G = (S, A)$. S l'ensemble des sommets de G correspond à l'ensemble des familles de gènes (voir section 0.2 page 15). Une famille de gène est identifiée par un représentant. Le représentant peut être un des gènes de cette famille. En pratique on prendra le nom du gène chez *Escherichia coli K12*, l'organisme le plus étudié et

entrée : $C^i = g^i_1 ... g^i_n$ pour $1 \leq i \leq m$ et $E = \{1, ..., n\}$
début
1. p ≔ 0
2. s ≔ 0
3. gauche ≔ **Faux**
4. **Tant Que** s $= 0$ **et** p $\leq \llcorner n/2 \lrcorner$ **Faire**
5. p ≔ p $+ 1$
6. **Pour chaque** $i \in \{1, ..., m\}$ **Faire**
7. **Si** $\Delta_{C^i}(g^i_p, g^i_{p+1}) > \delta$ **ou** $\Delta_{C^i}(g^i_{n-p}, g^i_{n-p+1}) > \delta$ **Alors**
8. s ≔ i
9. **Si** $\Delta_{C^i}(g^i_p, g^i_{p+1}) > \delta$ **Alors**
10. gauche ≔ **Vrai**
11. **FinSi**
12. **FinSi**
13. **Fait**
14. **Fait**
15. **Si** s > 0 **Alors**
16. **Si** gauche **Alors**
17. **Pour chaque** $i \in \{1, ..., m\}$ **Faire**
18. $C'^i =$ Extract $\left(g^s_1 ... g^s_p, C^i\right)$
19. **Fait**
20. **Sinon**
21. **Pour chaque** $i \in \{1, ..., m\}$ **Faire**
22. $C'^i =$ Extract $\left(g^s_{n-p+1} ... g^s_n, C^i\right)$
23. **Fait**
24. **FinSi**
25. MFFT $\left(\{C'^1, ..., C'^m\}, \{g^s_1 ... g^s_p\}\right)$
26. MFFT $\left(\{C^1 \setminus C'^1, ..., C^m \setminus C'^m\}, \{g^s_{p+1} ... g^s_n\}\right)$
27. **Sinon**
28. affiche $\left(g^1_1, ..., g^1_n\right)$
29. **FinSi**

Algorithme 5: Recherche de Gene Teams : MFFT, la méthode diviser pour régner pour m chromosomes.

donc le mieux annoté. L'ensemble des arcs, A, est l'ensemble des couples de gènes qui pour au moins trois génomes sont distants d'au plus deux gènes et localisés sur le même brin d'ADN. Le brin d'ADN sur lequel est localisé le couple de gènes donne l'orientation de l'arc.

La deuxième étape est une recherche de chemins maximaux dans le graphe G avec la contrainte supplémentaire qu'un chemin passant successivement par les nœuds a, b et c vérifie que la suite de gènes (a, b, c) existe dans au moins l'un des génomes (i.e. – en omettant les gènes paralogues – a, b, c forme une δ-chaîne pour $\delta = 2$). Ces chemins maximaux donnent des ensembles de gènes nommés *tableaux de paires de gènes conservées*.

La troisième et dernière étape effectue le regroupement des tableaux de paires de gènes conservées entre eux : dès qu'ils partagent au moins trois gènes si les tableaux sont de tailles

au moins quatre ; ou dès qu'ils partagent deux gènes si les tableaux sont de taille trois. Ces groupes de gènes sont appelés des Connected Gene Neighborhoods. La figure 1.3 représente les différentes étapes de la sélection des Connected Gene Neighborhoods.

On peut remarquer la difficulté de donner une définition aux CGN, la seule manière de les définir et de décrire un algorithme qui s'apparente à une suite de filtres. Par conséquent il est difficile d'interpréter les résultats donnés par cette méthode.

Pris indépendamment, chaque filtre s'explique par des considérations biologiques. Le premier filtre sélectionne tous les couples de gènes distants d'au plus deux gènes. En effet on observe chez les procaryotes que les gènes adjacents ont souvent des fonctions apparentées. En autorisant l'insertion de gènes entre les paires de gènes, la notion d'adjacence est étendue au voisinage proche des gènes. Le deuxième filtre sélectionne les paires de gènes présentes dans au moins trois génomes. En appliquant ce filtre, les adjacences n'étant pas lié au partage d'une fonction similaire mais dues au hasard sont éliminées. Les paires de gènes doivent aussi être localisées sur le même brin d'ADN dans au moins trois génomes. Puis vient la sélection des tableaux de paires de gènes. Ces deux contraintes sont liées à la notion d'opéron : un opéron est une suite de gènes transcrits à partir du même brin d'ADN ; et les gènes d'un même opéron sont corégulés et codent souvent pour une fonction apparentée. Le filtre suivant oblige ces triplets de paires de gènes à exister dans au moins trois génomes. Cela pour éviter que le lien entre les gènes soit trop faible et que les triplets n'existent dans aucun des génomes. Finalement, les tableaux de paires de gènes sont regroupés dès qu'ils partagent suffisamment de gènes.

Malgré l'absence de définition claire des groupes de gènes qu'on obtient avec les CGN, la méthode permet de détecter des structures conservées à plusieurs génomes. Non seulement les duplications sont prises en compte, ce qui est indispensable pour comparer des génomes de procaryotes étant donné leur nombre, mais les clusters recherchés ne doivent pas obligatoirement appartenir à tous les génomes comparés. Pour utiliser la méthode des CGN, nous n'avons pas besoin de connaissance *a priori* sur les clusters recherchés, ni sur les espèces à comparer, ni sur leur nombre.

1.3 Les limites des approches existantes

Nous avons introduit la définition de clusters de gènes et trois méthodes permettant d'en détecter. Ces méthodes sont très différentes et se classent soit dans la catégorie des approches formelles, soit dans la catégorie des approches informelles.

Les méthodes formelles reposent sur un modèle mathématique et des définitions strictes, les objets détectés ne seront pas dûs au hasard dès que l'on compare suffisamment de gènes, et il est aisé d'interpréter les résultats obtenus. Par contre, les problèmes de ces méthodes sont liés aux données biologiques qui nous intéressent : les génomes de procaryotes ne se prêtent pas aux modèles définis, lesquels n'autorisent pas les gènes dupliqués ; phénomène pourtant hautement présent dans les génomes de procaryotes. Par exemple chez *Escherichia coli K12*, 1482 gènes sont des gènes dupliqués pour un total de 4311 gènes. Et cette caractéristique est commune à l'ensemble des procaryotes car en moyenne 30% de leur gènes sont des gènes paralogues [Gevers et al., 2004].

Le deuxième point faible de ces méthodes vient de leur algorithmique certes efficace mais du coup trop restrictive. La grande variété des génomes ne permet pas de découvrir des clusters présents dans tous les génomes, il suffit d'un seul génome ne contenant pas tel cluster pour qu'il ne soit plus détectable. Pour k fixé, un cluster recherché dans un ensemble de k génomes est recherché commun à exactement k génomes. La contrainte liée à la présence des clusters dans tous les génomes est bien trop stricte pour être appliquée aux données biologiques. Rechercher des clusters avec cette contrainte nécessite de sélectionner judicieusement les espèces comparées, et d'ailleurs on ne découvrira que des clusters déjà étudiés, où dont on a une bonne idée *a priori*. Du point de vue algorithmique, tester tous les sous ensembles d'espèces de toutes les tailles n'est pas praticable. Pour n espèces, rechercher tous les clusters de gènes communs à exactement k espèces parmi n revient à tester $\begin{pmatrix} n \\ k \end{pmatrix}$ configurations différentes. Le nombre de configurations à tester si on prend $k \geq m$ est $\sum_{m \leq k \leq n} \begin{pmatrix} n \\ k \end{pmatrix}$.

De même, les définitions de ces clusters de gènes obligent les clusters à contenir précisément le même ensemble de gènes sur tous les génomes comparés. Les Gene Teams autorisent les gènes insérés mais le cluster est constitué, dans tous les génomes, du même ensemble de gènes. Réciproquement les gènes supprimés par une délétion, le sont dans tous les génomes, et n'appartiennent donc à aucun cluster. Un cluster peut très bien être incomplet dans un des génomes comparés, tout en restant un cluster biologiquement valide. Un gène non présent dans une ou plusieurs espèces ne doit pas empêcher de détecter un cluster le prenant en compte.

Ces trois critiques n'ont plus lieu d'être dans la méthode descriptive, les CGN, mais celle-ci pose d'autres problèmes. La critique propre à la méthode informelle est justement la succession de filtres nécessaires pour obtenir les clusters de gènes. On applique une suite de filtres compréhensibles pris séparément, mais quelle est leur implication dans le résultat final ? Et par ailleurs, les CGN font appel à la résolution d'un problème NP-complet.

Aucune de ces trois méthodes ne permet de mesurer la fiabilité des clusters proposés. Nous n'avons pas d'indication sur la probabilité d'observer de tels clusters à partir des données de départ. Une première approche à été proposé dans [Durand and Sankoff, 2003] mais ne permet pas, pour l'instant, d'obtenir des probabilités sur les clusters obtenues à partir des données réelles. Le besoin n'est pas le même pour les trois méthodes. Les Gene Teams et les Intervalles communs, par leurs nombreuses contraintes, donnent des clusters fiables pour peu que l'on compare suffisamment de génomes. Par contre les CGN ne se prêtent pas à cette interprétation naturelle et auraient nécessité d'avoir une mesure de la fiabilité des objets détectés. Car plus le nombre de génomes comparés est important, plus les clusters de gènes seront de grande taille et par conséquent peu informatif.

On peut faire une dernière critique à ces définitions, celle-ci est commune aux CGN et aux Gene Teams. Les clusters de gènes identifiés par les méthodes présentées ne sont valables qu'à un seul niveau de résolution : chaque gène appartient à un unique cluster (dans une moindre mesure pour les CGN). Cela pourrait suffire s'il n'existait pas plusieurs niveaux de relations et différentes interactions entre les protéines. En effet un gène peut appartenir à plusieurs clusters liés à des mécanismes étrangers les uns aux autres. Un cluster peut aussi correspondre à un niveau d'interaction : les clusters sont composés de sous clusters constitués de gènes avec

des interactions plus directes entre eux ; et ces sous clusters sont eux mêmes constitués de sous clusters. On peut citer par exemple le complexe du ribosome formé de deux sous unités. Le ribosome permet de lire l'ARN messager et de produire des séquences peptidiques. Les deux sous unités sont différentes et participent toutes les deux à la phase de traduction mais sont aussi des complexes indépendants. Au début de l'initiation de la traduction, les deux sous-unités du ribosome sont dissociées. Puis, la petite sous-unité forme un complexe avec l'ARN messager. La grande sous-unité vient alors s'ajouter à cet ensemble pour former le ribosome. Les méthodes décrites donnent une seule solution, figée, et leurs résultats ne représentent donc qu'une seule facette de la réalité biologique des clusters de gènes. Ces méthodes ne permettent pas d'obtenir à la fois les deux sous unités et leur union formant le ribosome. Seul les Intervalles communs proposent des niveaux hiérarchiques car leurs clusters se chevauchent les uns les autres, mais leur modélisation est tellement éloignée des réalités biologiques qu'ils ne sont concrètement que rarement utilisable.

Prédire des clusters de gènes, et donc des ensembles de gènes avec des fonctions apparentées, en se basant sur les réarrangements génomiques est sujet à quelques artefacts. Il est évident que des gènes peuvent être regroupés sur un génome sans pour autant partager une même fonction. L'hypothèse faite dans les méthodes de prédiction est qu'en observant suffisamment de fois le même groupe de gènes dans des génomes distincts, cette localisation en groupe des gènes est due à un bénéfice pour l'organisme vivant. On suppose alors que le bénéfice est lié au partage d'une fonction par les gènes regroupés. Lawrence et Roth ont été les premiers à signaler des alternatives à ce regroupement spatial des gènes : les opérons indépendants ou *Selfish operons* [Lawrence and Roth, 1996]. Une suite de gènes, conservée dans plusieurs génomes, peut être composée de gènes indépendants des autres gènes composant les génomes. Ces gènes seraient soumis aux transferts horizontaux et se regrouperaient pour pouvoir être plus facilement transférables, en un seul événement évolutif, d'un génome à un autre. Détecter de tels clusters génère alors des faux positifs. Un deuxième artefact au regroupement des gènes dans les génomes a été identifié par Rogozin *et al.* Ils ont observé que, dans un ensemble de gènes regroupés, peuvent cohabiter un ou plusieurs gènes sans que leurs fonctions aient un rapport quelconque avec les fonctions des autres gènes du regroupement. Le bénéfice de ces gènes à être localisés conjointement avec les autres gènes est lié à leur corégulation. Les gènes intrus profitent du mécanisme de régulation propre aux gènes voisins. Ce phénomène est nommé le covoiturage (*Car-pooling*) dans le cas de plusieurs gènes insérés et l'auto-stop (*Hitch-Hiking*) lorsqu'un seul gène est inséré [Rogozin et al., 2002]. Lorsque l'on prédit des clusters de gènes, il faut faire attention à ces artefacts de localisation et aucune des méthodes proposées ne permet de différencier les vrais clusters de gènes, dont les gènes ont des fonctions apparentées, des clusters de gènes issus de transferts horizontaux ou de phénomènes de covoiturage. Cette distinction entre les clusters ne semble pas réalisable en se basant simplement sur les caractéristiques spatiales des gènes. C'est un problème inhérent à l'utilisation des adjacences conservées entre les gènes.

1.4 Résumé et conclusions

Nous avons présenté deux classes de méthodes pour la détection de clusters de gènes. La première classe correspondant aux méthodes formelles est représentée par les Intervalles

communs et les Gene Teams. Les Intervalles communs définissent les clusters de gènes comme des suites de gènes présentes (de manière exacte) à l'ordre près. Les Gene Teams généralisent cette notion en autorisant l'insertion de gènes et prennent en compte, en tant qu'une insertion, les gènes non communs à tous les génomes comparés. Ces deux définitions sont trop strictes car leur modèle de clusters ne se prête pas aux données biologiques où les duplications de gènes sont présentes en grand nombre. De plus, L'algorithmique efficace mise en place impose la présence de ces clusters dans tous les génomes comparés. Leur point fort est dans l'interprétation des résultats : leur définition formelle stricte et relativement simple permet d'être catégorique sur l'importance des clusters détectés dès qu'ils sont détectés dans suffisamment de génomes.

La deuxième classe de définition, les méthodes descriptives, souffrent du manque de formalisme de leur définition et, par conséquent, on ne peut être formel sur l'importance des objets identifiés. Par contre, leur relative relâche des contraintes permet de prendre en compte les gènes dupliqués et les gènes non communs à tous les génomes ; mais surtout les clusters de gènes identifiés ne doivent pas nécessairement être présents dans tous les génomes. Revers de la médaille, les clusters trouvés peuvent n'être présents dans aucun des génomes. Nous avons présenté les Connected Gene Neighorhoods, la seule méthode descriptive à notre connaissance. Celle-ci fait d'ailleurs appel à un problème NP-complet.

Nous pouvons à partir des méthodes présentées, définir le cahier des charges d'une méthode idéale, résolvant les problèmes cités ci-dessus. Au niveau des séquences génomiques, cette méthode doit prendre en compte les gènes dupliqués et autoriser l'insertion et la délétion des gènes. Au niveau des relations entre les gènes, cette méthode doit identifier les différents niveaux hiérarchiques des clusters. Elle doit aussi mettre en avant l'organisation en modules des fonctions biologiques. Notamment cette méthode doit être capable d'associer plusieurs clusters à un même gène ; chacun de ces clusters représentant des mécanismes différents. Pour découvrir de nouvelles structures sans *a priori*, les groupes de gènes recherchés ne doivent pas obligatoirement faire partie de tous les génomes comparés. Étant donné un ensemble de génomes, on cherche des clusters de gènes communs à un sous ensemble de ces génomes, mais sans préciser lequel. Nous considérons qu'ajouter un génome à la comparaison ne doit pas appauvrir les résultats en ajoutant des contraintes mais au contraire doit donner de nouvelles possibilités. L'algorithmique mise en place doit évidemment correspondre à la définition des objets d'intérêt tout en restant praticable pour permettre la comparaison simultanée d'une multitude de génomes. Les objets détectés doivent effectivement exister dans les génomes et avoir une réalité biologique fondée. Et finalement, on doit être capable de mesurer la pertinence des groupes de gènes détectés.

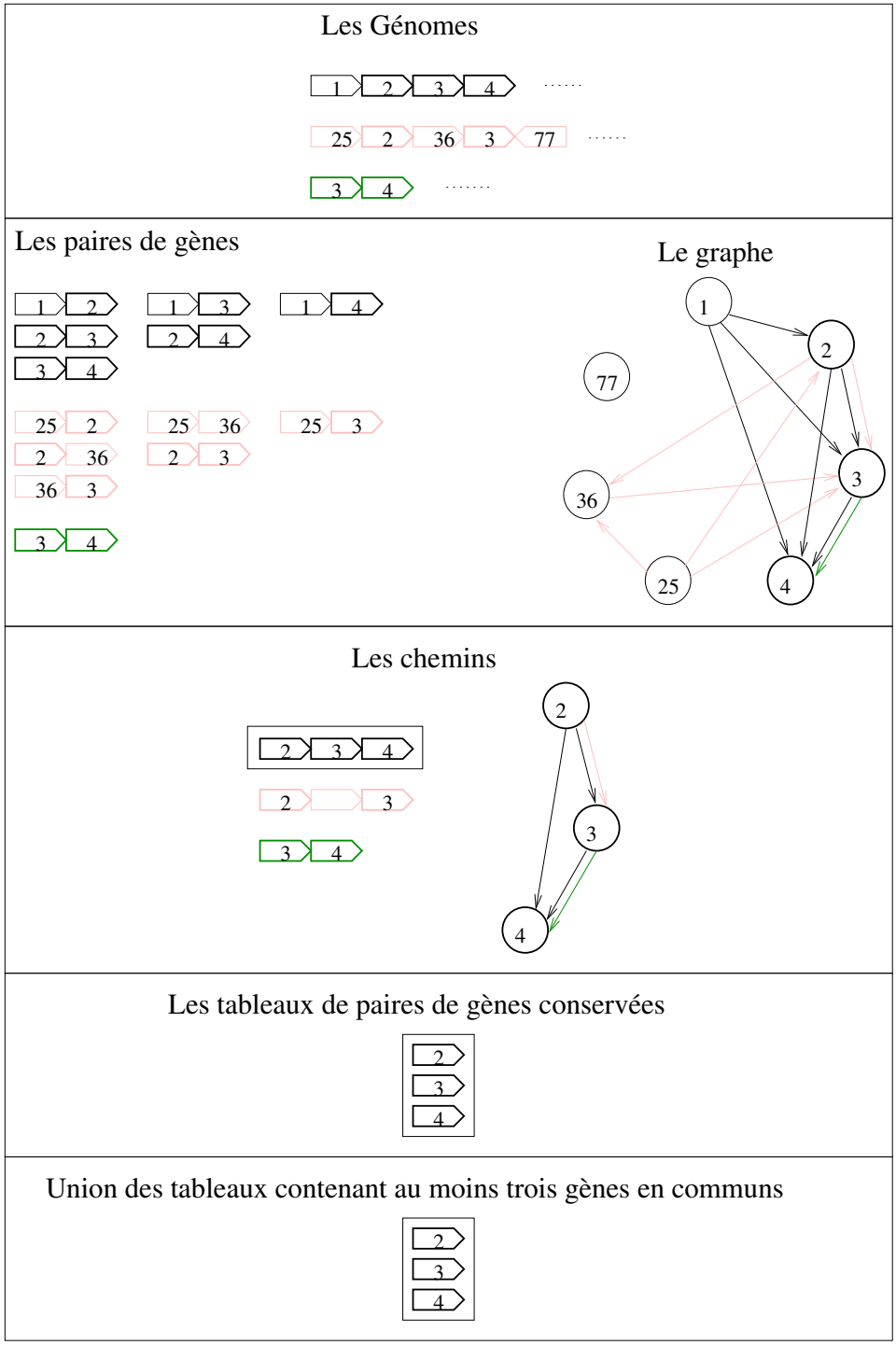


Figure 1.3: Les étapes permettant la détection des Connected Gene Neighborhoods. Les génomes sont représentés sur une ligne. Les boîtes représentent des gènes annotés par un nom de famille de gènes. Le sens des boîtes reflète le brin à partir duquel seront transcrits les gènes. Nous avons mis en avant les familles de gènes $2, 3$ et 4 qui se retrouvent souvent dans le même voisinage et qui donneront un CGN. Pour chacun de ces génomes nous construisons les paires de gènes conservées. On suppose que les trois génomes donnés sont chacun présent en trois exemplaires. Les paires de gènes permettent d'obtenir le graphe G. On recherche dans la graphe les chemins maximaux avec comme contrainte que les suites de trois gènes des chemins doivent réellement exister dans au moins un des génomes. Le chemin $2, 3, 4$ correspond à ces contraintes. Les chemins donnent les tableaux de paires de gènes conservées. Finalement en faisant l'union de tous les tableaux contenant au moins trois gènes en commun on obtient les CGN dont celui formé par les familles de gènes $2, 3$ et 4.

chapitre 2

Détection d'über-operons

Le dogme central de la biologie, introduit par Francis Crick, décrit les différentes étapes nécessaires pour passer des séquences d'ADN aux protéines. Dans un premier temps, c'est la transcription des séquences d'ADN, les gènes, en ARN messagers. Puis dans un second temps, ces ARN messagers sont traduits en protéines qui permettront par exemple l'élaboration des tissus musculaires, le transport de l'oxygène mais aussi la lecture de l'ADN et sa transcription.

Le dogme central est une vue simplifiée des mécanismes de l'ADN, en effet certaines séquences d'ADN ne donneront pas de protéines mais resteront sous la forme d'ARN. Par exemple, la machinerie de traduction elle-même fait intervenir deux types d'ARN ne codant pas des protéines, l'ARN des ribosomes et les ARN de transfert. De même, la transcription est le résultat d'un réseau de régulation complexe et dynamique. La transcription ne se fait pas qu'au niveau des gènes mais, pour les procaryotes, en grande partie au niveau des opérons.

Les études sur les relations entre les gènes d'un même opéron montrent qu'ils sont, dans la quasi totalité des cas, des gènes codant pour des fonctions apparentées. Par exemple l'opéron PhoP PhoQ chez *Escherichia coli K12* est constitué de deux gènes faisant partie du mécanisme du phosphate. De plus, comme l'organisation en opérons des génomes est vérifiée chez tous les procaryotes, il est naturel de comparer ces structures entre diverses espèces.

Nous présentons dans ce chapitre le fruit de notre travail sur la conservation des clusters de gènes et plus précisément, des opérons. Le concept de clusters de gènes sur lequel nous avons travaillé se nomme les über-operons. Ce terme, introduit par Lathe *et al.* en 2000, composé sur le nom opéron et sur l'adverbe über (au-dessus de), décrit une structure abstraite, construite à partir des opérons, et surtout conservée entre les espèces. Dans un premier temps, nous présentons les über-operons et leur structure qui en fait des clusters de gènes de choix.

Dans un deuxième temps, nous présentons l'algorithmique utilisée pour prédire ces clusters de gènes. Dans une troisième partie, nous présentons toutes les étapes nécessaires à la mise en place du cœur de l'algorithme proposé dans la partie précédente. La quatrième partie décrit les calculs de pertinence fait sur les über-operons. Elle montre aussi comment estimer la probabilité d'observer un über-operon dans des génomes aléatoires. La cinquième et dernière partie, compare notre méthode aux méthodes exposées dans le chapitre précédent en se basant sur des exemples réels de prédiction de clusters de gènes.

2.1 Le concept des über-operons

Les génomes de procaryotes sont organisés en opérons, lesquels constituent des suites de gènes aux fonctions apparentées. Les gènes des opérons forment donc des clusters de gènes et d'ailleurs des méthodes comme les Intervalles communs, lorsqu'elles prennent en compte le signe des gènes, s'approchent d'une méthode de détection d'opérons communs à plusieurs espèces. Étant donné que la structure en opérons a tendance à être conservée dans les génomes évolutivement proches [Terai et al., 2001], une telle méthode de détection de clusters de gènes semble envisageable. Mais la conservation des opérons diminue lorsque les espèces comparées sont évolutivement distantes et aussi lorsqu'on compare beaucoup d'espèces évolutivement proches. En effet au plus on compare d'espèces, mêmes proches, plus la probabilité d'une modification des opérons est grande. Pour fixer ce principe communément admis, nous avons étudié en pratique la conservation des opérons dans les génomes de bactéries. Pour cela, nous avons sélectionné trois espèces proches : *Escherichia coli K12*, *Escherichia coli O157:H7* et *Yersinia pestis*, toutes trois des protéobactéries de la sous famille gamma[2]. *Escherichia coli K12* et *Escherichia coli O157:H7* sont deux souches différentes de la même espèce. Et de même nous avons sélectionné des espèces distantes en prenant une espèce de chaque sous famille des protéobactéries : *Pseudomonas aeruginosa* pour la sous famille gamma, *Neisseria meningitidis Z2491* pour la sous famille beta, *Campylobacter jejuni* pour la sous famille epsilon/delta et *Mesorhizobium loti* pour la sous famille alpha. Le choix de ces espèces dans les sous familles a été fait en prenant l'espèce la mieux annotée par rapport aux autres espèces sélectionnées. Les résultats pour les espèces proches sont exprimés figure 2.1 et ceux pour les espèces des sous familles des protéobactéries sont exprimés figure 2.2. Malheureusement les opérons ne sont pas connus pour la plupart des espèces et nous avons dû utiliser un algorithme de prédiction qui sera développé dans la section 2.3. Cette expérimentation montre que les opérons, en dehors de quelques cas particuliers, ne sont pas conservés entre les bactéries. Et c'est d'ailleurs, entre autre, à cause de la non conservation des structures simples comme les opérons que des méthodes basées sur un modèle stricte, les Gene Teams par exemple, donnent peu de résultats.

Si les opérons ne sont pas conservés entre les espèces, c'est que les réarrangements génomiques, dont le résultat est la rupture de ces structures, existent. D'un autre côté, comme les opérons forment des unités fonctionnelles, les réarrangements les modifiant ne sont pas favorisés et sont moins souvent pérennisés que les autres. Même si les réarrangements génomiques modifient l'ordre des gènes et cassent des opérons, le résultat de ces réarrangements, lorsqu'ils sont fixés par l'évolution, est la formation de nouveaux opérons avec plus ou moins les mêmes caractéristiques. Les gènes des opérons sont mélangés, tout en gardant leur principe de base : les gènes d'un même opéron codent pour des fonctions apparentées. Les gènes associés à une fonction ou à un mécanisme doivent donc être regroupés en un ensemble d'opérons constituant un même ensemble de gènes. Si cette hypothèse est vérifiée, il reste des traces d'opérons morcelés entre les espèces. C'est sur cette hypothèse que sont bâtis les über-operons. Les über-operons forment une structure abstraite, au dessus des opérons, et conservée entre les différentes espèces. Le concept d'über-operon a été présenté par Lathe *et al.* dans [Lathe et al., 2000] et nous le reformulons de la façon suivante : un über-operon est l'ensemble des familles de gènes formant des traces d'opérons apparentés. C'est le produit d'une union d'opérons.

[2] Une classification des procaryotes est disponible en annexe

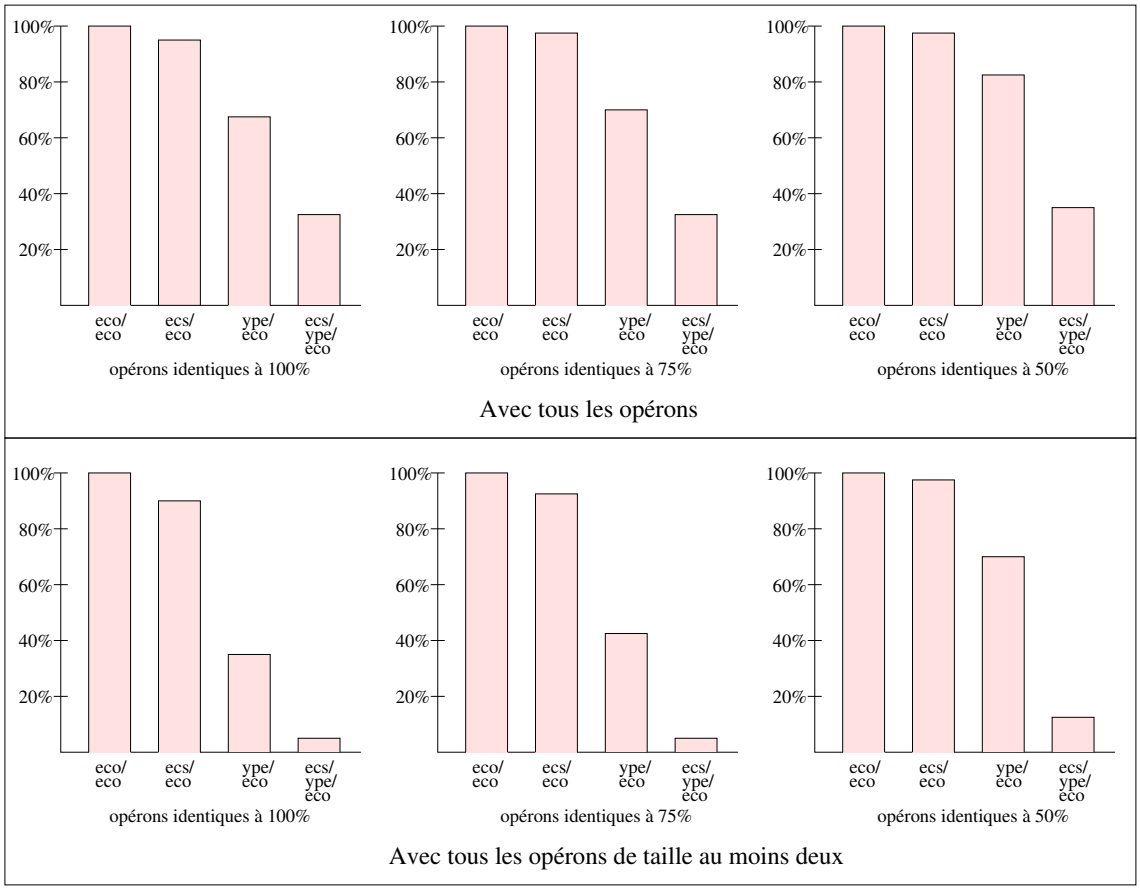


Figure 2.1: Étude de la conservation des opérons entre espèces évolutivement proches. Comparaison des opérons de *Escherichia coli K12* (eco) avec ceux de *Escherichia coli O157:H7* (ecs) et/ou ceux de *Yersinia pestis* (ype). Nous avons supprimé à l'intérieur des opérons tous les duplications de gènes. Le nombre d'opérons prédits comptabilisés chez *Escherichia coli K12*, l'espèce de référence de notre comparaison, est de 1 339. Il y en a 1 587 chez *Escherichia coli O157:H7* et 1 460 chez *Yersinia pestis*. Les trois graphiques du haut donnent le pourcentage d'opérons de eco identiques à 100%, 75% et 50% à ceux de eco, puis à ceux de ecs puis à ceux de ype, et finalement à ceux de ecs et de ype. L'identité entre deux opérons est définie comme le nombre de gènes en commun. Les trois graphiques du bas représentent la même information pour différents taux d'identité en ne comptabilisant que les opérons comprenant au moins deux gènes distincts. Le nombre de tels opérons est de 449 pour eco, de 561 pour ecs et de 404 pour ype. On constate effectivement que les opérons sont conservés entre les deux génomes proches de eco et de ecs. Mais qu'entre eco et ype, malgré le fait que ces deux espèces soient relativement proches, les opérons ne sont conservés qu'à moins de 70% et à moins de 40% si on considère les opérons de taille au moins deux. Le nombre d'opérons similaires décroît largement quand on considère les trois espèces réunies.

Avant de définir plus formellement les über-operons, nous nous attardons sur leur structure et ce qu'ils représentent. Nous reprenons l'exemple développé par Lathe *et al.* pour montrer l'existence d'une telle structure. La version que nous présentons est quelque peu modifiée ce qui nous permet de mieux comprendre et de mieux appréhender les caractéristiques permettant la

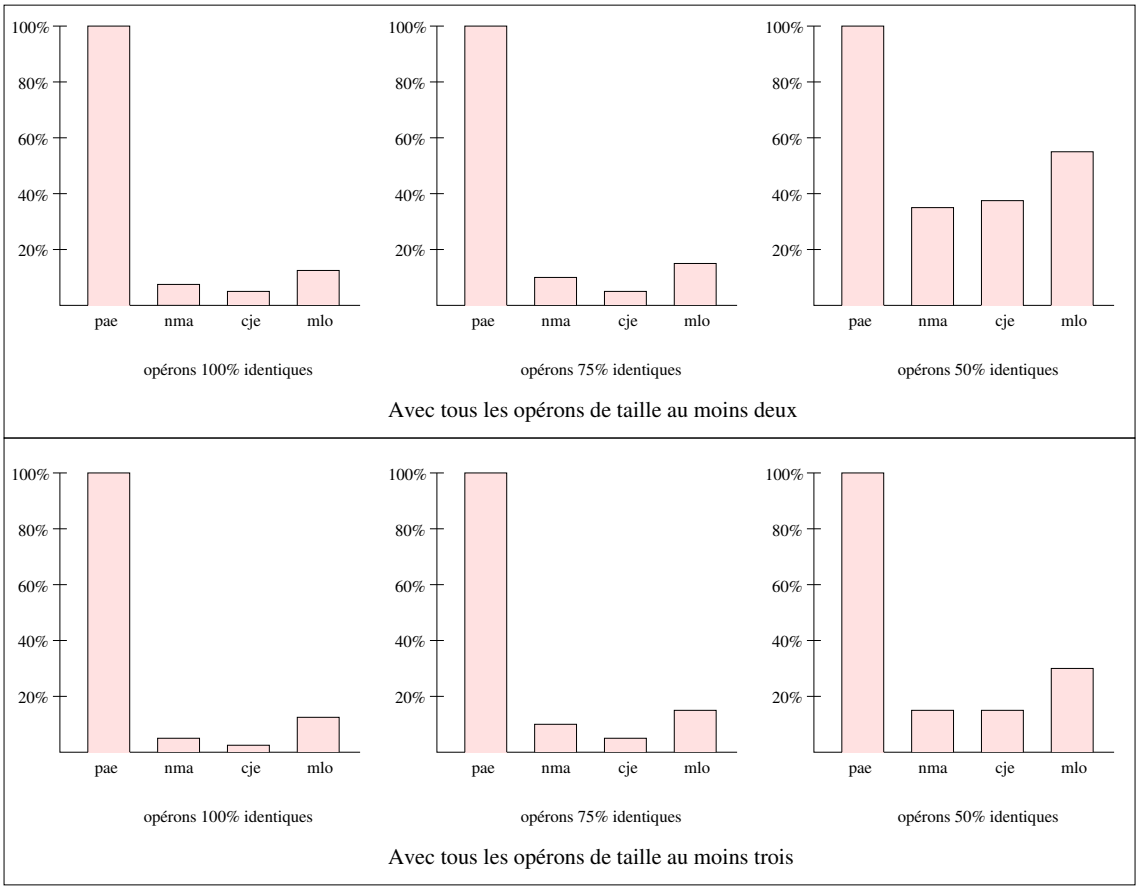


Figure 2.2: Étude de la conservation des opérons entre espèces provenant de sous familles de la famille des protéobactéries. Comparaison des opérons de *Pseudomonas aeruginosa* (pae) avec ceux de *Neisseria meningitidis Z2491* (nma) ou ceux de *Campylobacter jejuni* (cje) ou ceux de *Mesorhizobium loti* (mlo). Nous avons supprimé à l'intérieur des opérons tous les duplications de gènes. Le nombre d'opérons prédits de taille au moins deux comptabilisés chez pae est de 600 ; il y en a 269 chez nma, 218 chez cje et 552 chez mlo. Les trois graphiques du haut donnent le pourcentage d'opérons de pae identiques à 100%, 75% et 50% à ceux de pae, puis à ceux de nma puis à ceux de cje, et finalement à ceux de mlo. L'identité entre deux opérons est définie comme le nombre de gènes en commun. Les trois graphiques du bas représentent la même information pour différents taux d'identité en ne comptabilisant que les opérons comprenant au moins trois gènes distincts. Le nombre de tels opérons est de 286 pour pae, de 109 pour nma, de 132 pour cje et de 222 pour mlo. On constate effectivement que les opérons ne sont pas exactement conservés entre les génomes de différentes sous familles ; mais en prenant 50% pour le minimum d'identité entre deux opérons, on observe qu'il reste quelques traces des opérons.

détection des über-operons. Cet exemple est basé sur les gènes liés au mécanisme du flagelle chez les bactéries.

Les bactéries comme *Escherichia coli* ont environ six flagelles, positionnés aléatoirement à leur surface [Maki et al., 2000]. Le flagelle est une cellule filamenteuse qui permet aux bactéries qui en sont équipées de se déplacer en fonction de leur environnement. Les flagelles tournent

par défaut dans le sens contraire aux aiguilles d'une montre. Dans ce sens le flagelle ne fait pas avancer la bactérie mais lui permet de tourner aléatoirement sur elle même. Lorsque le flagelle est actionné dans le sens des aiguilles d'une montre, la bactérie se déplace propulsée par le flagelle. Par défaut les flagelles sont alternativement actionnés dans un sens puis dans l'autre. Suivant les stimulus détectés, un sens ou l'autre devient préférentiel et la bactérie se déplace en conséquence. S'il manque dans son environnement un composé nécessaire à son activité, les flagelles seront actionnées dans le sens des aiguilles d'une montre et celle-ci explorera son environnement à la recherche du composé manquant. Le flagelle est un mécanisme complexe, d'une part il est constitué d'un ensemble de protéines formant la cellule filamenteuse, et d'autre part il est constitué de "moteurs" actionnant la structure filamenteuse (figure 2.3). De plus les "moteurs" interagissent avec d'autres systèmes comme la détection de stimulus extérieurs.

Le flagelle ainsi que les mécanismes liés à son utilisation sont des structures complexes. Il n'existe pas d'espèce ayant l'ensemble des gènes codant pour ces mécanismes dans un seul opéron. Par contre, ces gènes étant plus ou moins apparentés entre eux peuvent se retrouver en partie dans les mêmes opérons et former ainsi un über-operon. En comparant suffisamment d'espèces, on peut espérer identifier tous les gènes impliqués dans les mécanismes liés au flagelle. Pour tester cette hypothèse, nous avons sélectionné quatre espèces, et pour chacune nous regardons où sont positionnés les gènes d'intérêt et surtout dans quels opérons.

D'après la base de données KEGG, *Kyoto Encyclopedia of Genes and Genomes* [Ogata et al., 1999], le flagelle et les mécanismes permettant le déplacement des bactéries sont constitués de 36 gènes : `fliC, fliD, fliE, fliF, fliG, fliH, fliI, fliJ, fliK, fliM, fliN, fliO, fliP, fliQ, fliR, fliS, flgA, flgB, flgC, flgD, flgE, flgF, flgG, flgH, flgI, flgK, flgL, flgM, flgN, motA, motB, flhA, flhB, flhC, flhD et flhT`. Les opérons auxquels ils appartiennent pour les quatre espèces suivantes, *Escherichia coli K12*, *Bacillus subtilis*, *Clostridium acetobutylicum* et *Yersinia pestis*, sont représentés figure 2.4.

La figure 2.4 montre pour les espèces *Escherichia coli K12* et *Yersinia pestis* un ensemble d'opérons prédits bien conservés : il y a peu de gènes dont les fonctions n'ont pas de lien avec les mécanismes du flagelle (ce sont les gènes en blanc). Si on prend l'union de ces opérons, on retrouve à peu près les gènes impliqués dans les mécanismes liés au flagelle. On obtient 136 gènes pour les deux espèces avec 20 d'entres eux qui n'ont *a priori* pas de rapport avec les mécanismes du flagelle. Ces gènes représentent $14,7\%$ des gènes issus de l'union des opérons. Ces opérons forment donc une structure d'über-operons. L'organisation de ces même gènes pour *Clostridium acetobutylicum* et *Bacillus subtilis* est moins claire, beaucoup de gènes sans rapport avec le flagelle sont insérés dans les opérons ($53,2\%$ des gènes isolés), mais d'un autre côté les gènes sont quand même distribués sur un petit nombre d'opérons. La structure en über-operon est moins conservée mais il en reste des traces évidentes. Le regroupement des gènes, en un certain nombre d'opérons, variant suivant les espèces considérées montre effectivement l'existence des über-operons.

Cet exemple confirme l'existence de la structure en über-operon des opérons. Mais un über-operon ne peut être strictement une union d'opérons partageant le même ensemble de gènes, comme il est proposé dans [Lathe et al., 2000]. En effet, nous avons vu que des gènes sont insérés sans pour autant faire partie du même mécanisme. Une méthode de détection d'über-operon doit détecter des opérons partageant les mêmes gènes, dont le contenu est mélangé

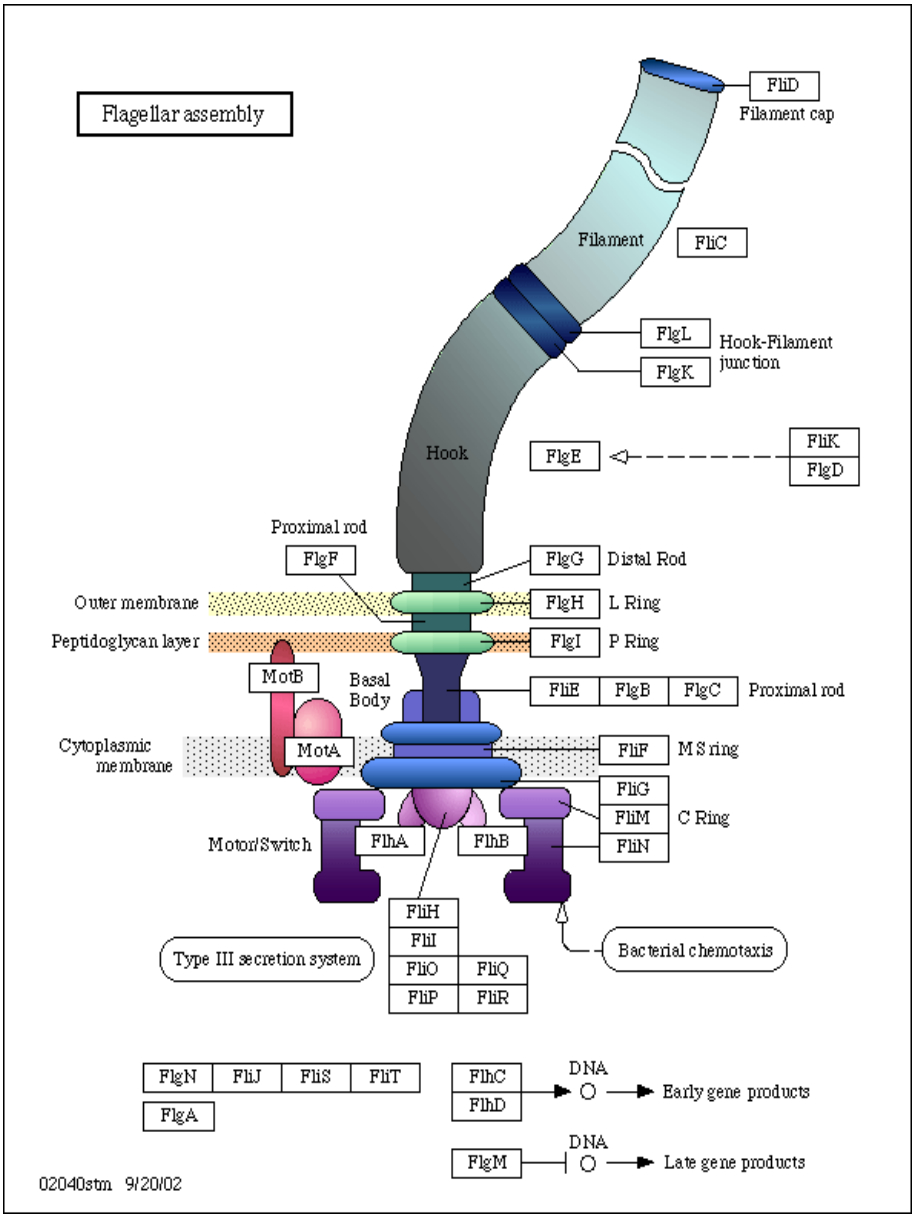


Figure 2.3: Le mécanisme du flagelle chez les bactéries. Le flagelle est un mécanisme complexe, d'une part il est constitué d'un ensemble de protéines formant la cellule filamenteuse, et d'autre part il est constitué de "moteurs" actionnant la structure filamenteuse. De plus "les moteurs" interagissent avec d'autres systèmes comme la détection de stimulus extérieurs (source : KEGG [Ogata et al., 1999]).

entre les opérons, tout en autorisant l'insertion de gènes étrangers et la délétion d'autres gènes.

La méthode et la caractérisation proposées par Lathe, Snell et Bork est la suivante : Ils proposent de sélectionner un gène. Par exemple un gène lié aux mécanismes du flagelle. Puis Pour ce gène, il détermine dans toutes les espèces comparées, tous les autres gènes appartenant aux mêmes opérons que celui-ci. Ces gènes détectés forment un début d'über-operon. On

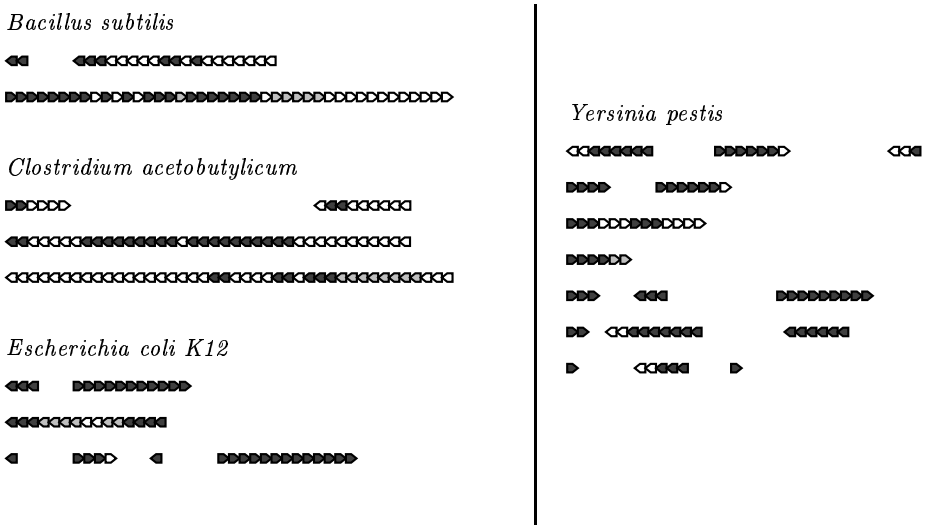


Figure 2.4: Projection des gènes impliqués dans les mécanismes du flagelle pour quatre génomes : *Bacillus subtilis, Yersinia pestis, Escherichiae coli* et *Clostridium acetobutylicum*. Les gènes côte à côte sont dans le même opéron prédit. Les gènes en gris foncés sont les gènes directement issus de la liste des gènes liés au flagelle donnée par KEGG ; les gènes en gris clair sont impliqués dans des mécanismes associés à celui du flagelle ; par contre, les gènes en blanc n'ont *a priori* pas de raison d'être associés au flagelle.

réitère le procédé avec comme gènes de départ les gènes nouvellement ajoutés à l'über-operon. Le processus s'arrête quand tous les gènes de l'über-operon ont été étudiés. Cette méthode n'est pas compatible avec les propriété des opérons que nous venons de mettre en avant. Le nombre de faux positifs prédits par cette méthode serait trop important. Dans notre exemple il faudrait ajouter tous les gènes des opérons où sont présents les gènes en blanc, et ainsi de suite.

2.2 Une modélisation simple et efficace

Nous avons défini dans la section précédente ce que représente un über-operon : un ensemble de gènes avec des fonctions apparentées. Nous définissons maintenant les über-operons au niveau des séquences génomiques et par conséquent une méthode pour les détecter.

Nous pouvons dans un premier temps définir les über-operons de manière informelle :

Définition 2.1 *Un über-operon est un ensemble de gènes formé par un ensemble maximal d'opérons partageant des gènes orthologues, des gènes paralogues et avec quelques gènes insérés ou délétés.*

Avant de présenter formellement les über-operons, nous introduisons quelques définitions nécessaires à leur définition. Rappelons d'abord qu'une famille de gènes est un ensemble de gènes codant pour une fonction identique. Soit Σ l'ensemble des familles de gènes. Toute famille de gènes présente dans une seule espèce, i.e. réduite à un singleton, sera renommée $\bigstar$ (étoile). Soit $\hat{\Sigma}$ l'ensemble de toutes les familles de gènes présentes au moins deux fois dans

l'ensemble des espèces. $\hat{\Sigma}$ est relatif aux espèces mises en jeu. Nous introduisons aussi quelques définitions sur les graphes :

Définition 2.2 *Un α-chemin dans un graphe non orienté pondéré $G = (S, A, \omega)$ est une suite de nœuds $s_1, ..., s_n$ de S tel que pour tout $1 \leq i < n$, $(s_i, s_{i+1}) \in A$ et $\omega(s_i, s_{i+1}) \geq \alpha$.*

Définition 2.3 *Une composante α-connectée d'un graphe non orienté pondéré $G = (S, A, \omega)$ est un sous graphe $G' = (S', A')$ de G tel que pour tout $u \in S'$ et $v \in S'$ il existe un α-chemin dans G' passant par u et v.*

Définition 2.4 *Une composante α-isolée d'un graphe non orienté pondéré $G = (S, A, \omega)$ est un sous graphe $G' = (S', A')$ de G tel que pour tout $u \in S'$ et pour tout $v \in S \setminus S'$, il n'existe pas de α-chemin passant par u et v.*

Définition 2.5 *Une composante α-connexe (maximale) d'un graphe non orienté pondéré $G = (S, A, \omega)$ est un sous graphe $G' = (S', A')$ de G tel que*

- *G' est une composante α-connectée*
- *G' est une composante α-isolée.*

Nous pouvons maintenant définir formellement les über-operons. Un génome est représenté par une liste ordonnée sur $\hat{\Sigma} \cup \{\bigstar\}$, appelée permutation. Soit $\Pi = \{\pi^i\}$ pour $i \in [1, n]$ l'ensemble des n permutations représentant les n génomes d'intérêt. Un opéron d'une permutation π est défini comme une suite de gènes de π. Les opérons ne se chevauchent pas et l'ensemble des opérons d'une permutation π forment une partition de $\hat{\Sigma}$. Soit un graphe orienté et pondéré $G = (S, A, \omega)$ nommé graphe des opérons. S l'ensemble des sommets est identique à $\hat{\Sigma}$ et il existe un arc $a = (s_1, s_2, \omega_{s_1,s_2})$ si les gènes représentés par les nœuds s_1 et s_2 sont exactement ω_{s_1,s_2} fois présents dans un même opéron.

Définition 2.6 *Un über-operon est une composante α-connexe (maximale) du graphe des opérons.*

La figure 2.5 donne un exemple de graphe des opérons pour trois séquences :

$$\begin{array}{lcccccccccccccc} \pi^1 = & 1 & 2 & 3 & 4 & 5 & 6 & \bigstar & 8 & 9 & 10 & 11 & \bigstar & 13 & 14 & 15 \\ \pi^2 = & 10 & 2 & \bigstar & 1 & 11 & 9 & 8 & 14 & \bigstar & 11 & 3 & 13 & 4 & 6 & \\ \pi^3 = & 5 & 15 & 8 & 9 & \bigstar & 14 & \bigstar & 15 & 13 & 11 & 2 & & & & \end{array}$$

Sur la figure, les traits sous les gènes, au niveau des génomes, délimitent les opérons. En dessous des séquences est représenté le graphe des opérons. Les arcs en pointillés sont les arcs de poids strictement inférieur à deux, par opposition aux arcs en gras de poids supérieur ou égal à deux. Nous avons représenté par différents niveaux de gris les ensembles de gènes se retrouvant dans les mêmes opérons. L'arc $(1, 2)$ est de poids deux car les gènes 1 et 2 sont deux fois présents dans un même opéron, une fois dans π^1 et une autre fois dans π^2. Bien que les gènes 3 et 4 soient adjacents dans π^1, il n'y a pas d'arc $(3, 4)$ dans le graphe des opérons car les

π^1 : 1 2 3 4 5 6 ★ 8 9 10 11 ★ 13 14 15

π^2 : 10 2 ★ 1 11 9 8 14 ★ 11 3 13 4 6

π^3 : 5 15 8 9 ★ 14 1 3 4 15 3 10 14 ★ 15 13 11 2

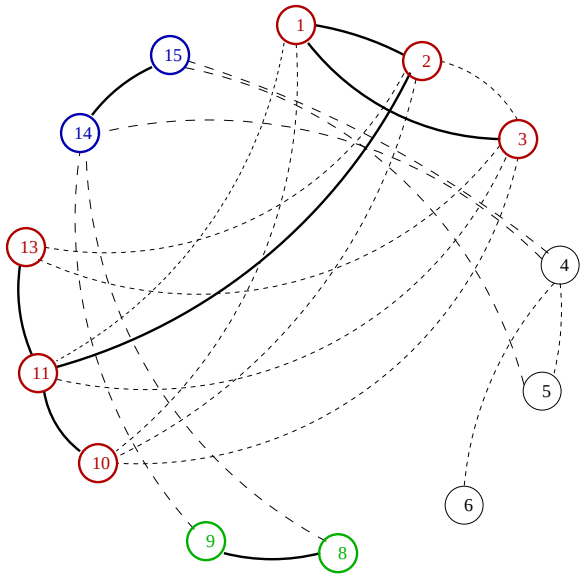


Figure 2.5: Trois chromosomes, leurs opérons et le graphe des opérons associé. Un über-operon est une composante α-connexe du graphe des opérons. Les arcs pleins sont les arcs de poids au moins égal à deux. Par exemple, pour $\alpha = 2$, les gènes $\{1, 2, 3, 10, 11, 13\}$ forment un über-operon.

gènes 3 et 4 ne font jamais partie d'un même opéron. Pour $\alpha = 2$, les composantes α-connexes sont représentées par les nœuds reliés par des arcs en gras. On obtient trois composantes α-connexes : $\{1, 2, 3, 10, 11, 13\}$, $\{8, 9\}$ et $\{14, 15\}$. Les über-operons déduits des composantes α-connexes sont bien les gènes grisés car ils se retrouvaient dans un même ensemble d'opérons.

Précisions sur les opérons et les duplications.

Un opéron peut être constitué de plusieurs unités de transcription comme dans l'exemple de la figure 2.6 avec les gènes cyoA, cyoB, cyoC, cyoD et cyoE. Suivant le promoteur utilisé, la transcription commence à différents endroits et produit des ARN messagers distincts. Non seulement nous avons pour cet exemple la co-régulation de cyoA, cyoB, cyoC, cyoD et cyoE mais nous pouvons avoir, par exemple et sous certaines conditions, la co-régulation de seulement cyoC, cyoD et cyoE. La définition des über-operons présentée ne prend pas en compte ces caractéristiques des opérons et ne définit un lien entre les gènes que pour l'unité de transcription englobante : cyoA, cyoB, cyoC, cyoD et cyoE. En effet, indépendamment des sous unités de transcription, les liens entre les gènes sont bien une co-régulation entre tous les gènes de l'opéron malgré la possibilité, suivant les conditions environnementales, d'avoir des

sous-unités de transcription. Les opérons définis pour le graphe des opérons, parce qu'ils ne se chevauchent pas, représentent bien les opérons et non leurs sous unités.

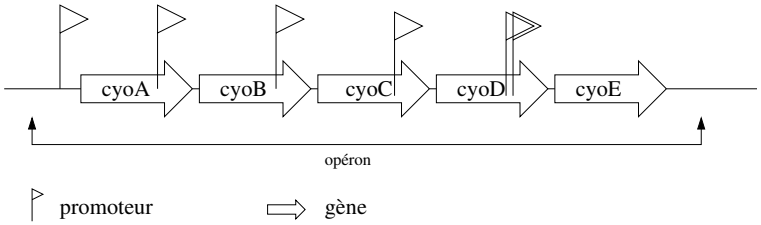


Figure 2.6: Un exemple d'opéron avec plusieurs unités de transcription (Transcription Unit). La figure représente l'opéron contenant les gènes CyoA CyoB CyoC CyoD CyoE chez Escherichia Coli [Salgado et al., 2004]. Les différents promoteurs à l'intérieur de l'opéron permettent plusieurs débuts de transcription distincts et par conséquent des ARN messagers dissemblables.

Il est nécessaire de préciser comment sont prises en compte les duplications. Les gènes dupliqués ne se retrouvant pas dans un même opéron ne posent pas de problème et sont analysés indépendamment les uns des autres. Un gène présent deux fois est pris en compte deux fois : chacune de ses occurrences génère un ensemble d'arc suivant les autres gènes présents dans le même opéron. Par contre, quand deux gènes issus d'une duplication sont dans un même opéron, leur présence dans l'opéron doit elle être comptabilisée deux fois ou une seule ? Pour répondre à cette question, il faut se demander si la présence de cette duplication implique des liens doublement plus forts entre ce gène et les autres gènes de l'opéron ? Que le gène soit présent deux fois dans un même opéron est certainement dû à une duplication en tandem. Mais pourquoi cette structure est elle conservée par l'évolution ? La double présence d'un gène dans un même opéron peut indiquer que ce gène doit être transcrit deux fois plus souvent que les autres gènes de l'opéron. S'il est transcrit deux fois, alors que les autres ne le sont qu'une seule fois, c'est que la concentration de sa protéine doit être double par rapport aux autres. Cela ne modifie pas la force des liens fonctionnels entres les gènes de cet opéron. Nous supposons que le lien entre les gènes de l'opéron et le gène dupliqué n'est pas plus important que les autres. Nous ne devons donc pas comptabiliser les multiples présences des gènes dans un même opéron. Un gène dupliqué, dans un même opéron, n'est comptabilisé qu'une seule fois alors qu'un gène dupliqué, dans des opérons distincts, représente différentes associations et est donc comptabilisé plusieurs fois.

Notes sur la méthode présentée et le paramètre α.

Le paramètre α permet de ne pas associer dans les über-operons les gènes présents moins de α fois dans ses opérons. Les gènes éliminés par ce paramètre sont des gènes insérés épisodiquement et ces insertions ne se retrouvent pas dans l'ensemble des espèces. Par exemple, les gènes insérés sont les gènes en blanc de la figure 2.4 page 53.

Les clusters de gènes détectés ne sont pas nécessairement présents dans un des génomes. Seulement une partie des gènes peut être conjointement présente dans un ou plusieurs génomes. Par contre, les clusters existent bien si on considère tous les génomes. Sans tenir compte des duplications, il suffit qu'un cluster appartienne à α génomes, pour qu'il soit identifié. Cette

caractéristique rend les clusters identifiés par notre méthode diamétralement opposés à ceux des Intervalles communs et des Gene Teams. Leur algorithmique efficace mais contraignante qui ne permet pas de détecter suffisamment de clusters dans des données réelles est en opposition avec l'algorithmique que nous proposons pour les über-operons.

On peut remarquer que les über-operons sont un mélange entre les méthodes basées sur les empreintes phylogénétiques et celles basées sur les réarrangements. En effet, prendre les composantes α-connexes revient à sélectionner les gènes qui sont co-présents, α fois, non plus dans un même génome, mais dans un même opéron.

Notes sur les objets détectés.

De manière informelle, un über-operon est un ensemble d'opérons partageant le même contenu. Le lien direct entre les gènes d'un même opéron est leur corégulation par la transcription. Le lien indirect, observé dans la plupart des cas, est qu'ils codent pour des fonctions apparentées. Les différents opérons formant un über-operon étant des opérons distincts, ils ne sont pas à première vue liés entre eux. Mais ces opérons partagent des gènes, c'est pour cela qu'ils forment un über-operon. En appliquant le principe de la transitivité entre les différents opérons partageant des gènes, un über-operon est formé d'un ensemble de gènes codant pour des fonctions apparentées. Les über-operons sont effectivement des clusters de gènes.

Même si les über-operons ne sont pas toujours exactement présents dans un génomes, ils restent des structures valides du point de vue biologique. Bien qu'abstraite, leur structure n'est simplement qu'une union d'opérons, laquelle par conséquent existe bien. Les objets détectés par notre méthode existent donc réellement dans les génomes, et les groupes de gènes correspondent bien à un ensemble de gènes avec des fonctions apparentées. Contrairement aux CGN, dont les groupes de gènes peuvent très bien ne rien représenter du point de vue biologique, les über-operons représentent un ensemble de gènes issus des mêmes opérons. En construisant nos groupes de gènes sur les opérons, nous nous assurons d'obtenir des clusters de gènes valides du point de vue biologique.

Les génomes à chromosomes multiples sont très simplement pris en compte dans notre méthode de détection des über-operons. Un génome avec k chromosomes est vu comme k génomes indépendants. Nous pouvons nous permettre cette représentation des génomes avec chromosomes multiples car les opérons ne peuvent appartenir à plusieurs chromosomes et parce que les génomes comparés, donc les chromosomes, ne doivent pas obligatoirement avoir les mêmes gènes.

Analyse de l'algorithmique

L'algorithme proposé ne permet pas, à lui seul et à partir de séquences d'ADN brut, de déterminer des über-operons. Nous avons présenté le cœur de l'algorithme ; lequel à partir d'un ensemble de permutations, avec le début et la fin des opérons connus pour chacune d'elle, et de l'ensemble des familles de gènes associées, permet de détecter des über-operons. Ces informations préalables, nécessaires pour la construction des permutations, seront discutées dans la section 2.3. Nous nous intéressons ici à la construction du graphe des opérons et à l'extraction des über-operons.

La construction du graphe des opérons se fait de façon optimale en parcourant chaque permutation π^i, $i \in [1, n]$. À la lecture de chacune de ces permutations on doit déterminer leurs opérons. Les opérons ne se chevauchant pas, les débuts et fin d'opérons peuvent être symbolisés par une cassure après un gène. D'ailleurs tout gène est dans un opéron, dût il être de taille un. On supposera que le test de la fin d'un opéron pour un gène donné s'effectue en temps constant. Le graphe des opérons se construit en parcourant les permutations pour déterminer leurs opérons et pour chaque opéron o de taille $|o|$, en ajoutant les (au plus) $\frac{|o|\times(|o|-1)}{2}$ arcs. Le nombre d'arcs ajouté au total est dans le pire des cas en $O(n \times m^2)$ pour n permutations, chacune de taille au plus m gènes. Finalement, l'algorithme détermine les composantes connexes du graphe des opérons en ne prenant en compte que les arcs de poids supérieur au seuil α fixé. Les composantes connexes d'un graphe $G = (S, A)$ sont recherchées en parcourant une et une seule fois les arcs du graphe G. La complexité de cette étape est dans le pire des cas en $O(|A|)$. Par conséquent, la complexité du cœur de l'algorithme de recherche des über-operons est, dans le pire des cas, en $O(n \times m^2)$.

En pratique, nous verrons dans la section 2.3 que le nombre de gènes des opérons est petit et borné. La complexité attendue en pratique de notre algorithme est donc en $\theta(n \times m)$ pour la détection d'über-operons parmi n génomes, chacun de taille au plus m gènes.

2.3 Les phases préliminaires

Le cœur de l'algorithme présenté précédemment ne permet pas à lui seul de détecter efficacement des über-operons. Nous avons extrait du concept des über-operons, un algorithme efficace qui présuppose certaines tâches : les familles de gènes doivent être définies et les opérons doivent être connus. De plus, comme comparer conjointement des génomes identiques peut perturber l'algorithme, les génomes que l'on veut analyser doivent former un ensemble cohérent.

Identification des familles de gènes

Les relations phylogénétiques entre les gènes comme l'orthologie et la paralogie, sont à la base de notre algorithme. Cette phase est donc cruciale pour obtenir des structures d'über-operons avec un sens biologique. Deux gènes provenant de deux génomes sont orthologues lorsqu'ils sont issus du même gène dans l'espèce ancestrale la plus proche. Par opposition, deux gènes paralogues sont issus d'un même gène s'étant dupliqué (voir figure 0.5 page 15).

Pour construire notre graphe des opérons nous avons besoin de définir l'ensemble des familles de gènes Σ. Une famille de gènes est formée d'un ensemble de gènes orthologues et paralogues. Cet ensemble de gènes issus d'un même gène ancestral a une fonction commune à chacun de ses gènes. Détecter des familles de gènes revient à détecter des groupes de gènes orthologues et paralogues. Plutôt que de recalculer toutes les relations entre les gènes à partir de leur similarité, nous avons choisi d'utiliser les COG, *Clusters of Orthologous Group* [Tatusov et al., 1997]. Les COG, malgré leur nom, sont bien des groupes de gènes orthologues et paralogues codant pour une fonction commune. Ils sont déterminés, non seulement en comparant la similarité des gènes mais aussi en prenant en compte les relations complexes avec les protéines à domaines multiples.

Ce choix nous permet de disposer d'informations sur les familles de gènes pour 66 génomes de procaryotes comportant 185 505 protéines. En se restreignant aux 66 génomes de procaryotes, les COG sont constitués de 4 873 groupes de gènes, ce qui représente une classification de 138 458 des 185 505 protéines. Mais surtout, les COG donnent des familles de gènes connues et reconnues.

Une des particularités des familles de gènes liées aux COG est que tous les groupes de gènes ont une occurrence d'un gène dans au moins trois espèces distantes. Nous avons défini $\hat{\Sigma}$ comme l'ensemble des familles de gènes avec pour chacune d'elle une occurrence d'un gène dans au moins deux génomes différents. Nous avons donc l'inclusion des COG dans $\hat{\Sigma}$ et pas l'inverse. $\hat{\Sigma}$ correspond aux familles de gènes optimales, les COG sont la version utilisée et ne correspondent pas exactement à $\hat{\Sigma}$.

La banque de données des COG contient quelques familles de gènes avec beaucoup de gènes paralogues entre eux. Ces groupes de gènes constituant une grande famille de gènes avec beaucoup de gènes paralogues n'ayant pas évolués semblent suspects. Ces groupes de gènes sont certainement une union de familles de gènes avec peut être des intersections non vide entre elles. Nous avons préféré éliminer ces groupes de gènes suspects pour éviter qu'ils ne faussent l'algorithme de prédiction des über-operons. Nous avons éliminé les groupes de gènes avec plus de dix paralogues dans un même génome, ce qui représente moins de cinq pour cent de la base de données des COG.

Identification des opérons

Il existe plus d'une centaine de génomes de procaryotes complètement séquencés mais seul une petite partie d'entre-eux est complètement annotée. Parmi ces centaines de génomes, on dispose principalement de la position de leurs gènes et pour quelques gènes, de leur famille. Les opérons ne sont connus que pour un petit nombre d'espèces très étudiées, dont presque exclusivement *Escherichia coli K12* et des génomes évolutivement proches.

Escherichia coli K12 est certainement le génome le plus étudié et par conséquent le mieux annoté. Ses opérons sont recensés dans la base de données RegulonDB [Salgado et al., 2004]. Pour *Escherichia coli K12*, le nombre d'opérons connus est de l'ordre de 809 alors que RegulonDB en prédit 2 325. Étant donné le peu d'informations disponibles et connues sur les opérons, nous avons décidé d'utiliser une méthode de prédiction.

Il existe plusieurs algorithmes de prédiction des opérons dans les génomes de procaryotes, nous en avons recensés quatre principaux. Une première méthode consiste à rechercher des signaux dans les séquences d'ADN, comme des promoteurs ou des terminateurs de la transcription. Cette méthode, utilisant des modèles de Markov cachés, est spécifique à chaque génome étudié et ne peut pas être appliquée sans analyse préalable des génomes d'intérêt. Une deuxième méthode, utilise les fonctions des gènes et leurs interactions pour prédire des opérons [Zheng et al., 2002]. Cette méthode de prédiction repose sur des informations non disponibles pour la plupart des génomes séquencés et de plus c'est ce type d'information que nous voulons prédire. Une troisième méthode repose sur la co-occurrence des mêmes paires de gènes adjacents dans différents génomes [Ermolaeva et al., 2001]. Nous voulons utiliser les opérons pour prédire des groupes de gènes partageant une structure commune dans divers génomes, utiliser à la base de notre algorithme une méthode de prédiction des opérons basée sur la co-occurrence

des mêmes voisinage est un cercle vicieux et n'est donc pas adapté. Une quatrième et dernière méthode, utilise la distance intergénique pour prédire les opérons [Salgado et al., 2000]. Cette méthode se généralise pour tous les génomes de procaryotes et peut donc être appliquée aux génomes séquencés et peu annotés. Moreno-Hagelsieb et Collado-Vides montrent que chez *Escherichia coli K12*, des gènes adjacents d'un même opéron se chevauchent d'une ou quatre bases et que les gènes adjacents n'appartenant pas à un opéron sont séparés en moyenne par 190 paires de bases, avec un pic à 120 paires de bases. Cette discrimination entre les paires de gènes adjacents d'un même opéron et ceux d'opérons différents se retrouve chez tous les procaryotes avec quelques variations dans les valeurs [Moreno-Hagelsieb and Collado-Vides, 2002]. La même approche à été utilisée dans [Hoon et al., 2004]. Une autre étude a montré que des gènes séparés de moins de 250 paires de bases ont tendance à coder pour des fonctions apparentées [Overbeek et al., 1999], et sont donc *a priori* dans un même opéron.

Nous avons choisi d'utiliser les principes de la dernière méthode car ils se généralisent à tous les procaryotes et ne demandent pas beaucoup d'informations sur l'organisation des génomes. Notre méthode repose sur les caractéristiques suivantes des opérons : deux gènes adjacents sont dans un même opéron s'ils sont sur le même brin d'ADN et s'ils ne sont pas distants de plus de cent paires de bases. Deux gènes adjacents ayant ces caractéristiques sont prédits dans le même opéron sinon ils font partis d'opérons différents.

Nous pouvons nous permettre de ne pas être trop stricts avec notre caractérisation des opérons, car pour détecter des über-operons, nous comparons les opérons de plusieurs génomes, ce qui nous permet d'éliminer des faux positifs.

La répartition des opérons prédits suivant leur taille pour 93 génomes séquencés est donnée figure 2.7. La taille moyenne des opérons par génome est de $2,79$ avec un écart type de $1,04$ entre les génomes. La taille moyenne des opérons de tous les génomes est de $2,41$ gènes avec un écart type de $5,59$. Il y a moins de variation entre les génomes qu'à l'intérieur de ceux-ci.

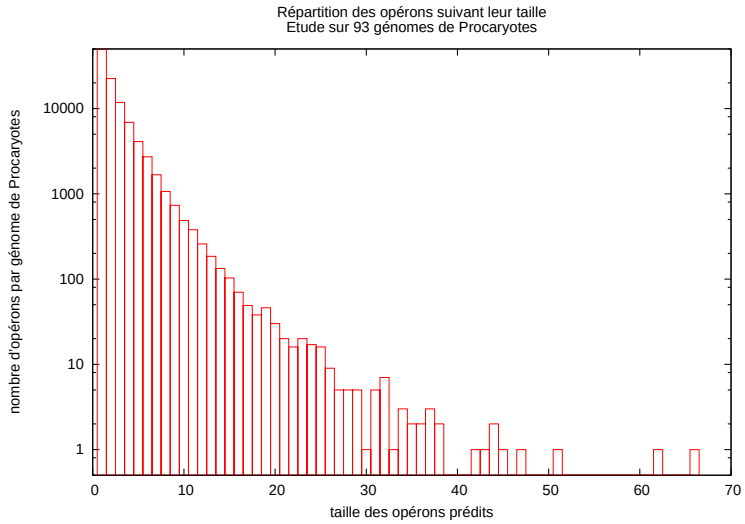


Figure 2.7: Distribution des opérons prédits suivant leur taille

Nous avons vu précédemment que nous nous intéressions aux opérons dans leur intégralité et non aux sous unités de transcription. La méthode de prédiction des opérons utilisée détecte

bien les opérons et non leurs sous unités de transcription, ce qui concorde avec le concept des über-operons.

Sélection des espèces

Les méthodes formelles de recherche de motifs détectent des motifs communs à toutes les séquences comparées. Notre méthode, comme les *Connected Gene Neighborhoods*, autorise la détection de motifs non présent dans toutes les permutations. Alors que pour les *Gene Teams* l'ajout d'une permutation à la comparaison risque de supprimer des clusters, ce problème n'existe pas pour notre méthode. Par contre, nous considérons dans le graphe des opérons tous les génomes de manière équitable, or deux génomes phylogénétiquement proches auront les mêmes opérons et contribueront de la même manière au graphe des opérons. Leurs opérons seront pris en compte, d'une certaine manière, deux fois alors que leur double présence n'est pas due à une contrainte fonctionnelle plus forte, mais à une spéciation récente de ces deux génomes.

Pour éviter de biaiser les über-operons avec des organismes évolutivement trop proches, nous proposons une phase préliminaire à notre algorithme. Le principe est d'éliminer les espèces qui ont divergé trop récemment tout en minimisant le nombre total d'espèces éliminées.

Nous estimerons la divergence de deux espèces par leur similarité au niveau de l'ordre de leurs gènes. Pour cela nous regardons tous les gènes présents dans les deux espèces comparées en travaillant sur des permutations dans $\hat{\Sigma} \cup \bigstar$. On pourrait envisager de comparer les opérons des deux espèces et d'en déduire leur divergence. Cette méthode mesure la divergence fonctionnelle entre ces deux espèces et non leur divergence dans le temps. La divergence dans le temps entraîne un certain nombre de réarrangements, lesquels peuvent modifier les opérons. Un réarrangement modifiant un opéron est plus susceptible de modifier le fonctionnement des mécanismes biologiques de l'espèce, ces réarrangements auront donc tendance à être moins souvent fixés par l'évolution. Deux espèces avec les mêmes opérons mais avec beaucoup de réarrangements implique que ces opérons conservés correspondent à des fonctions et ne peuvent être facilement modifiés. Par conséquent nous devons doublement les prendre en compte dans le graphe des opérons. Il est donc plus judicieux dans notre cas de mesurer la similarité de deux génomes en comparant les paires de gènes adjacents [Sankoff and Blanchette, 1997, Sankoff, 1999] qu'en comparant leurs opérons. Il existe des différences entre nos permutations et celles issus de la modélisation faite par Sankoff et Blanchette. Ces différences sont liées aux duplications, non autorisées dans leur modélisation ou interprétées différemment ; mais aussi parce que nous traitons différemment les gènes n'appartenant pas aux deux génomes, les étoiles. Pour ces raisons nous proposons de nouvelles définitions pour les adjacences.

Nous comparons deux permutations sur $\hat{\Sigma} \cup \bigstar$ où les étoiles sont définies uniquement pour ces deux permutations. Ces permutations autorisent les duplications. Une paire de gènes adjacents d'une permutation π sont deux gènes de $\hat{\Sigma}$ côte à côte dans π. L'ensemble des paires de gènes adjacents de π est noté $\mathcal{A}_\pi$. Pour deux permutations π et σ sur $\hat{\Sigma} \cup \bigstar$, π_1 et π_2 dans $\hat{\Sigma}$, une paire de gènes adjacents de π, forment une adjacence conservée à π et σ, s'il existe dans σ la paire de gènes adjacents $\sigma_{\pi_1}, \sigma_{\pi_2}$. L'ensemble des adjacences conservées de π et σ est noté $\mathcal{A}_{\pi,\sigma}$.

Nous définissons par ailleurs le taux d'adjacences conservées :

$$\tau \quad : \quad \Pi, \Pi \longrightarrow I\!R$$

$$\tau(\pi, \sigma) \quad = \quad \frac{|\mathcal{A}_{\pi,\sigma}|}{|\mathcal{A}_\pi|}$$

Clairement $\tau(\pi, \sigma)$ n'est pas symétrique et par conséquent nous estimerons la divergence entre deux permutations π et σ par la moyenne :

$$\text{sim } (\pi, \sigma) = \frac{\tau(\pi, \sigma) + \tau(\sigma, \pi)}{2}$$

On se fixe un seuil ε tel que pour deux espèces π et σ avec sim $(\pi, \sigma) \geq \varepsilon$, π et σ sont considérées comme deux espèces pas assez divergentes pour être conjointement utilisées dans l'algorithme de détection des über-operons.

Lorsque deux espèces sont évolutivement proches, une des deux doit être éliminée. Ce même principe pour plus de deux espèces devient : étant donné un ensemble d'espèces évolutivement proches, il faut éliminer le minimum d'espèces tel qu'il n'y ait plus de couples d'espèces proches. Ce problème modélisé sous forme de graphe est nommé MIS, *Maximum Independent Set* ou ensemble indépendant maximum. MIS est un des premiers problèmes à avoir été montré NP-complet [Garey and Jonhson, 1979] et on ne peut faire une approximation sur la taille d'un ensemble maximum avec un facteur inférieur à $n^{1-o(1)}$ en temps polynomial. Le meilleur algorithme exact résolvant ce problème est de complexité $O(2^{0.290n})$ en temps et nécessite un espace polynomial [Beigel, 1999].

Définition 2.7 *Problème de l'ensemble indépendant maximum :*

- *en entrée : un graphe non orienté $G = (S, A)$ avec S l'ensemble des sommets et A l'ensemble des arcs.*
- *en sortie : un ensemble maximal de nœuds $S' \subseteq S$ de G tel que pour tout arc $(u, v) \in S' \times S'$, il n'existe pas $(u, v) \in A$.*

Pour résoudre le problème des espèces proches, nous fixons ε et nous définissons le graphe des espèces : $G = (S, A)$ où S est l'ensemble des espèces et A l'ensemble des arcs tel que pour tout $(u, v) \in A$, sim $(u, v) \geq \varepsilon$. La sortie, S', d'un algorithme résolvant MIS, donne un ensemble d'espèces évolutivement distantes.

Nous considérons cette étape comme indépendante de l'algorithme de détection des über-operons. C'est une phase préliminaire, nécessaire une et une seule fois pour un jeu d'espèces donné.

2.4 Vraisemblance des über-operons et représentation

Le paramètre α permet non seulement d'éliminer des gènes insérés épisodiquement dans les opérons, mais surtout il mesure le nombre de fois où les gènes appartiennent aux mêmes opérons.

En faisant varier le paramètre α, on obtient plusieurs clusters de gènes, tous à des niveaux différents de relation entre les gènes. Si on pousse ce principe au maximum en prenant α égal à un, on obtient, avec suffisamment d'espèces, un seul über-operon contenant toutes les familles de gènes. Cet über-operon a globalement un niveau faible d'interaction entre ses gènes mais est un cluster valide car il représente l'ensemble des gènes des procaryotes. Inversement, en prenant pour α la valeur la plus grande possible, on n'obtient que des über-operons constitués d'une seule famille de gènes. Ce niveau de résolution n'apporte pas d'information supplémentaire sur les gènes mais il est bien entendu valable. Entre ces deux bornes, on peut faire varier α et obtenir des clusters de gènes à différents niveaux de résolution. Certains niveaux de résolution peuvent être moins valides que d'autres. Pour déterminer quelles valeurs du paramètre α nous devons utiliser, nous avons besoin de mesurer la vraisemblance des über-operons détectés.

Fitness d'un über-operon

Un über-operon pour un niveau de résolution donné, c'est-à-dire pour le paramètre α donné, est une composante connexe du graphe des opérons où l'on ne prend pas en compte les arcs de poids inférieur à α. Par conséquent, le score que l'on peut associer à un über-operon est aussi le score d'une composante α-connexe. Un über-operon doit avoir un score d'autant plus élevé que ses gènes sont souvent dans les mêmes opérons. De plus, un über-operon est d'autant plus conservé, donc doit avoir un score d'autant plus élevé, qu'il y a peu de gènes insérés. Autrement dit, plus la composante α-connexe est dense, plus le score doit être élevé, et plus la composante α-connexe est isolée du reste du graphe (elle n'est pas complètement isolée à cause du paramètre α) plus le score doit être élevé.

Nous définissons le degré d'un nœud comme la somme des poids de tous les arcs passant par ce nœud :

Définition 2.8 *Pour un graphe non orienté pondéré $G = (S, A, \omega)$, $\deg$, le degré du nœud $u \in S$ est tel que*

$$\deg(u) = \sum_{\{v \in S | (u,v) \in A\}} \omega(u, v).$$

Nous avons décidé d'utiliser la densité relative (δ_r) et la densité locale (δ_l) pour mesurer la force des composantes connexes [Virtanen, 2003].

$$\delta_l(S) = \frac{\deg_{in}(S)}{\sum_{v_i, v_j \in S} max(\deg(v_i), \deg(v_j))}$$

$$\delta_r(S) = \frac{\deg_{in}(S)}{\deg_{in}(S) + \deg_{out}(S)}$$

Où $\deg(v)$ est le degré du nœud v, $\deg_{in}(S) = \sum_{v_i, v_j \in S} w_{v_i, v_j}$ et $\deg_{out}(S) = \sum_{v_i \in S, v_j \notin S} w_{v_i, v_j}$.

δ_r mesure l'isolement de la composante connexe et δ_l mesure la densité des composantes connexes. Nous avons adapté ces mesures pour qu'elles prennent en compte le paramètre α. Le score utilisé, noté f, se sert de l'isolement et de la densité des composantes connexes en en prenant leur produit [Virtanen, 2003].

$$f = \delta_r \times \delta_l$$

Notre score f est tel que moins un über-operon est constitué d'opérons avec des gènes insérés, plus son score sera élevé. Plus un über-operon a ses gènes dans les mêmes opérons pour diverses espèces, plus son score sera élevé.

Calcul de la probabilité d'observer un über-operon

Nous nous intéressons au calcul de la probabilité d'observer un über-operon à partir de l'ensemble des espèces comparées. Pour cela nous calculons la probabilité d'observer une composante α-connexe étant donné l'ensemble des permutations Π' générés aléatoirement à partir de Π. Π' étant un ensemble de séquences aléatoires avec les mêmes familles de gènes que Π, chacune dans la même proportion, mais placées aléatoirement sur les séquences. Les opérons des séquences de Π' sont générés aléatoirement en respectant la distribution des opérons observée sur Π. Après avoir présenté comment nous générons Π', nous déterminons le degré des nœuds du graphe des opérons à partir de la donnée Π' ; puis dans un deuxième temps, nous calculons la probabilité d'observer une composante α-connexe à partir d'un graphe des opérons aléatoire construit à partir de la liste des degrés de ses nœuds.

Construction de Π'

On suppose que tous les génomes comparés sont concaténés en un seul génome géant de taille n gènes. Ce génome géant est vu comme une suite de n emplacements pour des gènes issus de $\hat{\Sigma} \cup \bigstar$. Le nombre total de familles de gènes de ce génome géant est $|\hat{\Sigma}|$. Pour i compris entre 1 et $|\hat{\Sigma}|$, le nombre de fois où est présent la famille de gènes i dans le génome géant est noté n_i. On note p_i la probabilité d'avoir le gène i à une position donnée du génome géant. On suppose que d'observer le gène i en tout point du génome est équiprobable :

$$p_i = \frac{n_i}{n}$$

A cause des $\bigstar$ présentes dans les génomes comparés, la somme pour i compris entre 1 et $|\hat{\Sigma}|$ de tous les p_i est inférieure ou égale à un, plus précisément $\sum_{1 \leq i \leq |\hat{\Sigma}|} n_i = |\hat{\Sigma}| \leq n$.

On suppose les opérons produits par une loi de Bernoulli avec une probabilité p_o d'avoir la fin d'un opéron dans une région intergénique, c'est-à-dire après un gène. La probabilité d'observer un opéron de taille k est donnée par la variable aléatoire Op selon une loi de Bernouilli :

$$P(Op = k) = (1 - p_o)^{k-1} p_o$$

Ce modèle d'opérons a déjà été proposé dans une méthode de prédiction des opérons [Hoon et al., 2004]. Comme tout gène est compris dans un opéron, la fin d'un opéron est aussi le début de l'opéron suivant.

En supposant l'équiprobabilité d'observer la fin d'un opéron dans toutes les parties intergéniques, on peut obtenir p_o à partir des données réelles, où l'on observe n_o operons, de la façon suivante :

$$p_o = \frac{n_o}{n}$$

La figure 2.8 compare la distribution des opérons prédits pour 93 génomes de procaryotes à la loi géométrique proposée ci-dessus. L'effet de la concaténation des génomes pour le calcul de p_o n'est pas trop important par rapport aux données réelles car la moyenne de la taille des opérons est de $2,79$ par génome avec un écart type de $1,04$ entre eux, alors qu'entre tous les opérons de tous les génomes l'écart type de la taille des opérons est de $5,59$.

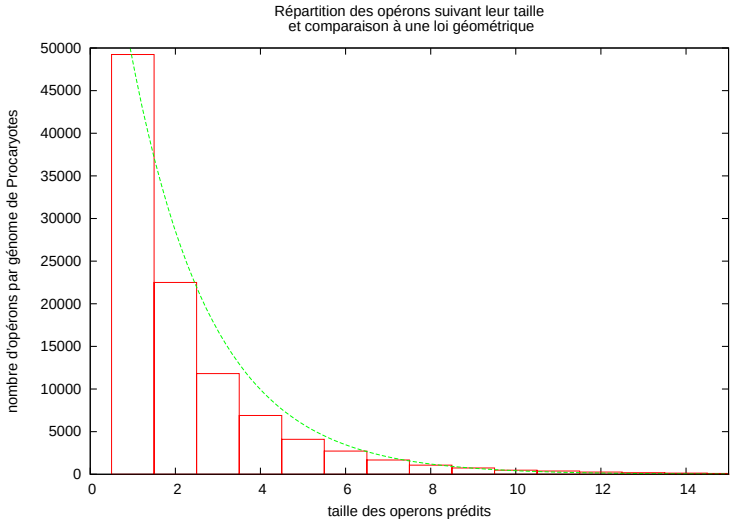


Figure 2.8: Comparaison de la distribution des opérons prédits observés chez 93 procaryotes à leur distribution prédite suivant la variable aléatoire Op. Les barres donnent la distribution observée et la ligne donne la distribution prédite.

L'ensemble des permutations aléatoires Π' issues de l'ensemble des permutations réelles Π, est généré suivant p_o, la probabilité d'avoir la fin d'un opéron dans une partie intergénique, et les p_i, les probabilités d'observer un gène représentant la famille de gènes i à un endroit donné du génome, pour $1 \leq i \leq \left|\hat{\Sigma}\right|$. Ces données sont calculées à partir de Π.

Calcul de la probabilité d'observer une composante α-connexe

Nous noterons d_i le degré attendu du nœud i, représentant la famille de gènes i, dans le graphe des opérons obtenus à partir de Π'.

En omettant les gènes paralogues, pour un opéron o fixé, contenant le gène u, le nombre d'arcs du graphe des opérons passant par le nœud u dû à l'opéron o est $\mid o \mid -1$.

Ne pas prendre en compte les duplications à l'intérieur des opérons, donne pour un opéron o de taille $|o|$ contenant le gène u, avec q gènes uniques, $q \leq |o|$, $q-1$ arcs dûs à o.

Comme le graphe des opérons ne comptabilise qu'une et une seule fois les gènes dupliqués dans un même opéron, nous utiliserons la première formule en faisant l'hypothèse qu'il n'existe pas de duplication d'un gène dans un même opéron pour le calcul de $|o|$. Cette hypothèse est fausse mais elle est souvent vérifiée : par exemple, *Escherichia coli K12* a environ 34% de ses gènes qui sont issus d'une évolution de type paralogue or seulement 3.5% de ses opérons prédits contiennent des gènes paralogues.

Nous calculons maintenant d_i, le degré du nœud i, étant donné Π' : si le gène i est dans un opéron de taille k, alors il faut comptabiliser $k-1$ arcs au nœud i. Le degré du nœud i est donc la somme, pour tous les opérons contenant le gène i, de la taille de ces opérons moins un.

$$\begin{aligned} d_i &= n_o \times \left(\sum_{k\geq 1} (k-1) \times P(Op=k) \times Pr(\text{gène } i \text{ dans l'opéron } O) \right) \\ &= n_o \times \left(\sum_{k\geq 1} (k-1) \times (1-p_o)^{k-1} p_o \times k \times \frac{n_i}{n} \right) \\ &= n_i p_o^2 \times \left(\sum_{k\geq 1} k(k-1) \times (1-p_o)^{k-1} \right) \\ &= n_i p_o^2 \times \frac{2(1-p_o)}{p_o^3} = 2n_i \times \frac{1-p_o}{p_o} \end{aligned}$$

Nous calculons maintenant n_{ij}, le nombre estimé d'arcs (i,j) du graphe des opérons obtenu à partir de génomes aléatoires.

$$n_{ij} = \frac{d_i \times d_j}{\sum_k d_k}$$

n_{ij} est aussi le poids de l'arc (i,j) dans le graphe des opérons, d'ailleurs on obtient d_i à partir de n_{ij} :

$$d_i = \sum_j n_{ij} = \sum_j \frac{d_i d_j}{\sum_k d_k} = d_i \frac{\sum_j d_j}{\sum_k d_k} = d_i$$

Le nombre total d'arcs du graphe des opérons obtenu à partir des génomes aléatoires est :

$$\begin{aligned} m &= \frac{1}{2} \sum_i \sum_j n_{ij} = \frac{1}{2\sum_k d_k} \sum_i (d_i \sum_j d_j) \\ &= \frac{1}{2\sum_k d_k} \sum_i d_i \sum_j d_i = \frac{1}{2} \sum_j d_j = (\frac{1}{p_o} - 1) \sum_i n_i \end{aligned}$$

Finalement, la probabilité d'observer un arc (i,j) dans ce graphe, sachant qu'il contient m arcs, dont n_{ij} arcs reliant i à j, est :

$$p_{ij} = \frac{n_{ij}}{m}$$

p_{ij} est aussi la probabilité d'observer un arc de poids supérieur ou égal à un. On définit la variable aléatoire discrète w_{ij} suivant une loi binomiale de paramètres m et p_{ij}, avec $P(w_{ij} = k)$ la probabilité d'observer un arc (i,j) de poids k.

$$P(w_{ij} = k) = \begin{pmatrix} m \\ k \end{pmatrix} p_{ij}^k (1 - p_{ij})^{(m-k)}$$

Laquelle peut être approximée par une loi de Poisson de paramètre $\lambda = m \times p_{ij} = n_{ij}$.

$$P(w_{ij} = k) = exp(-n_{ij}) \frac{n_{ij}^k}{k!}$$

Or pour toute variable aléatoire X suivant une loi de Poisson : $P(X = k+1) = \frac{\lambda}{\lambda+1} P(X = k)$. Ce qui nous donne pour la probabilité d'avoir un arc (i,j) de poids au moins k :

$$P(w_{ij} \geq k) = 1 - e^{-\lambda} \sum_{0 \leq i < k} \left(\frac{\lambda}{\lambda + 1} \right)^i$$

$P(w_{ij} \geq k)$ est aussi la probabilité d'avoir au moins k fois les gènes i et j dans un même opéron pour un ensemble de génomes aléatoires Π'.

Nous pouvons maintenant calculer la probabilité d'observer au moins une composante α-connexe. C'est la probabilité qu'il existe au moins un arc de poids supérieur à α dans le graphe des opérons :

$$P \leq \sum_{i,j \in S} P(w_{ij} \geq \alpha) \leq \frac{|S|(|S|+1)}{2} \times \max_{i,j \in S} P(w_{ij} \geq \alpha)$$

$P(w_{ij} \geq \alpha)$, pour α fixé, est une fonction monotone croissante. Nous en déduisons que les différences entre les $P(w_{ij} \geq \alpha)$, pour α fixé, dépendent uniquement de n_{ij}.

$$n_{ij} = \frac{n_i n_j \frac{1-p_o}{p_o}}{\sum_k n_k}$$

n_{ij} est également une fonction monotone croissante et les n_{ij} ne se distinguent entre eux que par les n_i et les n_j. Donc pour s_1 le gène de la famille de gènes de taille la plus grande, et pour s_2 le gène de la deuxième famille de gènes de taille la plus grande, $\max_{i,j \in S} P(w_{ij} \geq \alpha) = P(w_{s_1 s_2} \geq \alpha)$. Et finalement nous avons,

$$P \leq \frac{|S|(|S|+1)}{2} \times P(w_{s_1 s_2} \geq \alpha)$$

Nous avons calculé P comme étant la probabilité d'observer une composante α-connexe dans un graphe aléatoire et par conséquent P est aussi la probabilité d'observer au moins un über-operon à partir des séquences aléatoires Π'.

De même, la probabilité d'observer au moins une composante α-connexe de taille au moins trois est :

$$\begin{aligned} P_3 &\leq \sum_{i,j,k \in S} \left\{ \begin{array}{l} P(w_{ij} \geq \alpha) \times P(w_{jk} \geq \alpha) + \\ P(w_{ij} \geq \alpha) \times P(w_{ik} \geq \alpha) + \\ P(w_{ik} \geq \alpha) \times P(w_{jk} \geq \alpha) \end{array} \right. \\ P_3 &\leq \sum_{i,j,k \in S} 3 \times P(w_{s_1 s_2} \geq \alpha) \times P(w_{s_1 s_3} \geq \alpha) \\ &\leq \left(\begin{array}{c} |S| \\ 3 \end{array} \right) \times 3 \times P(w_{s_1 s_2} \geq \alpha) \times P(w_{s_1 s_3} \geq \alpha) \end{aligned}$$

Pour s_1 le nœud de degré le plus grand, s_2 le deuxième nœud de degré le plus grand et pour s_3 le troisième nœud de degré le plus grand.

En généralisant, on obtient une borne pour la probabilité d'observer une composante α-connexe de taille au moins r :

$$P_r \leq \left(\begin{array}{c} |S| \\ r \end{array} \right) \times |T_r| \prod_{\{s_i | i \in [2,r]\}} P(w_{s_1 s_i} \geq \alpha),$$

avec l'ensemble des nœuds $S = \{s_1, ..., s_{|S|}\}$ triés par ordre décroissant de degré, ce qui revient à les trier par ordre décroissant du nombre d'occurrences de la famille de gènes correspondante. Et $|T_r| = r^{r-2}$ le nombre d'arbres couvrants minimum reliant r nœuds. P_r est aussi la probabilité d'observer un über-operon de taille au moins r gènes étant donné des génomes aléatoires générés à partir de l'observation de la répartition des familles de gènes et la distribution des opérons sur les génomes comparés.

Nous pouvons estimer plus précisément la probabilité d'observer une composante connexe donnée. Les résultats ci-dessous sont en partie une adaptation de ceux de [Chung and Lu, 2002]. Soit une composante α-connexe constituée des nœuds $\mathcal{C} = \{s_1, ..., s_k\}$ triés par degré décroissant. La probabilité que cette composante α-connexe soit bien α-isolée est :

$$\prod_{s_i \in \mathcal{C}, s_j \notin \mathcal{C}} (1 - P(w_{ij} \geq \alpha))$$

Soit T un arbre α-couvrant minimal. Nous avons alors $A(T)$ son ensemble d'arcs. On peut alors calculer $P(T)$, la probabilité d'observer T :

$$P(T) = \prod_{(s_i, s_j) \in A(T)} P(w_{ij} \geq \alpha)$$

La probabilité d'avoir une composante α-couvrante est alors au plus :

$$\sum_T P(T) = \sum_T \prod_{(s_i,s_j)\in A(T)} P(w_{ij} \geq \alpha)$$

Or il y a au plus $|T_k|$ arbre α-couvrant composés de k nœuds. Comme précédemment, nous obtenons :

$$\prod_{(s_i,s_j)\in A(T)} P(w_{ij} \geq \alpha) \leq |T_k| \prod_{s\in\{s_2,\ldots,s_k\}} P(w_{s_1,s} \geq \alpha)$$

On obtient finalement pour $P(\mathcal{C})$, la probabilité que les nœuds de l'ensemble $\mathcal{C}$ forment une composante α-connexe et donc un über-operon :

$$P(\mathcal{C}) \leq \prod_{s_i\in\mathcal{C}, s_j\notin\mathcal{C}} (1 - P(w_{ij} \geq \alpha)) \times |T_k| \prod_{s\in\{s_2,\ldots,s_k\}} P(w_{s_1,s} \geq \alpha)$$

Études expérimentales

Le calcul de P_k se décompose en deux étapes : la première consiste à calculer n_{ij} le nombre d'arcs (i,j) obtenus à partir des génomes aléatoires ; la deuxième étape utilise n_{ij} pour calculer $p(w_{ij} \geq \alpha)$ la probabilité d'observer au moins α arcs (i,j) et P_k une borne supérieure de la probabilité d'observer une composante α-connexe de taille au moins k gènes.

Nous montrons dans un premier temps l'adéquation entre les n_{ij} observés à partir de génomes aléatoires et ceux calculés. Nous générons un génome géant de taille $n = 50 \times 1\,000$ gènes correspondant à la concaténation de 50 génomes de taille 1000 en moyenne. Pour ce génome géant, nous générons les opérons suivant une loi de Bernouilli avec $p_o = 0.3$. Nous plaçons alors sur le génome géant de manière équiprobable n_i gènes i et n_j gènes j. Nous construisons alors le graphe des opérons et nous pouvons compter le nombre d'arcs (i,j). Nous répétons cette expérience 10 000 fois pour différentes valeurs de n_i et de n_j. Nous comparons alors la moyenne du nombre d'arcs (i,j) observés avec la valeur théorique calculée. Les résultats sont dans le tableau suivant :

n_i	50	100	150	200	250	300	350	400	450	500
n_j	200	200	200	200	200	200	200	200	200	200
n_{ij} observée	0.93	1.82	2.73	3.64	4.55	5.45	6.32	7.19	7.97	8.90
n_{ij} calculée	0.93	1.86	2.80	3.73	4.66	5.60	6.53	7.46	8.40	9.33

Nous faisons remarquer que les expériences avec n_i plus grand que 300 correspondent à des cas limites où nous considérons plus de 6 duplications d'un même gène dans chacun des 50 génomes. Le calcul de n_{ij} donne des résultats légèrement supérieurs aux observations. Cela est dû aux duplications dans un même opéron qui ne sont pas prises en compte dans le modèle. Le nombre d'arcs calculé est donc logiquement supérieur au nombre d'arcs observés.

Nous montrons maintenant l'erreur commise par la borne P_k par rapport aux observations réelles. Nous construisons maintenant un graphe aléatoire à partir de la donnée des d_i, le

degré des nœuds i. Nous construisons un graphe aléatoire composé de 100 nœuds, le degré de chaque nœud est choisi aléatoirement et de manière équiprobable entre 1 et 15. À partir de cette donnée, on calcule pour chaque couple d'arcs (i, j) sa probabilité d'avoir un poids égal à α. On crée un arc (i, j) de poids k en suivant cette probabilité. Nous pouvons alors observer le nombre de composantes α-connexes de taille au moins r. r est un paramètre donné. Nous répétons cette expérience 5 000 000 de fois afin d'estimer la fréquence moyenne des composantes α-connexes de taille au moins r gènes. Nous comparons alors ce nombre moyen avec la probabilité P_r. Les résultats sont exprimés dans les tableaux ci-dessous.

P_r observées

$r\backslash\alpha$	3	4	5
2	6.72e-01	4.53e-02	1.66e-03
3	4.25e-02	9.35e-05	1.6e-07
4	2.40e-03	2e-07	0
5	1.54e-04	2e-08	0
6	1.15e-05	0	0
7	9.4e-07	0	0
8	8e-08	0	0
9	0	0	0

P_r calculées

$r\backslash\alpha$	3	4	5
2	1	5.70e-01	2.70e-02
3	1	6.44e-04	1.45e-05
4	4.62e-01	9.59e-05	1.02e-08
5	1.34e-01	1.66e-06	8.39e-12
6	4.31e-02	3.13e-08	7.53e-15
7	1.46e-02	6.29e-10	7.16e-18
8	5.16e-03	1.31e-11	7.09e-21
9	1.87e-03	2.82e-13	7.23e-24

Rapport observées/calculées

$r\backslash\alpha$	3	4	5
2	6.71e-01	7.95e-02	6.15e-02
3	4.25e-02	1.45e-02	1.10e-02
4	5.18e-03	2.08e-03	×
5	1.14e-03	1.21e-02	×
6	2.67e-04	×	×
7	6.43e-05	×	×
8	1.55e-05	×	×
9	×	×	×

Représentation hiérarchique

Nous avons vu précédemment qu'un über-operon est un ensemble d'opérons partageant le même contenu en gènes. De plus, les gènes d'un même opéron partagent la même corégulation et dans la plupart des cas, codent pour des fonctions apparentées. Un über-operon est donc formé, par transitivité, d'un ensemble de gènes codant pour des fonctions apparentées.

La représentation usuelle associant une protéine à une fonction déterminée, limite notre perception des interactions géniques. Les protéines ont une multitude de fonctions dépendant de leur contexte [Shrager, 2003]. Elles doivent être considérées dans des groupes et pas seulement indépendamment les unes des autres. C'est la représentation habituelle sous forme de clusters de gènes. Mais il n'existe pas qu'un seul groupe de gènes lié à une protéine mais bien souvent plusieurs. Les protéines forment des complexes et une même protéine peut appartenir à plusieurs complexes distincts.

[2]Par exemple l'utilisation d'EC (*Enzyme Commission number*).

Une même protéine peut aussi appartenir à plusieurs complexes sans pour autant qu'ils soient indépendants les uns des autres, ce sont les niveaux hiérarchiques des clusters de gènes. Nous avons déjà vu l'exemple du ribosome, formé de deux sous complexes. Il existe d'autres exemples. Reprenons le mécanisme du flagelle. Le flagelle est constitué de protéines formant un filament. Ces protéines correspondent à un niveau d'interaction. Si on ajoute les protéines assurant la fixation de ce filament aux membranes de la bactérie, on obtient un nouveau complexe qui peut-être vu indépendamment ; mais ces deux complexes associés, en forment un troisième qui représente un autre niveau d'interaction.

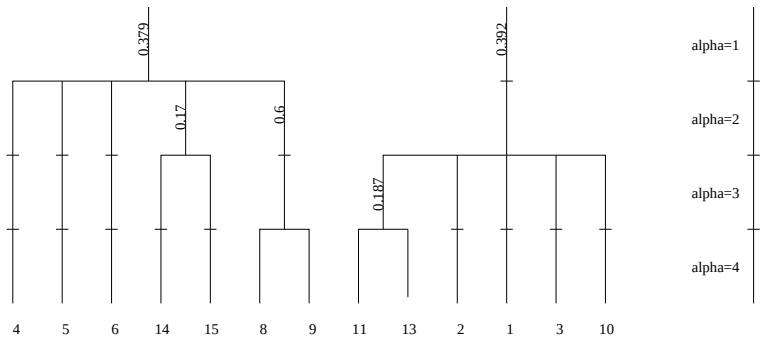
D'un premier point de vue, le paramètre α de notre algorithme permet d'éliminer les gènes insérés dans les opérons. Mais d'un autre point de vue, le paramètre α représente la connexité des über-operons et donc la force des liens entre les gènes. La force des liens traduit les niveaux hiérarchiques entre les complexes de protéines. Le paramètre α permet non seulement d'éliminer les gènes insérés mais aussi de détecter différents niveaux hiérarchiques.

Nous proposons d'utiliser le paramètre α pour identifier les différents niveaux d'interactions entre les gènes. Le paramètre α fixé à son minimum donne un seul über-operon associant presque l'intégralité des gènes. Les gènes de cet über-operon sont apparentés mais à un niveau hiérarchique très élevé. Ce niveau représente l'ensemble des gènes constituant les génomes de procaryotes. Inversement, on peut choisir pour α la valeur la plus élevée possible et n'obtenir que des gènes isolés. Entre ces deux valeurs extrêmes, l'utilisation de α révèle les niveaux hiérarchiques intermédiaires. Et pour chacun de ces niveaux, on peut évaluer sa pertinence en utilisant le score des composantes α-connexes vu précédemment.

La figure 2.5 page 55 représentant un graphe des opérons donne lieu à une représentation hiérarchique de ses über-operons. Les über-operons associés aux trois génomes sont :

$\{11, 13\}$ et $\{8, 9\}$	pour $\alpha = 3$	$P_2 \leq 0.36$
$\{1, 2, 3, 10, 11, 13\}$, $\{8, 9\}$ et $\{14, 15\}$	pour $\alpha = 2$	$P_2 \leq 0.58$
$\{1, 2, 3, 10, 11, 13\}$ et $\{4, 5, 6, 8, 9, 14, 15\}$	pour $\alpha = 1$	$P_2 \leq 1$

La représentation hiérarchique des ces über-operons est la suivante :



Tout sous arbre de la représentation hiérarchique a ses feuilles qui forment un cluster de gènes. La profondeur des nœuds représente les valeurs du paramètre α pour lesquels on a détecté le cluster de gènes correspondant. Sur les branches est donné le score des composantes connexes. Plus un nœud est haut placé dans l'arbre, plus le groupe de gènes correspondant

à ses feuilles a un niveau hiérarchique élevé. Et inversement, plus le nœud est proche des feuilles, plus le niveau hiérarchique de ses feuilles est petit. En opposition, quand le niveau hiérarchique est bas, les liens entre les gènes sont forts, et quand le niveau hiérarchique est élevé, les liens entre les gènes sont faibles.

Détermination du α minimal

La représentation hiérarchique des über-operons permet de visualiser les groupes des gènes mais aussi leurs sous-groupes. Mais pour un gène donné quel est son niveau hiérarchique le plus élevé ? C'est-à-dire quelle est la valeur du paramètre α représentant la racine ? En effet chaque nœud peut-être une racine potentielle. Si nous ne fixons pas de seuil à α alors tous les gènes seraient dans le même groupe de gènes. Le paramètre α d'une branche, dans la représentation hiérarchique, donne le nombre d'opérons contenant conjointement les gènes représentés par ses feuilles. La question précédente revient à se demander à partir de combien d'opérons considère-t-on la branche de l'arbre suffisamment informative ?

Une des manières de répondre est de fixer un seuil universel pour α. A partir d'un gène donné, sa représentation hiérarchique est obtenue en prenant l'arbre le contenant et dont la racine correspond à la branche qui à ce seuil pour valeur de α. Cette représentation hiérarchique pour ce gène est unique.

Certains gènes sont moins stables et bougent plus facilement dans les génomes. Avec la méthode du seuil fixe, il est possible que de tels gènes ne soient associés à aucun autre gène. La méthode des über-operons et la représentation hiérarchique ne permettraient pas d'obtenir d'information sur la fonction de ces gènes. Si ces gènes mobiles restent toujours dans le même contexte, c'est-à-dire que leurs voisins ne sont jamais les mêmes mais que ces gènes voisins appartiennent invariablement à un et un seul ensemble de gènes, alors ces gènes correspondent à la définition des über-operons. Dans ce cas précis, la méthode du seuil n'est pas conforme aux über-operons.

Pour obtenir la représentation hiérarchique complète, il faut se baser sur la connexité du graphe des opérons. Le α représentant la racine d'un arbre doit dépendre de la connexité des gènes, issus des feuilles de cet arbre, dans le graphe des opérons à cette racine. Pour mesurer cette connexité nous proposons d'utiliser le score f.

Pour déterminer la représentation hiérarchique associée à un gène, on prend le sous-arbre le contenant tel que sa racine soit la plus proche et de score le plus important. Pour un gène g donné nous commençons par le sous-arbre composé d'une seule feuille. Ce sous-arbre sera noté t_i. La racine de cet arbre nous donne un score f. On considère maintenant le sous-arbre t_{i+1} dont la racine est le père de la racine du sous arbre t_i. Tant que le score de la racine de t_{i+1} est supérieur à celui de t_i, l'arbre t_{i+1} devient la représentation hiérarchique du gène g. On réitère le procédé tant que la condition sur le score est vérifiée.

Ce processus nous permet de fixer les différentes racines des arbres en fonction des gènes de départ. La racine pour deux gènes donnés ne correspondra pas toujours à une même valeur du paramètre α. Et d'ailleurs les représentations hiérarchiques peuvent être incluses les unes dans les autres. La racine, et donc le α minimal, dépend de la connexité du graphe. Tant qu'ajouter des gènes au groupe de gènes augmente le score de connexité, on considère que le groupe de gènes est valide.

Dans l'exemple de la figure 2.5 page 55 et 71 la représentation hiérarchique du gène 8 a une racine correspondant au paramètre $\alpha = 2$. Celle du gène 14 a une racine correspondant au paramètre $\alpha = 1$.

2.5 Les Applications

Nous présentons maintenant des applications de la nouvelle définition des über-operons que nous avons proposée. Il est difficile de montrer objectivement le bien fondé d'une méthode de détection de groupes de gènes car il n'existe pas de données exhaustives sur les groupes de gènes avec des fonctions apparentées. Nous ne pouvons pas avec une méthode automatique donner un score aux groupes de gènes détectés pour mesurer leur vraisemblance.

Pour montrer l'intérêt des über-operons et de notre analyse, nous proposons des exemples de groupes de gènes détectés avec notre méthode. Nous nous intéresserons d'abord à un exemple facilement identifiable par toutes les méthodes de détection de groupes de gènes conservés. Cet exemple, basé sur le mécanisme du phosphate, montre que notre approche avec la représentation hiérarchique donne plus d'information que les autres méthodes. Un deuxième exemple, sur les mécanismes du chimiotactisme, montre quant à lui que nous ne devons pas *a priori* sélectionner les espèces à comparer. Et finalement un troisième exemple lié aux mécanismes du flagelle présente un groupe de gènes avec lequel aucune autre méthode ne peut rivaliser ; tant par le nombre de gènes associés entre eux que par la représentation hiérarchique de ceux-ci.

Sélection des espèces

Avant d'appliquer les principes des über-operons nous devons préparer nos données. La base de données des COG nous donne des annotations pour 63 espèces de procaryotes. Parmi ces 63 espèces, une n'est pas correctement annotée, car les noms des gènes ne correspondent pas. Nous avons aussi éliminé deux espèces comportant plus d'un chromosome. Nous avons donc des informations pour 60 génomes : *Archaeoglobus fulgidus, Halobacterium sp. NRC-1, Methanosarcina acetivorans str.C2A, Methanothermobacter thermautotrophicus, Methanococcus jannaschii, Methanopyrus kandleri AV19, Thermoplasma acidophilum, Thermoplasma volcanium, Pyrococcus horikoshii, Pyrococcus abyssi, Pyrobaculum aerophilum, Sulfolobus solfataricus, Aeropyrum pernix, Aquifex aeolicus, Thermotoga maritima, Synechocystis, Nostoc sp. PCC 7120, Fusobacterium nucleatum, Deinococcus radiodurans, Corynebacterium glutamicum, Mycobacterium tuberculosis H37Rv, Mycobacterium tuberculosis CDC1551, Mycobacterium leprae, Clostridium acetobutylicum, Lactococcus lactis, Streptococcus pyogenes M1 GAS, Streptococcus pneumoniae TIGR4, Staphylococcus aureus N315, Listeria innocua, Bacillus subtilis, Bacillus halodurans, Escherichia coli K12, Escherichia coli O157 :H7 EDL933, Escherichia coli O157 :H7, Yersinia pestis, Salmonella typhimurium LT2, Buchnera sp. APS, Vibrio cholerae, Pseudomonas aeruginosa, Haemophilus influenzae, Pasteurella multocida, Xylella fastidiosa 9a5c, Neisseria meningitidis MC58, Neisseria meningitidis Z2491, Ralstonia solanacearum, Helicobacter pylori 26695, Helicobacter pylori J99, Campylobacter jejuni, Sinorhizobium meliloti, Brucella melitensis, Mesorhizobium loti, Caulobacter vibrioides, Rickettsia prowazekii, Rickettsia conorii, Chlamydia trachomatis,*

Chlamydophila pneumoniae CWL029, Treponema pallidum, Borrelia burgdorferi, Ureaplasma urealyticum, Mycoplasma pulmonis, Mycoplasma pneumoniae et Mycoplasma genitalium.

Nous appliquons maintenant le principe de sélection des espèces suffisamment distante (voir section 2.3 page 61). Pour cela nous avons arbitrairement choisi pour seuil $\varepsilon = 0.2$. Nous obtenons 6 couples d'espèces proches : (*Chlamydophila pneumoniae CWL029* et *Chlamydia trachomatis*), (*Escherichia coli K12* et *Escherichia coli O157:H7 EDL933*), (*Actinobacteria Mycobacterium tuberculosis CDC1551* et *Actinobacteria Mycobacterium tuberculosis H37Rv*), (*Neisseria meningitidis Z2491* et *Neisseria meningitidis MC58*), (*Helicobacter pylori 26695* et *Helicobacter pylori J99*), et (*Mycoplasma genitalium* et *Mycoplasma pneumoniae*). On peut traiter le problème MIS de manière exacte car les 6 couples sont indépendants les uns des autres. On peut de manière optimale éliminer *Chlamydia trachomatis*, *Escherichia coli O157:H7 EDL933*, *Actinobacteria Mycobacterium tuberculosis H37Rv*, *Neisseria meningitidis MC58*, *Helicobacter pylori 26695* et *Mycoplasma pneumoniae*. Nous obtenons finalement un ensemble de 54 espèces suffisamment distantes.

Les calculs de probabilités effectués sur les données réelles sont donnés en annexe page 170.

HUGO

Nous avons développé un logiciel mettant en œuvre les principes exposés dans ce chapitre sur les über-operons. Nous avons nommé ce logiciel HUGO pour Union Hiérarchique des Gènes des Opérons (*Hierarchical Union of Genes from Operons*). Ce logiciel est disponible sur le réseau Internet pour les systèmes d'exploitation Windows et ceux de type Unix.

Le mécanisme du régulon phosphate

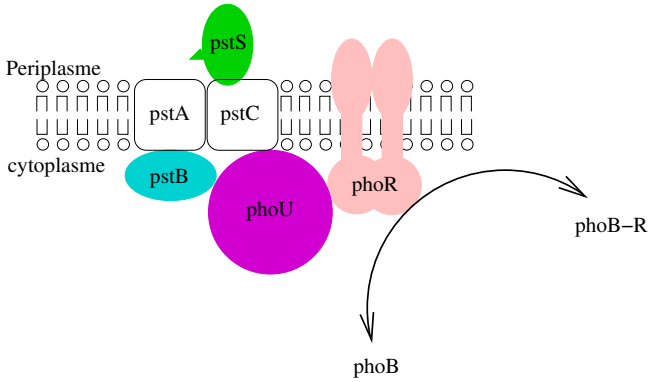
Le premier groupe de gènes apparentés du point de vue fonctionnel auquel nous nous intéressons est lié au mécanisme du phosphate. Le phosphate est la forme sous laquelle le phosphore peut-être assimilé par les êtres vivants. Le phosphore est un composant des acides nucléiques et interagit dans de nombreuses réactions enzymatiques. Chez les procaryotes il permet la récupération, l'accumulation et la distribution de l'énergie dans la bactérie. Pour résumer nous considérerons qu'il permet la croissance des bactéries. Il est principalement incorporé sous forme de phosphate inorganique.

Nous voulons détecter, à l'aide d'HUGO, un über-operon lié au mécanisme du phosphate. Mais pour cela nous sommes limités par les données que nous avons en entrée de notre algorithme. Et plus particulièrement par les COG, la base de données donnant les familles de gènes. Les gènes inclus dans le mécanisme du phosphate inorganique chez *Escherichia coli K12*, dont les fonctions sont connues et présentes dans les COG, sont : phoA, phoH, phnCDEFGHIJKLMNP, pstABCS et phoU [Wanner, 1996].

- Le gène phoA produit l'alkaline phosphatase une des formes du phosphate ;
- le gène phoH est lié aux mécanismes de l'énergie ;
- les gènes phnCDEFGHIJKLMNP correspondent au mécanisme de la dégradation et de l'assimilation du phosphonate (un dérivé de l'acide phosphoré) ;
- les gènes pstA et pstC induisent deux protéines membranaires impliquées dans le transport du phosphate inorganique ;

- le gène pstB produit une protéine impliquée dans les mécanismes d'énergies liés au transport du phosphate ;
- le gène pstS code pour la protéine de fixation du phosphate inorganique ;
- et le gène phoU est probablement le lien entre le système du transport du phosphate inorganique et le système PhoBR de régulation.

La figure 2.9 représente les interactions entre les gènes pstABCS et phoU.



Signal (Phosphate inorganique)	$\longrightarrow$	Contrôle (PstABCS-PhoU)	$\longrightarrow$	Capteur (PhoR)	$\longrightarrow$	Régulateur (PhoB)

Figure 2.9: Le mécanisme de détection de la présence de phosphate inorganique chez *Escherichia coli K12*. Le phosphate inorganique vient se fixer sur la protéine PstS, celle-ci le fait passer dans le cytoplasme via les protéines membranaires PstA et PstC. Le mécanisme utilise l'énergie proposée par pstB. PhoU transmet alors cette information à phoR qui avec phoB permet de réguler l'expression de PhoA.

Détection de clusters de gènes avec HUGO

HUGO permet de détecter un über-operon comportant cet ensemble de gènes. La représentation hiérarchique de l'über-operon est donnée figure 2.10. Détecter ce groupe de gènes conservés n'est pas d'une grande difficulté. En effet, l'opéron pstABCS-phoU est conservé dans la plupart des génomes bactériens comme le montre les valeurs élevées du paramètre α. Par contre, la représentation hiérarchique permet d'observer les différents niveaux de relation entre les protéines issues des gènes. On observe premièrement le lien très fort entre les deux protéines membranaires pstA et pstC. Deuxièmement, les trois gènes pstABC sont liés ensembles. Puis troisièmement, les gènes pstS et phoU leur sont aussi associés. Ces deux gènes ont un rapport moins fort avec les autres gènes car pstS sert à capter le phosphate inorganique de l'environnement et phoU sert de lien entre ces gènes et le système phoBR.

HUGO détecte aussi un groupe de gènes représentant le système de transport de l'acide phosphoré : phnCDE. Trois autres groupes de gènes peuvent être identifiés, ce sont phnMN et phnGHI et phnJK. Les fonctions de ces gènes ne sont pas bien connues et on ne peut donc pas conclure sur leur regroupement.

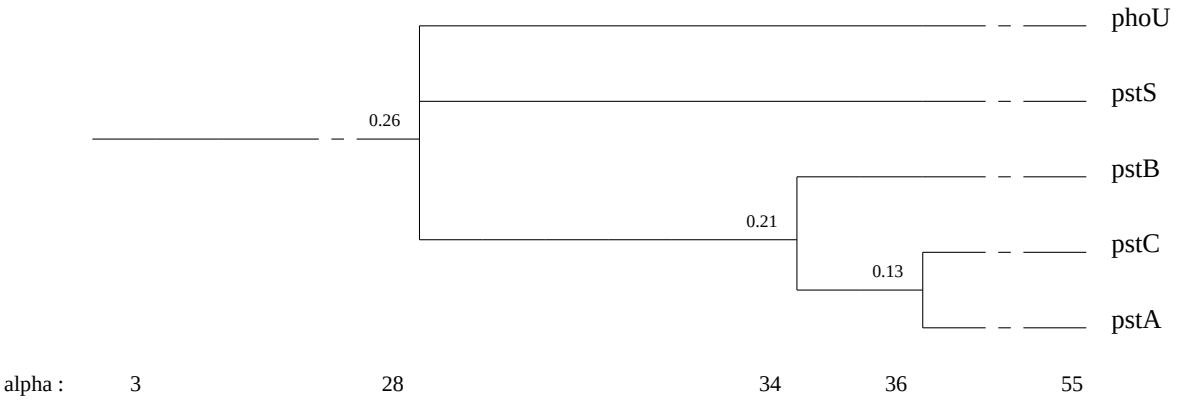


Figure 2.10: L'über-operon lié au mécanisme du phosphate : pstABCS-phoU.

Détection de clusters de gènes avec les Gene Teams

Les Gene Teams appliqués directement à notre jeu de données ne donnent aucun résultat. D'après les COG, il n'existe aucun gène orthologue à l'ensemble des 54 espèces sélectionnées. Pour comparer les méthodes nous devons préalablement sélectionner les espèces que nous allons comparer. Les seules espèces parmi les 60 disponibles qui contiennent les gènes pstABCS et phoU sans duplication de ceux-ci sont aux nombres de 8 : *Aquifex aeolicus*, *Clostridium acetobutylicum*, *Actinobacteria Corynebacterium glutamicum*, *Staphylococcus aureus N315*, *Bacillus halodurans*, *Escherichia coli K12*, *Xylella fastidiosa 9a5c*, et *Borrelia burgdorferi*. Pour ce jeu de données on obtient le groupe de gènes pstACS pour des valeurs de δ allant de zéro à cent. Par contre en sélectionnant parmi ces huit espèces les quatre suivantes : *Escherichia coli K12*, *Bacillus halodurans*, *Staphylococcus aureus N315* et *Actinobacteria Corynebacterium glutamicum*, on obtient le groupe de gènes complet : pstABCS et phoU.

Détection de clusters de gènes avec les Connected Gene Neighborhoods

Le groupe de gènes donné par les CGN sur leur jeu de données de 31 espèces est :

```
pstS COG0226 ABC-type phosphate transport system, periplasmic component
oui  COG0406 Phosphoglycerate mutase/fructose-2,6-bisphosphatase
     COG0477 Permeases of the major facilitator superfamily
     COG0494 NTP pyrophosphohydrolases including oxidative damage repair enzymes
     COG0500 SAM-dependent methyltransferases
pstB COG0573 ABC-type phosphate transport system, permease component
pstA COG0581 ABC-type phosphate transport system, permease component
oui  COG0642 Sensory transduction histidine kinases
     COG0643 Chemotaxis protein histidine kinase and related kinases
phoU COG0704 Phosphate uptake regulator
oui  COG0745 Response regulators consisting of a CheY-like receiver domain
             and a HTH DNA-binding dom (phoP)
     COG0834 Periplasmic amino acid binding proteins
     COG0835 Chemotaxis signal transduction protein
     COG0840 Methyl-accepting chemotaxis protein
pstB COG1117 ABC-type phosphate transport system, ATPase component
```

```
COG1309 Transcriptional regulator
COG1352 Methylase of chemotaxis methyl-accepting proteins
COG1776 Chemotaxis protein CheC, inhibitor of MCP methylation
COG1871 Chemotaxis protein; stimulates methylation of MCP proteins
COG2060 K+-transporting ATPase, A chain
COG2156 K+-transporting ATPase, c chain
COG2199 Diguanylate cyclase/phosphodiesterase domain 1 (GGDEF)
COG2200 Diguanylate cyclase/phosphodiesterase domain 2 (EAL)
COG2201 Chemotaxis response regulator CheB, consists of CheY-like
        receiver domain and a methyle
COG2205 Osmosensitive K+ channel His kinase sensor domain
COG2207 AraC-type DNA-binding domain-containing proteins
COG2216 High-affinity K+ transport system, ATPase chain B
```

Les cinq gènes pstABCS et phoU sont bien regroupés ensemble mais avec d'autres gènes. Les autres gènes sont annotés par un "oui" quand il y a un rapport entre ces gènes et les mécanismes du phosphate. Parmi les 27 gènes de ce groupe de gènes, il reste beaucoup d'incertitude sur le bien fondé du regroupement de certains.

Le mécanisme du chimiotactisme

Nous avons vu précédemment le mécanisme des flagelles. Ceux-ci permettent aux bactéries qui en sont équipées de se déplacer si leur environnement est pauvre en une certaine substance. Les flagelles sont constitués de trois parties, un filament hélicoïdal, un crochet et un corpuscule basal. Le corpuscule ou corps basal correspond au lieu d'insertion du flagelle dans la cellule. Les bactéries détectent dans leur milieu des gradients de concentration et s'approchent ou s'éloignent du gradient le plus élevé. Ce mécanisme de détection est appelé le chimiotactisme.

Les bactéries détectent donc des gradients de concentration grâce à des protéines membranaires appelées protéines du chimiotactisme acceptrices de méthyle (MCP, Methyl-accepting Chemotaxis Protein). Les MCP sont constituées des protéines Tap, Tar, Tip, Trg et Tsr correspondant pour chacune d'entres elles à des attractifs différents. Les attractifs se fixent et modifient les MCP. Les signaux sont ensuite transmis dans le cytoplasme jusqu'au moteur flagellaire par une suite de réactions. Cette cascade est constituée des protéines CheA, CheB, CheR, CheW, CheY et CheZ. Les interactions entre ces protéines sont données figure 2.11.

Le gène tip n'est pas présent dans les COG et les gènes tap, tar, trg et tsr ont été éliminés des COG car ils comportaient trop de paralogues. Ces derniers gènes sont d'ailleurs identifiés comme des gènes paralogues.

Détection de clusters de gènes avec HUGO

HUGO nous permet de retrouver un cluster de gènes comportant les gènes cheABCDRW (voir figure 2.12). Les gènes cheC et cheD sont présents chez les archaes et quelques bactéries. La représentation hiérarchique nous permet d'isoler le sous groupe de gènes cheA et cheW. Ces deux gènes forment bien un sous groupe car ils constituent une unité fonctionnelle qui répond à l'état de signalisation des chimiorécepteurs. De même cheB et cheR sont regroupés ensemble

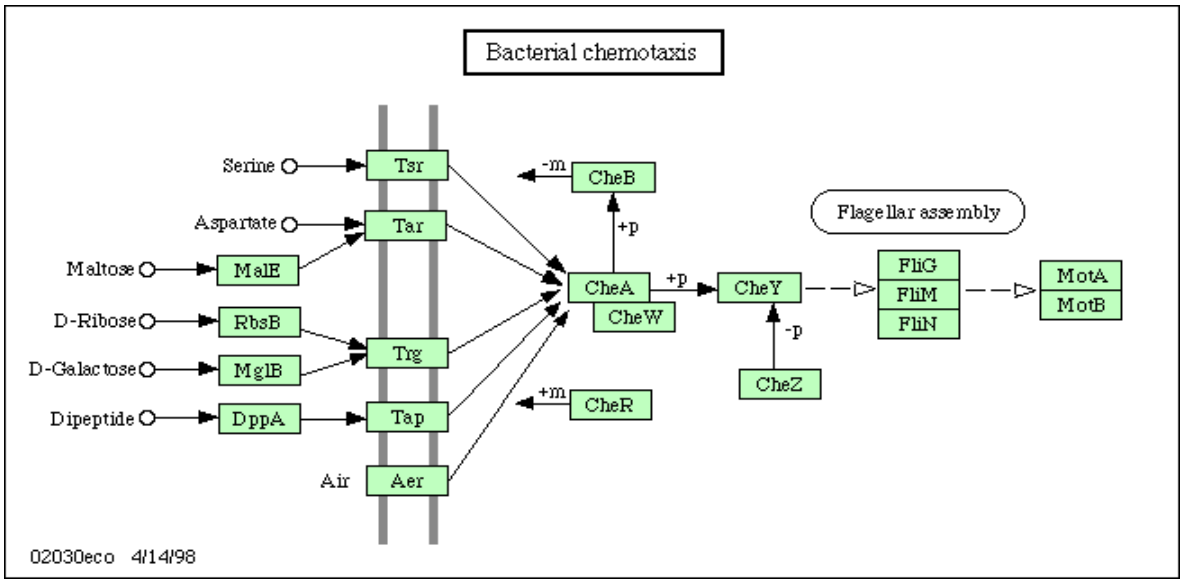


Figure 2.11: Le mécanisme du chimiotactisme chez escherichia coli K12. Figure provenant de KEGG.

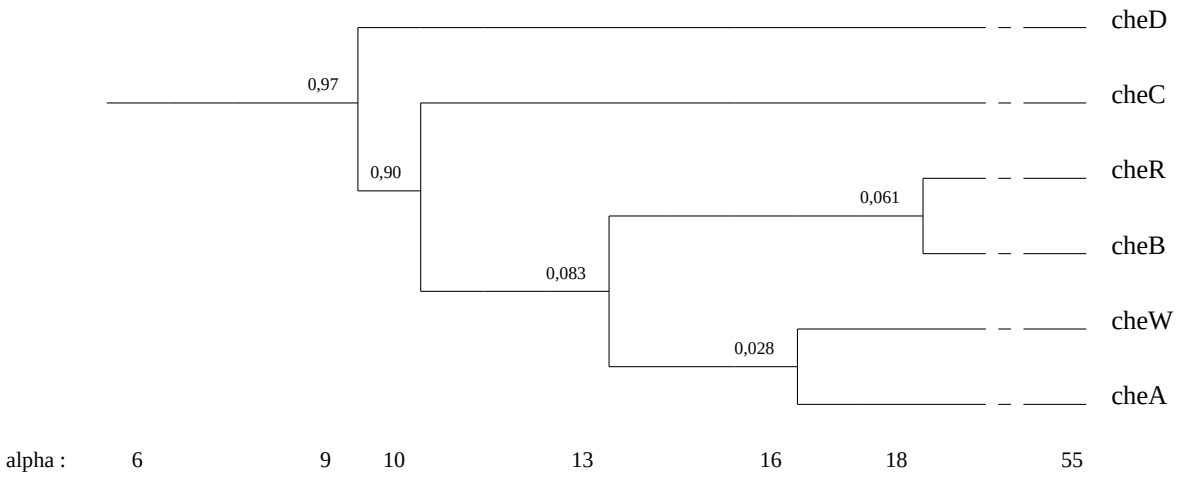


Figure 2.12: L'über-operon lié au mécanisme du chimiotactisme.

et ce sont les deux enzymes qui contrôlent l'état des MCP. Il manque dans cet über-operon les deux gènes cheY et cheZ qui ne sont compris dans aucun über-operon. Ces deux gènes sont donc très mobiles.

Il est intéressant de noter que cet über-operon est facilement détectable par une autre méthode comme les Gene Teams si on sélectionne les espèces qui ont ces gènes présents et dans un même opéron. Mais nous avons réussi à le détecter sans connaître *a priori* les espèces à sélectionner.

Détection de clusters de gènes avec les Gene Teams

Pour appliquer les Gene Teams nous devons, de même que précédemment, sélectionner les espèces contenant les gènes cheABRWYZ. Aucune espèce des COG ne contient un et un seul

exemplaire de tous ces gènes. Si on décide d'éliminer le gène cheY, on obtient un ensemble de quatre espèces : *Escherichia coli K12*, *Escherichia coli O157:H7*, *Escherichia coli O157:H7 EDL933* et *Yersinia pestis*. Ce sont quatre espèces très proches et la détection de groupes de gènes communs à ces quatre espèces n'est pas très informative. On retrouve les gènes cheABRWZ associés ensemble avec les Gene Teams pour des valeurs de δ allant de dix à cinquante.

Détection de clusters de gènes avec les Connected Gene Neighborhoods

Les CGN donnent le même groupe de gènes que précédemment :

```
pstS COG0226 ABC-type phosphate transport system, periplasmic component
     COG0406 Phosphoglycerate mutase/fructose-2,6-bisphosphatase
     COG0477 Permeases of the major facilitator superfamily
     COG0494 NTP pyrophosphohydrolases including oxidative damage repair enzymes
     COG0500 SAM-dependent methyltransferases
pstB COG0573 ABC-type phosphate transport system, permease component
pstA COG0581 ABC-type phosphate transport system, permease component
     COG0642 Sensory transduction histidine kinases
cheA COG0643 Chemotaxis protein histidine kinase and related kinases
phoU COG0704 Phosphate uptake regulator
oui  COG0745 Response regulators consisting of a CheY-like
              receiver domain and a HTH DNA-binding dom (phoP)
     COG0834 Periplasmic amino acid binding proteins
cheW COG0835 Chemotaxis signal transduction protein
oui  COG0840 Methyl-accepting chemotaxis protein
pstB COG1117 ABC-type phosphate transport system, ATPase component
     COG1309 Transcriptional regulator
cheR COG1352 Methylase of chemotaxis methyl-accepting proteins
cheC COG1776 Chemotaxis protein CheC, inhibitor of MCP methylation
cheD COG1871 Chemotaxis protein; stimulates methylation of MCP proteins
     COG2060 K+-transporting ATPase, A chain
     COG2156 K+-transporting ATPase, c chain
     COG2199 Diguanylate cyclase/phosphodiesterase domain 1 (GGDEF)
     COG2200 Diguanylate cyclase/phosphodiesterase domain 2 (EAL)
cheB COG2201 Chemotaxis response regulator CheB, consists of CheY-like
              receiver domain and a methyle
     COG2205 Osmosensitive K+ channel His kinase sensor domain
     COG2207 AraC-type DNA-binding domain-containing proteins
     COG2216 High-affinity K+ transport system, ATPase chain B
```

Le mécanisme du flagelle

Le mécanisme du flagelle est un mécanisme complexe permettant aux bactéries qui en sont équipées de se déplacer. D'après la base de données KEGG, *Kyoto Encyclopedia of Genes and Genomes* [Ogata et al., 1999], le flagelle et les mécanismes permettant le déplacement des bactéries sont constitués d'une trentaine de gènes : `fliC`, `fliD`, `fliE`, `fliF`, `fliG`, `fliH`,

`fliI, fliJ, fliK, fliM, fliN, fliO, fliP, fliQ, fliR, fliS, flgA, flgB, flgC, flgD, flgE, flgF, flgG, flgH, flgI, flgK, flgL, flgM, flgN, motA, motB, flhA, flhB, flhC, flhD et flhT.`

Détection de clusters de gènes avec HUGO

HUGO nous permet de détecter encore une fois un groupe de gènes lié au mécanisme du flagelle. Non seulement nous avons regroupé des gènes directement liés au flagelle mais aussi des gènes liés aux mécanismes du système de sécrétion de type III (voir figures 2.3 et 2.13). Ce dernier mécanisme interagit d'ailleurs avec les mécanismes du flagelle. La représentation hiérarchique calculée par HUGO est donnée figure 2.14.

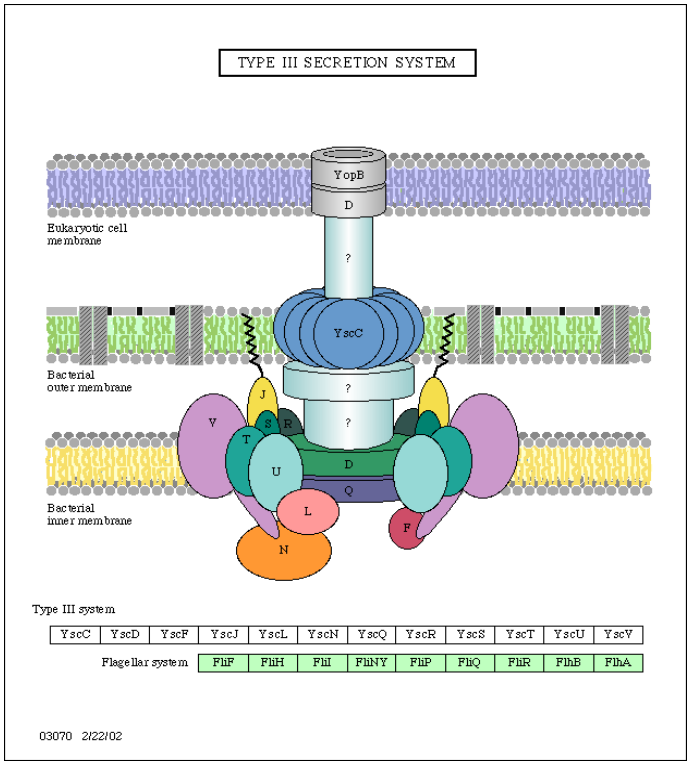


Figure 2.13: Le système de sécrétion de type III chez les bactéries est lié au mécanisme du flagelle. Figure provenant de KEGG.

L'über-operon détecté par HUGO non seulement regroupe 34 gènes liés au flagelle mais aussi 5 gènes du système de sécrétion de type III. L'über-operon forme ainsi un groupe de 39 gènes. On observe bien dans la représentation hiérarchique le regroupement des gènes liés au système de sécrétion de type III et celui des gènes liés plus spécifiquement au flagelle. Il est intéressant de noter que si on baisse α de un, nous obtenons la représentation hiérarchique de la figure 2.15. Nous avons alors toujours les 39 gènes précédents mais aussi 6 gènes du chimiotactisme et 6 autres gènes en rapport avec le flagelle. On remarque aussi le regroupement des gènes motA et motB avec ceux du chimiotactisme. Or le chimiotactisme influence MotA et MotB, les moteurs des flagelles. De plus, les gènes dont les protéines interagissent étroitement ensembles sont regroupés, comme flhAB (groupe A sur la figure 2.15), flgBC (groupe C), flgFGHI (groupe F) et d'autres encore.

Détection de clusters de gènes avec les Gene Teams

Nous ne savons pas *a priori* quelles espèces sélectionner pour les Gene Teams. Pour les expérimentations précédentes, le nombre de gènes recherché étant petit, il était facile de choisir un sous-ensemble des espèces contenant ces gènes. Dans cet exemple, le nombre de gènes liés aux mécanismes du flagelle est tellement important que la moindre sélection d'espèces, élimine *ipso facto* une grande partie des espèces.

Pour évaluer les Gene Teams avec la plus grande impartialité, nous présentons deux exemples. Le premier utilise les quatre espèces cités dans la section 2.1 lorsque nous avons introduit le concept des über-operons. Ces quatre espèces, *Escherichia coli K12*, *Bacillus subtilis*, *Clostridium acetobutylicum* et *Yersinia pestis* sont relativement similaires et contiennent une grande partie des gènes liés aux mécanismes du flagelle. Dans un second temps, nous étudions les résultats donnés par les Gene Teams pour deux génomes de la même souche : *Escherichia coli K12* et *Escherichia coli O157:H7 EDL933*.

Les Gene Teams avec l'exemple constitué de quatre espèces donnent les clusters de gènes suivants : {`fliEFGHPQ`}, {`fliDS`}, {`tors cheAW`}, {`fliJL`} et {`flgBCDG`}. Ces cinq ensembles de gènes comptent un total de 17 gènes liés aux mécanismes du flagelle. Nous avons utilisé des valeurs allant de 11 à 50 pour le paramètre δ. Il est évident que si on ajoute encore des espèces à la comparaison, les groupes de gènes seront de plus en plus morcelés. Les prédictions des Gene Teams ne sont pas mauvaises mais sont limitées par le nombre de gènes associés ensemble.

Nous nous intéressons maintenant à la comparaison de deux génomes évolutivement très proches, *Escherichia coli K12* et *Escherichia coli O157:H7*. Pour $\delta = 2$, on obtient les cinq groupes de gènes {`fliADS`} , {`fliEFGH,fliJKLM,fliOPQR`}, {`cheBRZ`}, {`cheAW,motA`} et {`yceH,mviN,flgMN,flgABCDEFGHI,flgK`}. Alors que pour $\delta = 3$, on obtient les quatre groupes de gènes : {`yedO,fliADS`}, {`fliEFGH,fliJKLM,fliOPQR`}, {`cheABRWZ,motA`} et {`pyrC,grxB,yceH,mviN,flgMN,flgABCDEFGHI,flgK`}. Ces groupes de gènes restent à peu près stable jusqu'à $\delta = 5$, puis ils incorporent beaucoup d'autres gènes. Du fait du peu d'espèces comparées simultanément, le paramètre δ ne peut être fortement augmenté.

Les résultats obtenus dans la deuxième expérimentation sont intéressants bien que ces gènes soient regroupés dans des sous-ensembles indépendants. De plus, il est clair qu'étant donné le peu d'espèces comparées et parce que celles-ci sont évolutivement très proches, on ne peut avoir de certitude sur la véracité des objets détectés.

Détection de clusters de gènes avec les Connected Gene Neighborhoods

Les CGN donnent six groupes de gènes relatifs aux mécanismes du flagelle : {`COG0455 fliALMNPQR flhAB COG1419 COG1582 cheC flgD`}, {`fliEFGHIJ flgBC`}, {`COG0630 COG1955 COG2874 COG3351 COG3352 COG3353`}, {`COG1334 fliCDS`} ,{`flgEDHI`} et {`flgKL COG1699`}. Ces groupes de gènes forment des regroupements valides du point de vue biologique. Mais, contrairement à HUGO, les CGN n'ont pas été capables d'associer ces groupes de gènes entre-eux.

2.6 Résumé et conclusions

Nous avons représenté le concept des über-operons introduit dans [Lathe et al., 2000]. Nous avons retravaillé ce concept pour y introduire les insertions et délétions de gènes. Le concept des über-operons est maintenant clairement défini.

Nous avons défini les über-operons de deux manières. Premièrement du point de vue biologique : un über-operon est un ensemble de gènes formé par un ensemble maximal d'opérons partageant des gènes orthologues, des gènes paralogues et avec quelques gènes insérés ou délétés. Dans un deuxième temps, nous les avons défini du point de vue mathématique : un über-operon est une composante α-connexe (maximale) du graphe des opérons. Nous avons proposé la définition mathématique avec l'optique qu'elle soit la plus proche de la définition biologique.

À partir de la redéfinition de ce concept, nous avons proposé une méthode simple et efficace de détection des über-operons. Pour cette méthode, nous avons essayé de raisonner en dehors des sentiers battus des méthodes formelles de détection de clusters de gènes. Ces méthodes sont, de notre point de vue, trop strictes pour être appliquées aux procaryotes. En parallèle, nous nous sommes efforcés de ne pas faire les mêmes erreurs que la méthode descriptive des CGN. Notre algorithme ne repose pas sur une série de filtres et le résultat final est facilement compréhensible. De plus, la complexité algorithmique de notre méthode la rend parfaitement praticable. Nous avons finalement montré avec des exemples l'intérêt de notre méthode de détection de clusters de gènes, basé sur le concept des über-operons, par rapport aux autres méthodes.

Par rapport à la définition de clusters de gènes que nous avons proposés aux début de ce chapitre, nous avons introduit avec HUGO le principe des clusters hiérarchiques. Cette représentation permet d'identifier les différents niveaux de relation entre les gènes. De plus, les clusters de gènes détectés par notre méthode ne doivent pas être présents sur tous les génomes comparés. Nous autorisons aussi les duplications, les insertions et les délétions de gènes.

Finalement, nous avons montré des exemples réels de clusters de gènes détectés par notre méthode. Les autres méthodes, soit donnent des résultats après un prétraitement réalisé à la main en connaissance de cause, soit donnent des résultats erronés.

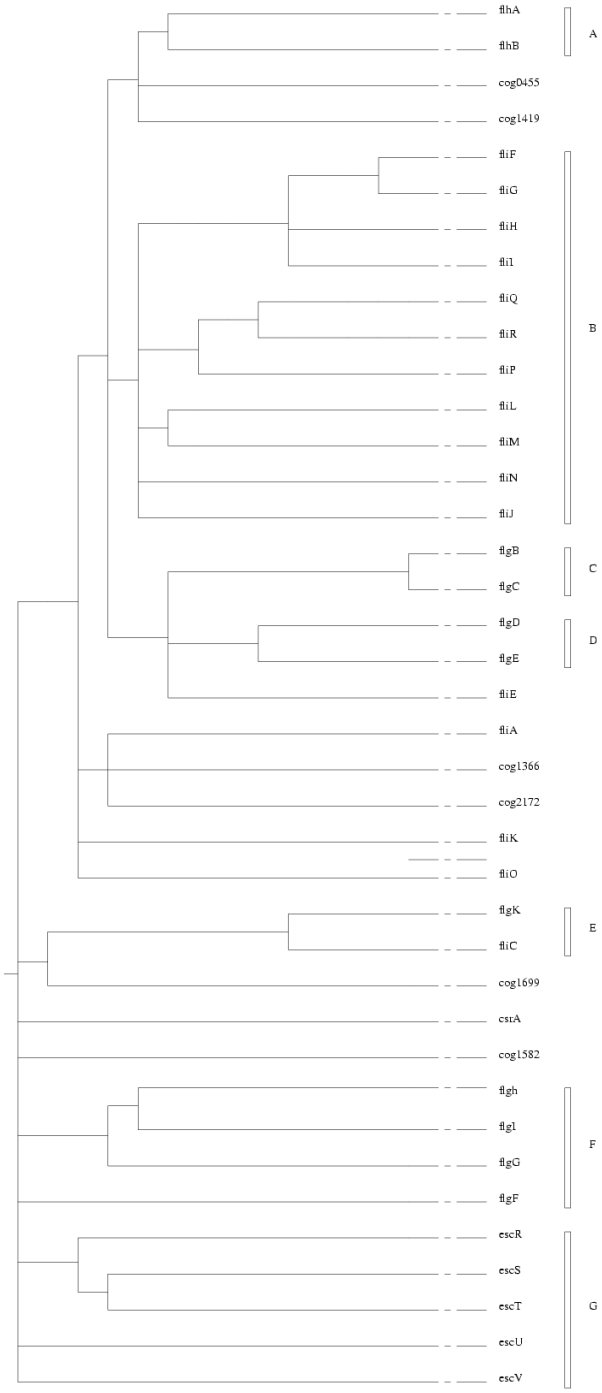


Figure 2.14: L'über-operon lié au mécanisme du flagelle.

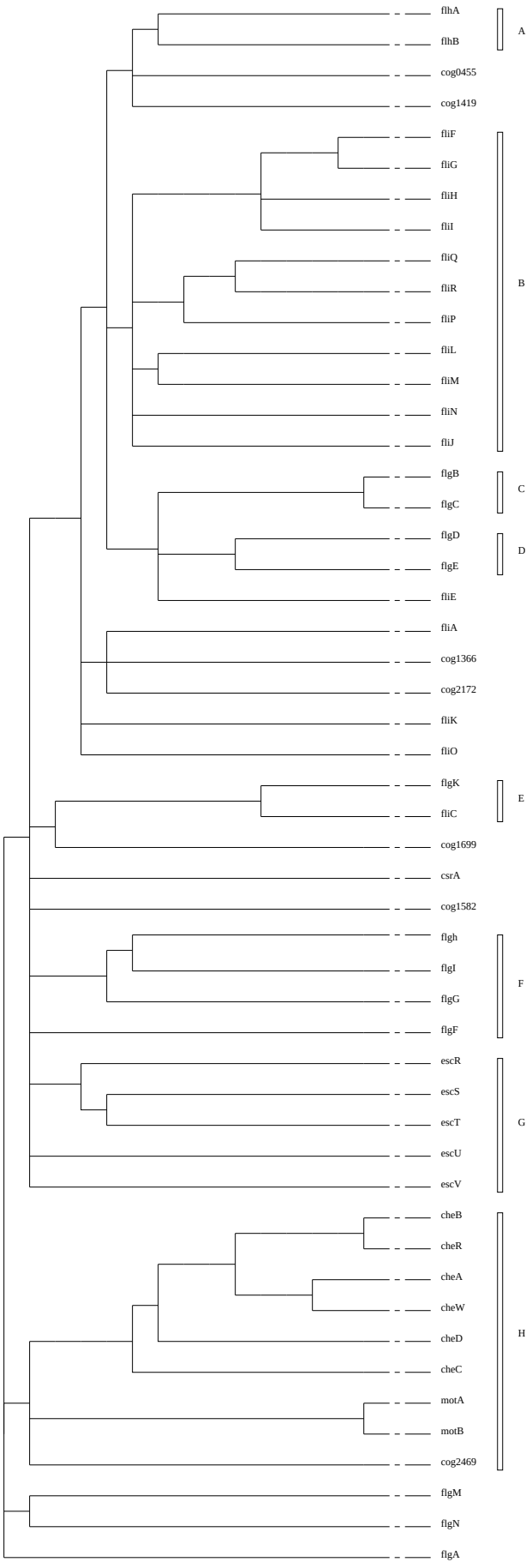


Figure 2.15: Un groupe de gènes lié au mécanisme du flagelle, du chimiotactisme et du système de sécrétion de type III.

chapitre 3

Détection de méta-clusters

Nous avons présenté dans le chapitre précédent une méthode de détection de clusters de gènes basée sur le concept des über-operons. Les über-operons se démarquent des autres méthodes de détection de clusters de gènes car ils ne sont pas construits à partir des adjacences conservées mais des opérons. Nous présentons dans ce chapitre, une nouvelle méthode de détection des clusters de gènes, basée sur l'algorithme proposé pour la détection des über-operons mais construite sur les adjacences. Nous avons nommé cette méthode, les méta-clusters. Les méta-clusters sont effectivement des clusters de gènes, mais ils existent à un niveau méta dans les génomes : ils forment un ensemble de gènes avec des fonctions apparentées sans pour autant être toujours regroupés dans les génomes. En cela, ils sont proches des CGN car ils peuvent très bien ne pas exister réellement dans les génomes.

Les méta-clusters permettent de détecter des clusters de gènes en prenant en compte les duplications ; mais surtout, ils identifient des clusters de gènes communs à un sous ensemble des génomes et non communs à tous les génomes. L'intérêt des méta-clusters à ne pas reposer sur les opérons est d'être adapté à tous les organismes vivants (Eucaryotes, mitochondries, etc).

Dans un premier temps, nous présentons la méthode de détection proposée. Puis nous présentons de même que pour les über-operons, quelques calculs sur la vraisemblance des objets identifiés.

Nous ne présentons pas d'expérimentation car il est très difficile, pour l'instant, d'obtenir des informations sur les familles de gènes pour suffisamment d'espèces d'eucaryotes.

3.1 Présentation de la méthode

Les méthodes telles que les Gene Teams sont très efficaces du point de vue algorithmique mais pêchent par leur définition de clusters. Tout d'abord, ils n'autorisent pas les duplications alors que celles-ci représentent environ 30% des gènes des génomes de procaryotes. Les Gene Teams permettent de rechercher des clusters de gènes communs à un ensemble de génomes. Mais leur définition est trop stricte car elle oblige les clusters détectés à être présents sur tous les génomes comparés. L'algorithmique efficace mise en place est intrinsèquement liée a cette propriété. Pour résoudre ces problèmes, nous présentons les méta-clusters, mais comme nous le verrons, la méthode n'est pas parfaite et amène des artefacts.

L'hypothèse de base de toutes les méthodes de recherche de clusters de gènes basées sur les réarrangements génomiques est que des gènes adjacents codent pour des fonctions apparentées. Plus on retrouve ces adjacences sur plusieurs génomes, plus cette hypothèse est renforcée. Les réarrangements auraient dû avec le temps casser ces adjacences, or si elles sont conservées, c'est qu'il y a un bénéfice quelconque pour les organismes vivant à leur présence.

A partir de cette constatation basique, on peut construire une méthode naïve de détection de clusters de gènes. La méthode consiste à associer, entre eux, les gènes qui sont adjacents dans suffisamment de génomes. On ne recherche pas des suites de gènes conservées dans le même ordre, mais bien des ensembles de gènes partageant des adjacences.

La définition formelle en découle. Nous reprenons d'abord les définitions sur les familles de gènes. Une famille de gènes est un ensemble de gènes codant pour une fonction identique. Soit Σ l'ensemble des familles de gènes. Toute famille de gènes présente dans une seule espèce sera nommée $\bigstar$ (étoile). Soit $\hat{\Sigma}$ l'ensemble de toutes les familles de gènes présentes au moins deux fois dans l'ensemble des espèces. $\hat{\Sigma}$ est relatif aux espèces mises en jeu. Un génome est représenté par une liste ordonnée sur $\Sigma \cup \{\bigstar\}$, appelée permutation. Soit $\Pi = \{\pi^i\}$ pour $i \in [1, n]$ l'ensemble des n permutations représentant les n génomes d'intérêts. Nous construisons un graphe non orienté pondéré $G = (S, A, \omega)$ nommé, graphe des adjacences. S, l'ensemble des sommets, est identique à $\hat{\Sigma}$ et il existe un arc $a = (s_1, s_2, \omega_{s_1,s_2})$ si les familles de gènes représentés par les nœuds s_1 et s_2 sont exactement ω_{s_1,s_2} fois adjacents. Une définition naïve de clusters de gènes, que nous nommons les clusters d'adjacences, est la suivante :

Définition 3.1 *Un cluster d'adjacences est une composante α-connexe (maximale) du graphe des adjacences.*

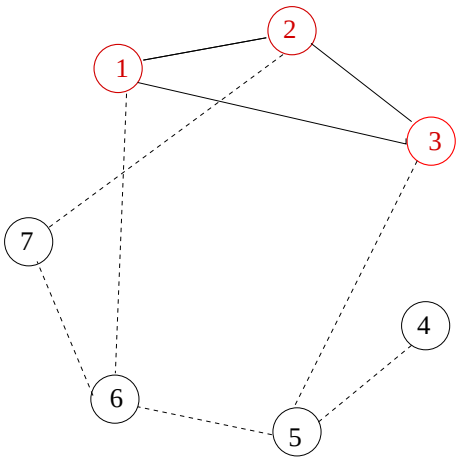
Pour la définition des composantes α-connexe, le lecteur est invité à se référer au chapitre précédent, section 2.2.

Les duplications ne posent pas de problèmes et sont traitées indépendamment les une des autres. Les duplications donnent plusieurs adjacences possibles pour une même famille de gènes, ce qui représente plusieurs possibilités. Quand on a deux gènes paralogues et adjacents, on obtient alors un nœud connecté sur lui même, ce qui n'interagit pas avec les composantes α-connexes du graphe des adjacences.

La méthode présentée précédemment est loin d'être parfaite. En effet, elle ne permet pas toujours de détecter des objets aussi simple que les Intervalles communs. Prenons par exemple ces trois permutations :

$$\begin{array}{lcccccccc} \pi^1 : & 1 & 2 & 3 & \bigstar & 4 & 5 & 6 & \\ \pi^2 : & 2 & 1 & 3 & 5 & \bigstar & 7 & & \\ \pi^3 : & 7 & 2 & 3 & 1 & 6 & 7 & \bigstar & 4 \end{array}$$

Le cluster de gènes $\{1, 2, 3\}$ mis en avant par la couleur rouge, forme un Intervalle commun en omettant les $\bigstar$ et les gènes non communs aux trois permutations. Le graphe des adjacences associé à ces trois permutations est le suivant :



En pointillés sont représentés les arcs de poids égal à un, les arcs plein sont quant à eux de poids égal à deux. Si on fixe α à deux, on obtient bien le cluster de gènes $\{1, 2, 3\}$. Mais si on augmente la valeur du paramètre α à trois, le clusters de gènes $\{1, 2, 3\}$ n'est plus détecté, alors que les gènes $1, 2$ et 3 sont bien regroupés trois fois ensemble dans les permutations. Pour remédier à cette limitation du graphe des adjacences, nous introduisons quelques définitions sur les graphes :

Définition 3.2 *une chemin dans un graphe $G = (S, A)$ est une suite de sommets distincts, $s_1, ..., s_n$ de S, tel que pour tout $1 \leq i < n$, $(s_i, s_{i+1}) \in A$.*

Une manière de palier à ce problème est de renforcer certains arcs. Sur le graphe des adjacences, on observe effectivement que les gènes 1,2 et 3 forment une clique. Une telle structure dans notre graphe est importante et permet d'identifier un cluster de gènes. En effet on peut passer du gène 1 au gène 2 soit directement par l'arc $(1, 2)$, soit indirectement par le chemin $1, 3, 2$. Le lien entre le gène 1 et le gène 2 existe donc trois fois : deux fois par un lien direct $(1, 2)$ et une fois par un lien indirect $(1, 3)(3, 2)$. Prendre pour le poids des arcs le nombre de liens directs ou indirects, *i.e.* le nombre de chemins, au lieu du nombre d'adjacences, permet d'identifier, même pour α égal à trois, la composante α-connexe $\{1, 2, 3\}$. Nous pouvons remarquer qu'il existe deux sortes de chemins : ces chemins existent non seulement dans le graphe des adjacences mais surtout dans les permutations.

Définition 3.3 *Un chemin dans une permutation π sur $\hat{\Sigma} \cup \bigstar$ est une sous suite de gènes $g_1, ..., g_n$ de $\hat{\Sigma}$ incluse dans π.*

Définition 3.4 *Un γ-chemin dans une permutation π sur $\hat{\Sigma} \cup \bigstar$ est une sous suite de gènes $g_1, ..., g_n$ de $\hat{\Sigma}$ incluse dans π tel qu'il existe une position de g_1 et une position de g_n avec $\text{POS}_\pi(g_n) - \text{POS}_\pi(g_1) \leq \gamma$.*

Un chemin dans une permutation donne lieu au même chemin dans le graphe mais la réciproque est fausse. Alors qu'un chemin dans le graphe ne représente aucune information réelle, un chemin dans une permutation représente une suite de gènes réellement présente dans cette ordre dans un des génomes ; ce qui est une information bien réelle. L'utilisation des chemins au lieu des adjacences permet d'associer deux gènes pas directement adjacents et permet ainsi de renforcer ou d'ajouter certains arcs au graphe des adjacences.

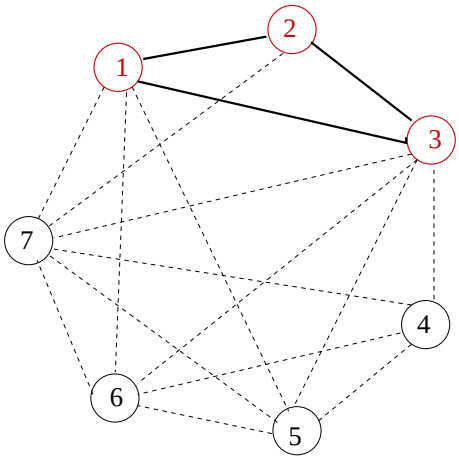
Pour γ fixé, Soit $\hat{\Sigma}$ l'ensemble de toutes les familles de gènes présentes au moins deux fois dans l'ensemble des espèces. $\hat{\Sigma}$ est relatif aux espèces mises en jeu. Un génome est représenté par une liste ordonnée sur $\Sigma \cup \{\bigstar\}$, appelée permutation. Soit $\Pi = \{\pi^i\}$ pour $i \in [1, n]$ l'ensemble des n permutations représentant les n génomes d'intérêts. Nous construisons un graphe non orienté pondéré $G = (S, A, \omega)$ nommé, graphe des γ-chemins, avec S l'ensemble des sommets qui est identique à $\hat{\Sigma}$. Et il existe un arc $a = (s_1, s_2, \omega_{s_1,s_2})$ si et seulement s'il existe ω_{s_1,s_2} fois des γ-chemins dans les permutations reliant s_1 à s_2.

Définition 3.5 *Pour γ fixé, un méta-cluster est une composante α-connexe (maximale) du graphe des γ-chemins.*

Nous opposons les γ-chemins des permutations aux γ-chemins des graphes :

Définition 3.6 *un γ-chemin dans un graphe $G = (S, A)$ est une suite de sommets distincts, $s_1, ..., s_n$ de S, pour $n \leq \gamma$ tel que pour tout $1 \leq i < n$, $(s_i, s_{i+1}) \in A$.*

Pour γ égal à deux, il existe dans le graphe des adjacences de l'exemple précédent, deux γ-chemins (du graphe) de 1 vers 5 et deux autres de 3 vers 6. Les gènes $\{1, 2, 3\}$ sont regroupés ensemble pour $\alpha = 2$ qu'on utilise les chemins ou les adjacences. Or d'après les chemins cités plus haut, si on utilisait les chemins du graphe des adjacences, les gènes $\{1, 2, 3, 5, 6\}$ formeraient une composante α-connexe pour $\alpha = 2$. Mais $\{1, 2, 3, 5, 6\}$ ne forment pas un cluster de gènes et ne sont pas souvent dans le même voisinage car pour γ égal à deux, il n'existe qu'un seul chemin dans les permutations reliant 1 à 5, 1 à 6, 3 à 5, 3 à 6, 2 à 5 et 2 à 6. Le graphe des γ-chemins pour γ égal à deux est le suivant :



Les arcs en pointillés sont de poids égal à un alors que les arcs pleins et en gras sont de poids égal à trois. Donc pour α égal à deux ou trois et γ égal à deux, on obtient exclusivement le méta-cluster $\{1, 2, 3\}$.

Il existe des cas, ou des clusters de gènes comme les Intervalles communs ne peuvent être détectés par la méthode des méta-clusters. Mais dès qu'il existe suffisamment de génomes avec ces Intervalles communs, les méta-clusters permettent de les détecter. En dehors de cette objection, les méta-clusters prennent en compte les duplications et surtout, ils permettent de détecter des clusters de gènes présent dans un sous ensemble des génomes.

Le paramètre γ permet de relier deux gènes séparés par au plus $\gamma - 1$ autres gènes. En apparence, le paramètre γ ressemble au paramètre δ des Gene Teams, lequel autorise les insertions de gènes. Mais en pratique, le paramètre γ est tout autre car il sert surtout à renforcer les arcs du graphe des adjacences en le transformant en graphe des chemins. Le graphe des chemins, permet entre autre, d'avoir des insertions de gènes dans les clusters identifiés.

Le problème inhérent à cette méthode est que les clusters ne sont pas forcément réels. Par rapport aux méta-clusters, les über-operons sont composés d'opérons qui eux existent et sont des objets biologiques. Les über-operons existent donc vraiment dans les génomes ce qui n'est pas le cas des méta-clusters.

Note sur les duplications

Les duplications de gènes, lorsque les gènes sont distants d'au plus γ gènes, font surestimer le nombre de liens, donc d'arcs, associés à ces gènes. Pour éviter ce phénomène, les gènes dupliqués sont traités différemment des autres gènes. Pour deux gènes g_1 et g_2 issus d'une duplication, s'ils sont dans le même γ-chemin d'une permutation, des arcs issus de ce gène g_1 et de ceux issus de ce gène g_2, il ne faut comptabiliser qu'une et une seule fois les arcs comptés deux fois.

Supposons un γ chemin de la permutation π contenant g_1 et g_2. Sans traitement spécifique des duplications, le gène g_1 dans la permutation π génère les arcs $A_1 = \{(g_1, g_i), ..., (g_1, g_{i+\gamma-1})\}$ pour ce γ-chemin, et le gène g_2 dans la permutation π génère les arcs $A_2 = \{(g_2, g_j), ..., (g_2, g_{j+\gamma-1})\}$ pour ce γ-chemin. Les arcs g_1 et g_2 générent pour ce γ-chemin les arcs $A_1 \cup A_2$. Pour ces deux ensembles d'arcs générés, certains se recouvrent,

car g_1 et g_2 font parties de la même famille de gènes. Nous proposons de prendre en compte spécifiquement les duplications à l'intérieur d'un même γ-chemin et les arcs générés par g_1 et g_2 sont alors : $\{(A_1 \cup A_2) \setminus (A_1 \cap A_2)\}$.

Analyse de l'algorithmique

De même que pour les über-operons, nous discutons ici du cœur de l'algorithme des méta-clusters, en supposant que les permutations sont déjà construites. On peut construire le graphe des chemins de la façon suivante : on parcourt chaque permutation, pour chacune de celle-ci, on la parcourt de gauche à droite et pour chaque gène $g_i \in \hat{\Sigma}$ rencontré, on construit les au plus γ arcs reliant g_i aux (au plus) γ gènes de $\hat{\Sigma}$ à sa droite.

La figure 3.1 montre le traitement des gènes dupliqués. On suppose que les arcs sont ajoutés en parcourant les séquences de gauche à droite. Les ronds noirs représentent un gène dupliqué, les ronds blancs des gènes quelconques sans duplication et les étoiles représentent les ★. Les arcs de cercle représentent les arcs ajoutés au graphe des chemins. De haut en bas sont décrites les différentes étapes pour ajouter les arcs en lisant les permutations de gauche à droite. À la première étape, il n'y a pas de gènes dupliqués pour ce γ-chemin, on ajoute donc pour $1 \leq j \leq \gamma$, tous les arcs (g_i, g_{i+j}). Dans les étapes suivantes, le γ-chemin contient deux fois le même gène, donc la première occurrence est interprétée comme une étoile.

Figure 3.1: Prise en compte des duplications pour les méta-clusters. Les ronds noirs représentent un gène dupliqué, les ronds blancs des gènes quelconques sans duplication et les étoiles représentent les ★.

Pour n permutations, chacune de taille au plus m gènes, la construction du graphe des chemins se fait donc en $O(n\gamma m)$. Et comme la recherche de composantes connexes se fait en temps linéaire par rapport au nombre d'arcs du graphe des chemins, la complexité finale de l'algorithme de détection des méta-clusters est en $O(n\gamma m)$.

3.2 Vraisemblance des méta-clusters et représentation

Le score et la représentation hiérarchique des über-operon peuvent être directement utilisés pour mesurer et visualiser la force des composantes α-connexe du graphe des γ-chemins. Par contre, le calcul de la probabilité d'observer un méta-cluster doit être adapté.

Les changements sont minimes car ils concernent exclusivement m le nombre d'arcs et d_i le degré du nœud i, correspondant au gène i. Le reste est inchangé : les génomes sont concaténés en un seul génome géant et les formules s'appliquent sans modification. Pour chaque gène i, il sera connecté à ses $2\gamma - 1$ gènes voisins. Le degré du nœud i sachant qu'il est n_i fois présent dans les génomes, est donc

$$d_i = n_i \times (2\gamma - 1)$$

Le nombre d'arcs total du graphe des γ-chemins est

$$m = (2\gamma - 1) \sum_i n_i$$

Les autres formules restent inchangées :

$$p_{ij} = \frac{n_{ij}}{m}$$

$P(w_{ij} = k)$ la probabilité d'observer un arc (i, j) de poids k.

$$P(w_{ij} = k) = \left(\begin{array}{c} m \\ k \end{array} \right) p_{ij}^k (1 - p_{ij})^{(m-k)}$$

Pour T_k l'ensemble des arbres couvrants minimaux dans un graphe de k gènes. On obtient pour $P(\mathcal{C})$, la probabilité que les nœuds de l'ensemble $\mathcal{C}$ forment une composante α-connexe et donc un méta-cluster :

$$P(\mathcal{C}) \leq \prod_{n_i \in \mathcal{C}, n_j \notin \mathcal{C}} (1 - P(w_{ij} \geq \alpha)) \times |T_k| \prod_{s \in \{n_2, \ldots, n_k\}} P(w_{n_1,s} \geq \alpha)$$

la probabilité d'observer une composante α-connexe de taille au moins r est quant à elle :

$$P_r \leq \left(\begin{array}{c} |S| \\ r \end{array} \right) \times |T_r| \prod_{\{s^i | i \in [2,r]\}} P(w_{s^1 s^i} \geq \alpha),$$

avec l'ensemble des nœuds $S = \{s^1, \ldots, s^{|S|}\}$ triés par ordre décroissant de degré, ce qui revient à les trier par ordre croissant du nombre d'occurrences de la famille de gènes correspondante.

3.3 Résumé et conclusions

Nous avons adapté le concept des über-operons aux génomes d'eucaryotes. Cette adaptation, bien que non complètement justifiée du point de vue biologique, se base sur le principe communément admis que des gènes dans un même voisinage ont souvent des fonctions apparentées.

Tous les outils développés pour les über-operons s'adaptent pour ces nouveaux objets : la sélection des espèces, les calculs de pertinence et de vraisemblance et la représentation hiérarchique. La différence réside dans l'utilisation du graphe des γ-chemins au lieu de celui des opérons. En effet les opérons sont des structures rares chez les eucaryotes.

La méthode que nous proposons n'a pas pu être validée par des exemples car les données sur les familles de gènes sont encore trop rares pour les eucaryotes. La démarche que nous avons développée ici est une première approche du problème et bien que simpliste doit déjà donner de bons résultats. C'est pourquoi nous avons volontairement choisi le terme de méta-clusters pour représenter les objets détectés.

Deuxième Partie

Scénarios évolutifs contraints

chapitre 4

État de l'art sur les réarrangements

Non seulement les génomes évoluent par leur composition avec les mutations ponctuelles, mais ils évoluent aussi par l'ordre de leurs gènes. Les génomes sont soumis aux réarrangements génomiques. Des événements évolutifs comme les inversions ou les transpositions de segments de gènes modifient l'organisation des génomes de procaryotes. Ces modifications ne permettent pas directement l'apparition de nouvelles fonctions mais modifient en tout cas la régulation des protéines au niveau de la transcription. Indirectement, les duplications permettent aux gènes d'être présents dans les génomes sous la forme de plusieurs copies et par conséquent ces gènes dupliqués peuvent évoluer plus facilement.

Il est d'usage de mesurer l'évolution et de comparer les espèces en analysant les mutations ponctuelles. En se limitant à ce type d'évolution, on ne prend pas en compte toute l'information évolutive disponible. Depuis le début des années 90, l'intérêt pour les réarrangements génomiques et pour leur aptitude a rendre compte de l'évolution est grandissant.

Ce type d'évolution et les problèmes qu'elle soulève lorsque l'on veut comparer des espèces donnent des problèmes combinatoires très intéressants. On s'intéresse par exemple à calculer le nombre minimum d'inversions nécessaire pour transformer un génome en un autre. Ce champ de la bioinformatique est relativement nouveau et n'en est qu'à ses balbutiements.

Nous proposons dans ce chapitre quelques problèmes combinatoires liés aux réarrangements génomiques et à l'évolution des espèces. Dans un premier temps, nous présentons les inversions comme mesure de l'évolution. Nous montrons comment inférer un scénario évolutif parcimonieux avec des inversions, dans le cas des génomes signés et de ceux non signés. Dans une dernière partie, nous exposons les observations récentes qui montrent des lacunes dans les théories précédentes.

4.1 Les réarrangements : les inversions, une mesure de l'évolution

La comparaison de séquences d'ADN ou de séquences peptidiques est à la base de beaucoup de problèmes biologiques. Quels gènes sont similaires ? Quelles séquences partagent des domaines conservés ? Quelle est la conservation de cet ensemble de séquences ? Quel est le scénario évolutif des séquences ? La comparaison de séquences consiste à mettre en avant les parties similaires à deux ou à plusieurs séquences.

La comparaison de séquences est avant tout basée sur leur alignement. Un alignement de séquences est une représentation graphique de celles-ci qui met en avant leurs sites[3] communs. Un exemple d'alignement pour deux séquences d'ADN est le suivant :

séquence 1 :	C	A	C	C	C	T	G	G	A	_	T	G	G
séquence 2 :	C	_	C	C	C	A	G	A	A	C	T	G	G
sites :	1	2	3	4	5	6	7	8	9	10	11	12	13

Pour identifier les sites communs on a introduit des *insertions* avec le caractère "_" sur la deuxième ligne, on parlera de *gap* ; on a aussi introduit des *mismatchs* avec par exemple l'alignement sur la même colonne d'un A avec un T en position 6 ; ou encore la *délétion* avec un gap sur la première ligne en position 10. On peut remarquer que l'insertion et la délétion sont la même opération mais dépendent du sens de lecture de la comparaison : une délétion pour la première séquence correspond à une insertion pour la deuxième et réciproquement. À toutes les autres positions, les nucléotides concordent et on parlera de *matchs*. La comparaison de ces alignements permet d'inférer des relations évolutives entre les séquences d'ADN et par conséquent entre les espèces. En écrivant cet alignement, nous avons supposé quatre évènements évolutifs entre les deux séquences. Tous les événements évolutifs décrits sont des mutations ponctuelles. C'est-à-dire qu'elles modifient un caractère (de la séquence) en un point précis. Ces principes s'adaptent à la comparaison de séquences peptidiques, bien que pour celles-ci on n'utilise que principalement les mismatchs et rarement les insertions et délétions.

Pour comparer des génomes on peut confronter une partie de leur séquence d'ADN et en particulier un de leur gène. On extrait deux gènes orthologues du génome complet des espèces comparés et on les aligne afin de les examiner. La comparaison de séquences d'ADN issus de gènes n'est pas toujours suffisante. D'abord il faut identifier des gènes orthologues. De plus en se limitant à un seul gène on réduit l'information disponible. Les gènes sont aussi sujets aux transferts horizontaux et peuvent ne pas refléter l'évolution des espèces. Pour pallier à ces problèmes, une solution est de comparer les génomes complets. On aligne la séquence nucléique en entier. Dans le cas des procaryotes, celle-ci peut être de plusieurs millions de nucléotides. Un des problèmes majeur lié à cette solution, en plus de la complexité du problème (au sens algorithmique du terme), est que les génomes sont soumis aux réarrangements génomiques. Un morceau de génome peut être déplacé ou copié vers un autre endroit du même chromosome ou d'un autre. Dans ce cas, un alignement des séquences de gauche à droite n'est pas possible.

Pour parer à ces problèmes et difficultés, la comparaison de génomes complets est souvent basée sur une modélisation différentes des génomes : de l'analyse des séquences nucléiques on

[3] Un site correspond aux nucléotides d'une même colonne d'un alignement multiple.

passe à l'étude de l'ordre des gènes. Alors que les réarrangements génomiques empêchent la comparaison des séquences d'ADN car ils modifient l'ordre des gènes, on utilise ces réarrangements pour inférer des relations évolutives. On confronte l'ordre des gènes d'un génome par rapport à un autre et on essaye de déterminer les réarrangements génomiques qui les séparent.

Distance et scénario évolutif

Dans notre étude, nous avons uniquement considéré les génomes avec un seul chromosome. L'information comparée étant l'ordre des gènes, les génomes sont représentés par une liste de gènes. Les gènes sont signés, c'est-à-dire qu'ils sont positifs s'ils sont transcrits à partir du brin d'ADN dit de Watson, et négatif s'ils sont transcrits à partir du brin d'ADN dit de Crick. En appliquant successivement des opérations, une liste de gènes peut être transformée en une autre. Dans notre cas, les opérations autorisées seront l'inversion d'une sous-liste de gènes : l'ordre de ces gènes est inversé et s'ils sont signés, leur signe est aussi inversé. Cette suite d'opérations définit un scénario évolutif entre deux génomes. Un gène est non signé lorsque cette information n'est pas disponible.

Il existe une multitude de scénarios évolutifs entre deux listes de gènes. Pour choisir parmi tous ces scénarios celui qui nous semble le plus probable, on applique le principe de parcimonie. On choisit un scénario le plus court possible en supposant que l'évolution réelle entre deux génomes est probablement la plus économe. On parle de scénarios parcimonieux.

Conjointement aux scénarios évolutifs inférés, on utilise la notion de distance pour mesurer l'évolution de l'organisation de deux génomes. Une distance sur un ensemble E est une fonction $d : E \times E \rightarrow \mathbb{R}$ telle que :

- $d(x, x) = 0$ pour tout $x \in E$;
- $d(x, y) > 0$ pour tout $x, y \in E$ avec $x \neq y$;
- $d(x, y) = d(y, x)$ pour tout $x, y \in E$;
- $d(x, y) \leq d(x, z) + d(z, y)$ pour tout $x, y, z \in E$.

La troisième condition vérifie que d est symétrique et la dernière condition est l'inégalité triangulaire.

Une distance entre deux listes de gènes est la somme minimale des coûts des transformations permettant de transformer l'une en l'autre. Le calcul d'un scénario évolutif ou d'une distance sont deux problèmes similaires bien que leur résultat final soient différents. Il est facile à partir d'un scénario d'obtenir la distance évolutive. On fixe pour chaque opération évolutive un coût ; la distance est alors la somme des coûts associés à chaque opération de ce scénario. À partir d'une distance il est aussi possible d'obtenir un scénario évolutif si les coûts associés aux opérations sont constants et égaux. Il faut appliquer une transformation à un des génomes telle que la nouvelle distance calculée soit moindre que l'ancienne. En itérant ce processus jusqu'à avoir une distance nulle, on obtient un scénario évolutif parcimonieux.

Le formalisme utilisé pour inférer des scénarios évolutifs est basé sur la représentation des substitutions et est le même que pour les Intervalles communs (voir section 1.2.1 page 27). Deux génomes sont représentés par deux permutations sur des entiers. A ceci près qu'on représente le signe des gènes directement sur les permutations. On peut donc numéroter les gènes d'une espèce pour que sa représentation soit la permutation identité avec tous ses gènes positifs. Transformer, par des réarrangements, une permutation en la permutation identité

revient à la trier. On parle donc du tri par inversions de permutations signées à la place du calcul d'un scénario évolutif minimal. Dans certains cas, le signe des gènes n'est pas connu et on manipule des permutations non signées.

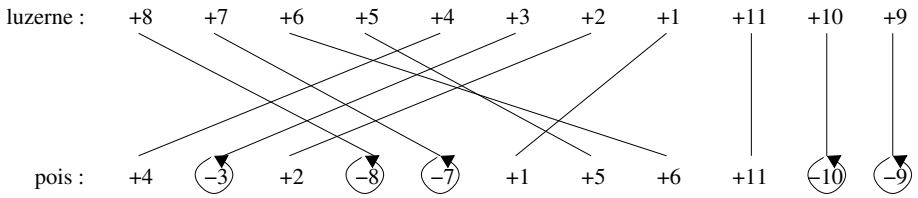
L'étude du tri de permutations a généré beaucoup de littérature. Il a commencé au début des années 1990 avec les articles de Kececioglu et Sankoff [Kececioglu and Sankoff, 1993, Kececioglu and Sankoff, 1995]. Le premier algorithme polynomial en temps pour le calcul de la distance entre deux permutations signées fut proposé par Hannennalli et Pevzner [Hannenhalli and Pevzner, 1995]. Ce premier algorithme a ensuite été amélioré par D. Bader, B. Moret et M. Yan [Bader et al., 2000], lesquels ont proposé un algorithme linéaire en temps par rapport à la taille des permutations. Pour le tri de permutations non signées, le problème ne s'est pas révélé être de la même classe de complexité. En 1997, Alberto Caprara a montré que ce problème était NP-dur [Caprara, 1997]. Et en 1998, P. Berman et M. KArpinski ont montré qu'il était APX-dur. C'est-à-dire qu'il n'existe pas une $(1+\varepsilon)$-approximation polynomiale pour tout $\varepsilon > 0$. Hannenhalli et Pevzner ont proposé un algorithme polynomial en temps sous certaines conditions [Hannenhalli and Pevzner, 1996]. Berman, Hannenhalli et Karpinski ont par la suite proposé une 1.375-approximation [Berman et al., 2002].

Par la suite, une série d'améliorations ou d'études ont été proposées : sur le plan algorithmique avec les travaux de Anne Bergeron [Bergeron, 2001] ; avec l'ajout d'autres opérations évolutives [Walter et al., 1998, Gu et al., 1999, El-Mabrouk, 2000, Eriksen, 2001] ; avec la correction d'erreur [Moret et al., 2001] ; pour la comparaison de plus de deux génomes [Siepel and Moret, 2001, Bader et al., 2001] ; et sur l'étude de tous les chemins parcimonieux [Ajana et al., 2002, Bergeron et al., 2002a].

4.2 Calcul de la distance minimale entre deux génomes signés

Nous commençons l'étude du calcul de la distance entre deux génomes par un exemple tiré de [Setubal and Meidanis, 1997]. Nous n'avons pas sélectionné de génomes de procaryotes pour l'exemple car ceux-ci sont de très grande taille et ne donnent par des exemples pédagogiques. L'exemple est basé sur les génomes de chloroplastes de la luzerne et du pois. Un gène est représenté par un entier. Le brin d'ADN à partir duquel il est transcrit est identifié, soit par le signe plus, soit par le signe moins.

L'organisation des génomes de la luzerne et du pois est la suivante :



Deux gènes sont identifiés par le même nombre s'ils sont orthologues. On supposera par la suite qu'il n'existe pas de gène paralogue. Un scénario évolutif parcimonieux basé uniquement sur les inversions est le suivant :

$$+8 \quad +7 \quad +6 \quad +5 \quad +4 \quad \underline{+3} \quad +2 \quad +1 \quad +11 \quad +10 \quad +9$$

+8 +7 +6 +5 +4 -3 +2 +1 +11 +10 +9

+8 -2 +3 -4 -5 -6 -7 +1 +11 +10 +9

+8 -2 +3 -4 -5 -1 +7 +6 +11 +10 +9

+4 -3 +2 -8 -5 -1 +7 +6 +11 +10 +9

+4 -3 +2 -8 -7 +1 +5 +6 +11 +10 +9

+4 -3 +2 -8 -7 +1 +5 +6 +11 -10 +9

+4 -3 +2 -8 -7 +1 +5 +6 +11 -10 -9

Ce scénario parcimonieux nous donne une distance de sept inversions pour transformer le génome de chloroplaste de la luzerne en celui de chloroplaste du pois.

Une permutation signée π^s est l'énumération des éléments d'un ensemble fini E avec pour chacun de ses éléments soit le signe positif, soit le signe négatif. Pour la représentation des génomes sous forme de permutation le lecteur est prié de se référer à la section 1.2.1 page 27.

On note $\rho(i,j)$ l'inversion des gènes de l'intervalle $[i,j]$ de la permutation signée π^s. Le résultat de cette inversion est l'inversion de l'ordre des éléments compris dans l'intervalle $[i,j]$ et le changement de leur signe. Soit la permutation $\pi^s = +4 \quad +3 \quad -5 \quad -6 \quad -2 \quad +1$ et l'inversion $\rho(2,4)$, alors $\rho(2,4)(\pi^s) = \pi^s.\rho(2,4) = +4 \quad +6 \quad +5 \quad -3 \quad -2 \quad +1$, en utilisant la notation usuelle.

La définition suivante de points de cassure (*breakpoints*) est cruciale pour le calcul de la distance entre deux permutations. Un point de cassure représente deux gènes adjacents dans un seul des deux génomes comparés.

Définition 4.1 *Pour deux permutations signées π^s et σ^s, un point de cassure est défini entre ${\pi^s}_i$ et ${\pi^s}_{i+1}$ si ${\pi^s}_i \quad {\pi^s}_{i+1}$ ou $-{\pi^s}_{i+1} \quad -{\pi^s}_i$ n'apparaissent pas dans σ^s.*

Si σ est la permutation identité, alors ${\pi^s}_i = j$ et ${\pi^s}_{i+1} = k$ forment un point de cassure dans π^s si et seulement si $k - j \neq 1$. Prenons par exemple la permutation signée $\pi^s = +4 \quad +5 \quad +8 \quad +7 \quad +1 \quad -3 \quad -2 \quad +6 \quad +9 \quad +10 \quad -11$, définie par rapport à la permutation identité positive. Il y a un point de cassure entre les éléments +5 et +8, entre +8 et +7, entre +7 et +1, entre +1 et -3, entre -2 et +6, entre +6 et +9 et entre +10 et -11.

Nous avons déjà vu que les génomes de procaryotes sont soit linéaires, soit circulaires. Dans le cas de génomes linéaires, on ajoute à la permutation de taille n l'élément $+0$ à gauche et l'élément $+(n+1)$ à droite. La permutation précédente devient : $\pi^s = +0 \quad +4 \quad +5 \quad +8 \quad +7 \quad +1 \quad -3 \quad -2 \quad +6 \quad +9 \quad +10 \quad -11 \quad +12$. Pour les génomes circulaires, on ne rajoute pas d'élément mais on considère que l'élément le plus à gauche et celui le plus à droite sont adjacents. Cela modifie les points de cassure. La même permutation que précédemment mais interprété comme une permutation circulaire a un nouveau point de cassure entre -11 et +4.

Pour π une permutation signée ou non, on note sa distance par inversions envers une autre permutation σ : $d(\pi,\sigma)$. Si σ est la permutation identité, on note simplement $d(\pi)$ la distance par inversions. Quand σ est la permutation identité, transformé π en σ revient à trier π. On parle alors du tri par inversions.

La théorie d'Hannenhalli et Pevzner

Nous montrons maintenant, sans aucune preuve, comment on peut trier parcimonieusement une permutation signée. Les résultats présentés ici ont été proposés par [Hannenhalli and Pevzner, 1995] et reformulés dans [Setubal and Meidanis, 1997].

Pour π^s une permutation signée de taille n définie sur E et par rapport à la permutation identité, nous définissons le graphe de la réalité et du désir (ou graphe des points de cassure) $RD = (S, A)$ de π^s. Pour chaque élément ${\pi^s}_i$, $1 \leq i \leq n$, de π^s on définit les sommets ${\pi^s}_i g$ et ${\pi^s}_i d$ représentant la gauche et la droite de l'élément ${\pi^s}_i$. Si π^s est linéaire on rajoute les éléments $0d$ et $(n+1)g$. S l'ensemble des sommets de RD est de taille $2n+2$ si π^s est linéaire ou de taille $2n$ si π^s est circulaire. Il y a deux types d'arcs dans le graphe RD, ceux de la réalité (représentés par des traits plein) et ceux du désir (représentés en pointillé). Pour tout $+{\pi^s}_i$ et $+{\pi^s}_{i+1}$, un couple d'éléments adjacents de π^s, on relie les arcs ${\pi^s}_i d$ et ${\pi^s}_{i+1} g$ par un arc de la réalité. Pour tout $-{\pi^s}_i$ et $+{\pi^s}_{i+1}$, un couple d'éléments adjacents de π^s, on relie les arcs ${\pi^s}_i g$ et ${\pi^s}_{i+1} g$ par un arc de la réalité. Pour tout $+{\pi^s}_i$ et $-{\pi^s}_{i+1}$, un couple d'éléments adjacents de π^s, on relie les arcs ${\pi^s}_i d$ et ${\pi^s}_{i+1} d$ par un arc de la réalité. Et pour tout $-{\pi^s}_i$ et $-{\pi^s}_{i+1}$, un couple d'éléments adjacents de π^s, on relie les arcs ${\pi^s}_i g$ et ${\pi^s}_{i+1} d$ par un arc de la réalité. On fait de même pour la permutation identité et on obtient les arcs du désir. Le graphe de la réalité et du désir de la permutation $\pi^s = +0 \ \ +4 \ \ +5 \ \ +8 \ \ +7 \ \ +1 \ \ -3 \ \ -2 \ \ +6 \ \ +9 \ \ +10 \ \ -11 \ \ +12$ est donné figure 4.1.

Trier une permutation revient à appliquer des opérations à la permutation et donc à son graphe de la réalité et du désir. Au lieu d'appliquer les inversions sur les permutations, on peut directement les appliquer au graphe de la réalité et du désir. Trier la permutation revient alors à faire correspondre désir et réalité. On observe sur le graphe de la réalité et du désir figure 4.1 que, pour les éléments 4 droit et 5 gauche, le désir correspond déjà à la réalité.

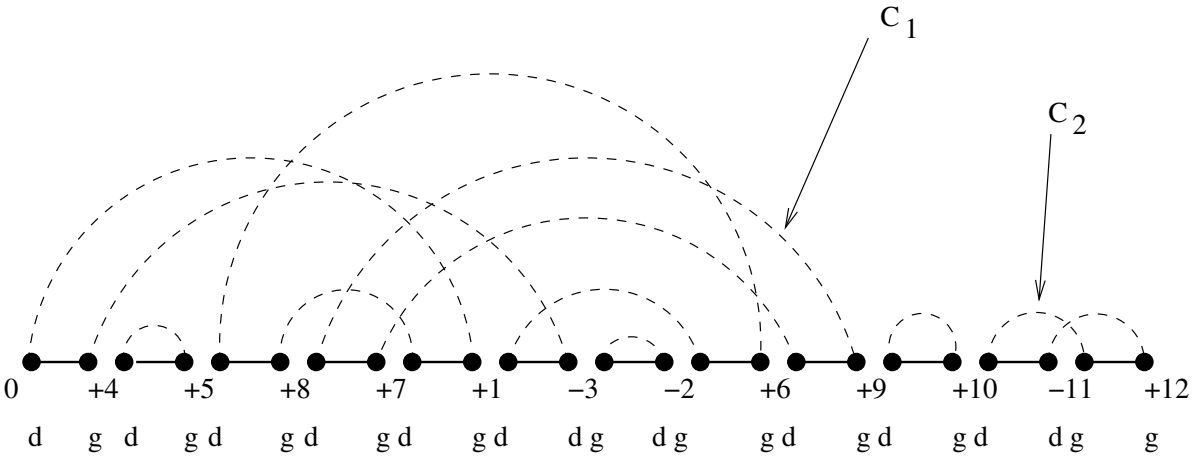


Figure 4.1: Le graphe de la réalité et du désir pour la permutation signée et linéaire $\pi^s = +0 \ \ +4 \ \ +5 \ \ +8 \ \ +7 \ \ +1 \ \ -3 \ \ -2 \ \ +6 \ \ +9 \ \ +10 \ \ -11 \ \ +12$. Les pointillés représentent les arcs du désir et les traits plein représentent les arcs de la réalité.

Nous allons maintenant donner une approximation de la distance par inversions et ensuite, par plusieurs raffinements successifs, nous arriverons au calcul exact de cette distance. Le

graphe RD de la permutation identité contre elle-même est un graphe avec le maximum de cycles. On peut voir l'action de trier une permutation comme l'action d'augmenter le nombre de cycles du graphe RD correspondant. On notera $c(\pi^s)$ le nombre de cycles du graphe de la réalité et du désir de la permutation signée π^s.

Pour un cycle du graphe, le parcourir donne un sens de lecture aux arcs de la réalité. Si on représente comme dans la figure 4.1 les arcs de la réalité sur un axe horizontal, les arcs sont soit parcourus de la gauche vers la droite, soit de la droite vers la gauche. Pour un cycle, si tous ses arcs de la réalité peuvent être parcourus dans le même sens, alors on dira qu'il est convergeant ; sinon on dira qu'il est divergeant. Dans le graphe de la figure 4.1 le cycle C_1 est convergeant alors que le cycle C_2 est divergeant. Dans un même cycle, pour deux arcs qui peuvent être parcourus dans la même direction on dit qu'ils convergent, sinon on dit qu'ils divergent.

Théorème 4.1 *[Setubal and Meidanis, 1997] Pour π^s une permutation signée, RD son graphe de la réalité et du désir et ρ une inversion délimitée par les arcs de la réalité u et v, alors*

- *si u et v appartiennent à deux cycles, $c(\pi^s.\rho) = c(\pi^s) - 1$;*
- *si u et v appartiennent au même cycle et convergent, $c(\pi^s.\rho) = c(\pi^s)$;*
- *si u et v appartiennent au même cycle et divergent, $c(\pi^s.\rho) = c(\pi^s) + 1$.*

La permutation π une fois triée correspond à la permutation σ et c'est dans cette configuration que l'on obtient le maximum de cycles. Donc pour tout scénario parcimonieux $\rho_1.....\rho_d$ triant une permutation signé π^s :

$$c(\pi^s.\rho_1.....\rho_d) = c(\sigma) = n + 1$$

Comme d'après le théorème 4.1, chaque inversion modifie le nombre de cycle de plus ou moins un, nous savons que pour chaque inversion nous avons gagné au plus un cycle :

$$\begin{aligned} 1 &\geq c(\pi^s.\rho 1) - c(\pi^s) \\ 1 &\geq c(\pi^s.\rho 1.\rho 2) - c(\pi^s.\rho 1) \\ &\vdots \\ 1 &\geq c(\pi^s.\rho 1.....\rho_d) - c(\pi^s.\rho 1.....\rho_{d-1}) \end{aligned}$$

En effectuant la somme de tous les termes, on obtient :

$$c(\pi^s.\rho 1) - c(\pi^s) + ... + c(\pi^s.\rho 1.....\rho_d) - c(\pi^s.\rho 1.....\rho_{d-1}) \leq d$$

Et finalement,

$$d(\pi^s) \geq n + 1 - c(\pi^s)$$

On peut utiliser la formule $n+1-c(\pi^s)$ pour estimer la distance par inversions mais nous allons voir que dans certains cas cette estimation est fausse.

Lorsque les cycles ont tous au moins deux arcs qui divergent, on peut leur appliquer une certaine suite d'inversions telle que l'on gagne un cycle à chaque étape (voir théorème 4.2). Or à chaque étape on peut gagner au plus un cycle. Dans ce cas, l'estimation précédente de la distance devient une formule exacte. Le problème vient des cycles qui convergent, car leur appliquer une inversion ne garantie pas que le nombre de cycles augmente. En réalité, s'ils sont entrelacés avec un autre cycle convergeant, appliquer une inversion sur deux arcs qui divergent du cycle divergeant, permet de rendre le premier cycle divergeant à son tour. Donc tout en augmentant le nombre de cycles, on a transformé un cycle convergeant en cycle divergeant. On peut donc dans ce cas quand même trier la permutation en augmentant le nombre de cycles du graphe de la réalité et du désir de 1 cycle pour chaque inversion. Et on obtient encore une fois que l'estimation $n+1-c(\pi^s)$ est la distance exacte.

La formule précédente n'est pas exacte quand on traite un ensemble de cycles entrelacés et tous convergeants. Dans ce cas là, d'après le théorème 4.1, leur appliquer une inversion soit ne modifie pas le nombre de cycles, soit le diminue de un. Un ensemble de cycles entrelacés et convergeants est appelé une mauvaise composante en opposition aux bonnes composantes.

Théorème 4.2 *[Setubal and Meidanis, 1997] Une inversion définie par deux arcs divergeant d'un même cycle est une inversion qui tri parcimonieusement si et seulement si son application ne crée pas de mauvaises composantes.*

On peut transformer une mauvaise composante en bonne composante en lui appliquant une inversion. On transforme alors un cycle convergeant en cycle divergeant sans modifier le nombre de cycles. Cette opération est montrée figure 4.2. On peut donc utiliser le nombre de mauvaises composantes pour améliorer notre estimation de la distance par inversions. A priori, chaque mauvaise composante nécessite une inversion supplémentaire pour être transformée en bonne composante. Mais toutes les mauvaises composantes n'ont pas besoin d'une inversion supplémentaire.

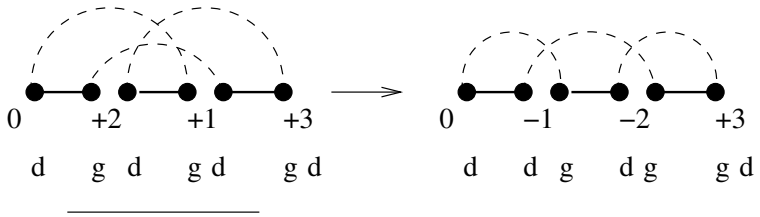


Figure 4.2: Un exemple d'inversion transformant une mauvaise composante en bonne composante. Initialement la mauvaise composante est constituée d'un seul cycle convergeant. En lui appliquant une inversion, on n'a pas augmenté le nombre de cycles mais on a transformé un cycle convergeant en cycle divergeant. Ce qui permet d'obtenir une bonne composante.

On dit qu'une composante A sépare une composante B et C si en traçant des liens des nœuds de B vers ceux de C, ceux-ci croisent tous des arcs de la composante A. Une des deux composantes B ou C est "sous" la composante A.

Appliquer une inversion sur un arc de B et un arc de C transforme ces deux mauvaises composantes en une bonne composante tout en transformant la mauvaise composante A en bonne composante. Une *haie* (*hurdle*) est une mauvaise composante qui n'en sépare aucune autre, sinon c'est une non haie. Nous avons besoin d'une inversion supplémentaire pour transformer un des cycles qui converge d'une haie en cycle divergeant. Les non haies sont transformées en bonnes composantes en triant les haies avoisinantes. Cette opération est expliquée figure 4.3.

Nous notons $h(\pi^s)$ le nombre de haies du graphe de la réalité et du désir. Nous obtenons une nouvelle approximation de la distance par inversions :

$$d(\pi^s) \geq n + 1 - c(\pi^s) + h(\pi^s)$$

Nous avons introduit les non haies car celles-ci sont des haies qui peuvent être supprimées en éliminant les autres haies. Et elles ne nécessitent par conséquent pas d'inversion supplémentaire. Il existe un cas particulier où on a besoin d'une inversion supplémentaire pour trier parcimonieusement une permutation. On supprime chaque non haie en appliquant une inversion sur deux mauvaises composantes qu'elle sépare. Mais si le nombre de non haies et le nombre de mauvaises composantes séparées ne concordent pas, on peut devoir effectuer une inversion supplémentaire pour supprimer une non haie. Une mauvaise composante qui est séparée par une non haie, laquelle ne sépare que cette mauvaise composante, est appelée une super haie. Quand le nombre de super haies est impair, on ne peut pas supprimer toutes les non haies en supprimant les haies simples. On a alors besoin d'une inversion supplémentaire pour éliminer cette non haie. Une telle structure est appelée une forteresse et on note $f(\pi^s)$ une fonction qui retourne un ou zéro suivant qu'il existe ou non une forteresse dans π^s.

Nous avons fini de caractériser le graphe de la réalité et du désir et nous pouvons calculer de manière exacte la distance par inversions. La distance par inversions est exprimée par la formule suivante :

$$d(\pi^s) = n + 1 - c(\pi^s) + h(\pi^s) + f(\pi^s)$$

La première version proposée par Hannenhalli et Pevzner identifiait les différentes structures nécessaires au calcul de $d(\pi^s)$ en $O(n^4)$. Par la suite, Berman et Hannenhalli ont proposés un algorithme identifiant ces structures en $O(n^2 \times \alpha(n))$ [Berman and Hannenhalli, 1996]. Puis Kaiplan, Shamir et Terjan ont proposé un algorithme en $O(n \times \alpha(n) + d \times n)$ [Kaplan et al., 2000]. Finalement, Bader, Moret et Yan ont proposé un algorithme linéaire pour le calcul de ces structures [Bader et al., 2000]. On peut donc calculer $d(\pi^s)$ en temps linéaire.

4.3 Calcul de la distance minimale entre deux génomes non signés

On ne dispose pas toujours d'information sur le brin d'ADN à partir duquel sont transcrits les gènes. Les éléments des permutations n'ont alors pas de signe et le problème devient de trier par des inversions des permutations non signées ; ou de calculer la distance minimale

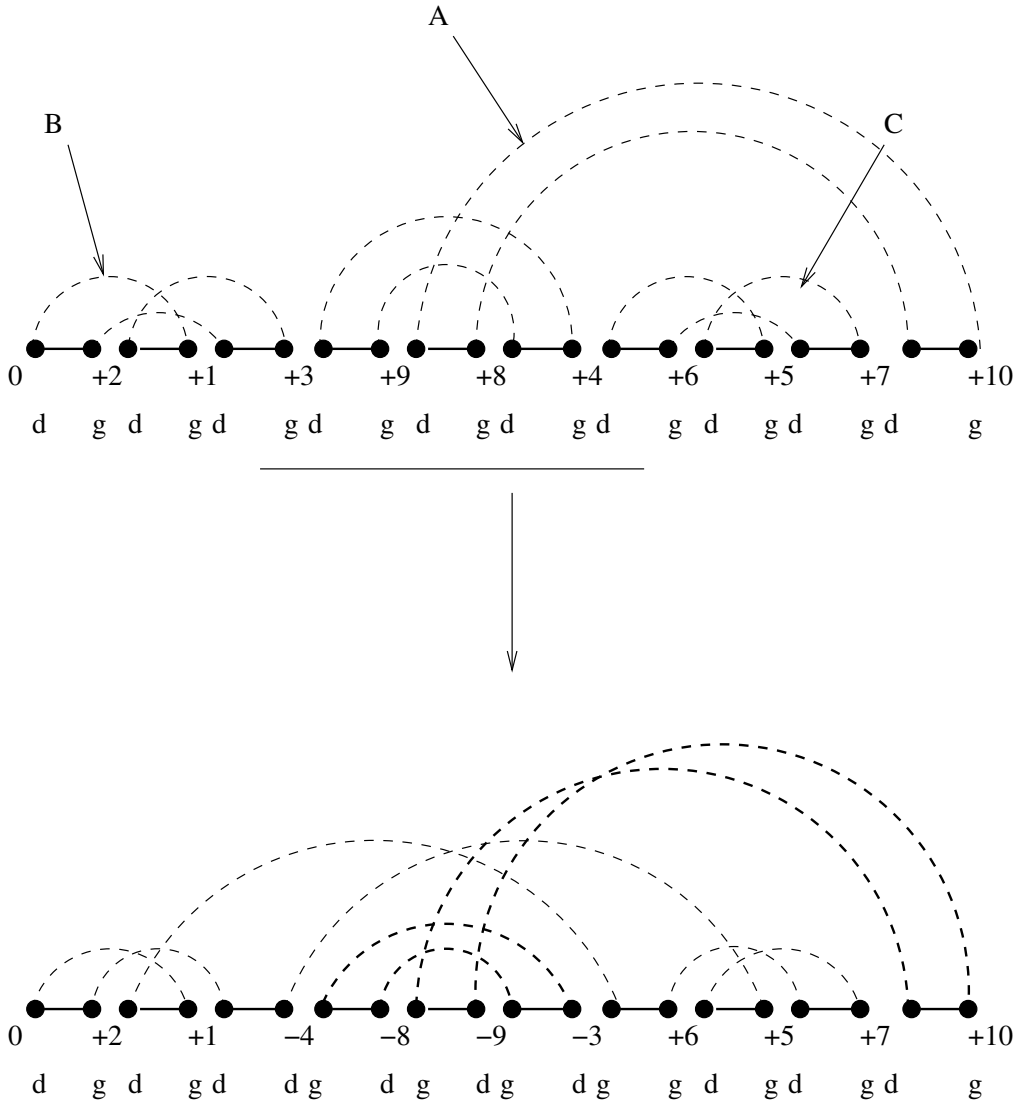


Figure 4.3: Un exemple d'inversion transformant trois mauvaises composantes (deux haies et une mauvaise composante qui les sépare) en deux bonnes composantes. La composante A sépare les composante B et C. Toutes les composantes sont mauvaises, on ne peut donc pas appliquer une inversion pour augmenter le nombre de cycles. Par contre, on peut appliquer une inversion qui relie les deux haies simples pour qu'elles ne forment plus qu'une bonne composante. On a perdu un cycle, mais on a transformé deux haies en une seule bonne composante. La mauvaise composante les séparant a été retournée et devient après cette inversion une bonne composante. Nous n'avons pas eu besoin d'inversion supplémentaire pour transformer A en bonne composante.

en nombre d'inversions entre deux permutations non signées. Nous présentons dans cette section les travaux d'Hannenhalli et Pevzner sur ce problème [Hannenhalli and Pevzner, 1996]. Nous rappelons que le tri de permutations non signées par inversions est un problème NP-dur [Caprara, 1997]. L'algorithme exact que nous présentons est bien sûr de complexité exponentielle en temps mais polynomiale sous certaines conditions.

Définition 4.2 *[Hannenhalli and Pevzner, 1996] Pour deux permutations non signées π^u et σ^u, un point de cassure est défini entre ${\pi^u}_i$ et ${\pi^u}_{i+1}$ si ${\pi^u}_i$ ${\pi^u}_{i+1}$ ou ${\pi^u}_{i+1}$ ${\pi^u}_i$ n'apparaissent pas dans σ .*

Si σ est la permutation identité, alors ${\pi^u}_i = j$ et ${\pi^u}_{i+1} = k$ forment un point de cassure dans π^u si et seulement si $||k - j|| \neq 1$.

Définition 4.3 *[Hannenhalli and Pevzner, 1996] Un bloc d'une permutation non signée π^u est un intervalle $[x, y]$ de π^u qui ne contient aucun point de cassure.*

Définition 4.4 *[Hannenhalli and Pevzner, 1996] Un strip d'une permutation non signée π^u est un bloc maximal, c'est-à-dire, un bloc $[x, y]$ tel qu'il existe un point de cassure entre ${\pi^u}_{x-1}$ et ${\pi^u}_x$ et entre ${\pi^u}_y$ et ${\pi^u}_{y+1}$.*

Une permutation signée π^s est un *spin* d'une permutation non signée π^u, telle que ${\pi^s}_i = {+\pi^u}_i$ ou ${\pi^s}_i = {-\pi^u}_i$. Pour une permutation non signée arbitraire π^u de taille n nous définissons Π^u l'ensemble des 2^n spins de π^u.

L'algorithme proposé par Hannenhalli et Pevzner consiste à calculer la distance par inversions pour tous les spins d'une permutation non signée et à choisir la plus petite distance. Cette distance est la distance minimale pour trier des permutations non signées avec des inversions. De plus, l'algorithme ne teste pas tous les 2^n spins car pour certaines structures, les strips, il est possible de déterminer *a priori* le signe des gènes d'un spin donnant la distance minimale. Un spin π^s de π^u est dit optimal, s'il donne la distance minimale, *i.e.* si $d(\pi^u) = d(\pi^s)$. L'algorithme est principalement basé sur le lemme suivant :

Lemme 4.1 *[Hannenhalli and Pevzner, 1996] Pour une permutation non signée π^u,*

$$d(\pi^u) = \min_{\pi^s \in \Pi^u} d(\pi^s)$$

Nous commençons par définir la façon dont seront triées les structures permettant de ne pas tester tous les spins. Un bloc ou un strip $[x, y]$ d'une permutation signée ou non est dit *croissant* si $||\pi_x|| < ||\pi_y||$, sinon il est dit *décroissant*. Un strip croissant (respectivement décroissant) d'une permutation signée est dit *canonique* si tous ses éléments sont positifs (respectivement négatifs). Un strip canonique n'a aucun point de cassure et correspond donc à un intervalle trié, soit positivement, soit négativement. Un strip décroissant avec tous ses éléments signés positifs est appelé un strip *anti canonique*.

Les premières structures permettant de ne pas tester tous les spins sont les strips de longueur au moins trois :

Lemme 4.2 *[Hannenhalli and Pevzner, 1996] Pour une permutation non signée π^u, il existe π^s un spin optimal tel que tous ses strips de taille au moins trois sont canoniquement signés.*

Maintenant nous nous intéressons aux strips de taille deux.

Lemme 4.3 *[Hannenhalli and Pevzner, 1996] Pour une permutation non signée π^u, il existe π^s un spin optimal de π^u tel que tous les strips de longueur deux sont soient signés canoniquement, soient signés anti canoniquement.*

Lemme 4.4 *[Hannenhalli and Pevzner, 1996] Pour une permutation non signée π^u, il existe π^s un spin optimal de π^u tel que, une bonne composante ne contient aucun strip de longueur deux anti canonique, et une mauvaise composante en contient au plus un.*

Un *super spin* de π^u est un spin de π^u dont tous les strips sont canoniques ; on a alors sélectionné dans chaque mauvaise composante un strip de longueur deux, s'il en existe, et on l'a transformé en anti strip.

Théorème 4.3 *[Hannenhalli and Pevzner, 1996] Tout super spin d'une permutation non signée composé uniquement de strips est un spin optimal.*

Pour calculer la distance par inversions entre deux permutations non signées de taille n, Hannenhalli et Pevzner ont proposé un algorithme en $O(n2^k+n)$. k étant le nombre d'éléments des permutations n'étant pas dans des strips. Il faut tester les 2^k spins différents pour ces k éléments puis en appliquant les règles de construction d'un super spin, qui peuvent être effectuées en temps linéaire, on obtient un spin optimal. Si le nombre d'éléments n'étant pas dans des spins est en $O(\log n)$, alors l'algorithme proposé est de complexité polynomiale en temps.

4.4 Les limites du modèle

La modélisation des génomes sous la forme de permutations signées a permis de proposer des algorithmes linéaires en temps pour résoudre le problème de la distance. Dans le cas des permutations non signées, le problème est NP-dur. Mais la représentation sous forme de permutations apporte ses propres inconvénients. D'abord le modèle des permutations est trop rigoureux car elles sont définies sur le même ensemble d'éléments. De plus on limite les scénarios inférés à un petit nombre d'opérations évolutives. Et d'autre part, le nombre de scénarios parcimonieux est très important ce qui signifie peut être que le modèle des permutations n'apporte pas assez de contraintes.

Les permutations, un modèle rigoureux

L'utilisation des permutations pour modéliser les génomes et ainsi permettre le calcul de la distance par inversions limite les données biologiques prises en compte. Les permutations obligent les génomes comparés à être définis sur le même ensemble de gènes. Cette hypothèse n'est pas réaliste car les génomes subissent des délétions et des insertions de gènes. Nadia El Mabrouk a travaillé sur la construction d'un scénario évolutif en ajoutant l'insertion et la délétion de segments de gènes [El-Mabrouk, 2000]. Les algorithmes proposés sont des heuristiques et ne donnent pas de solutions exactes.

De même, les permutations n'autorisent pas les gènes dupliqués et nous savons que pour certaines espèces, le nombre de gènes dupliqués avoisine 30% de leurs gènes.

Quelles opérations choisir ?

Nous avons présenté des algorithmes pour le calcul de la distance évolutive qui ne prennent en compte que les inversions des segments de gènes. Même si dans certains cas, les inversions sont l'opération évolutive la plus importante, elle n'est pas la seule. Il existe d'autres opérations comme les transpositions, les transreversal (transposition suivie d'une inversion du segment transposé), etc. Il existe bien souvent des algorithmes efficace lorsque l'on traite un seul type d'opération évolutive, mais dès que l'on veut ajouter d'autres opérations, les problèmes combinatoire deviennent plus dures. Nous synthétisons dans le tableau ci-dessous l'évolution actuelle des travaux dans ce domaine.

Réarrangements pris en comptes	note	résultat	auteurs
inversion signée	distance	$O(n)$	[Bader et al., 2000]
inversion signée	scénario	$O(n^{3/2} * \sqrt{logn})$	[Kaplan and Verbin, 2003]
inversion non signée	distance	NP-dur	[Caprara, 1997]
inversion non signée	distance	1.375-approximation	[Berman et al., 2002]
inversion signée	médiane	NP-dur	[Caprara, 1999]
inversion signée et transposition		$(1+\varepsilon)$-approximation	[Eriksen, 2001]
inversion signée, délétion et insertion	heuristique		[El-Mabrouk, 2000]
inversion signée, délétion et duplication	heuristique		[Marron et al., 2003]
transposition		1.5-approximation	[Bafna and Pevzner, 1995]
translocation non signée		1.5-approximation	[Hartman and Shamir, 2003]
translocation signée			[Hannenhalli, 1995]
inversion et translocation		$O(n^2)$	[Ozery-Flato and Shamir, 2003]
transposition et transreversal		1.5-approximation	[Hartman and Sharan, 2004]
fusion, fission et transposition	poids pour les opérations	Aussi difficile que la distance par transposition	[Dias and Meidanis, 2002]
fusion, fission et transposition	poids pour les opérations (trans=w)	$\frac{2}{w}$-approximation	[Dias and Meidanis, 2002]
fusion, fission et transposition		$O(n^2)$	[Dias and Meidanis, 2001]
fusion et fission	distance synténique	$O(n^2 + nc\alpha(nc, n))$	[Dias and Meidanis, 2002]
fusion, fission et translocation	distance synténique	NP-Dur	[DasGupta et al., 1998]
fusion, fission et translocation	distance synténique	2-approximation	[Liben-Novel, 2001]

Nombre de scénarios évolutifs minimaux

La première analyse du nombre de scénarios parcimonieux vient de la recherche liée à l'optimisation d'un algorithme pour résoudre le problème de la médiane (on cherche une permutation "ancestrale" la plus proche de trois autres permutations) [Siepel and Moret, 2001]. Typiquement, ces algorithmes vont tester tous les scénarios évolutifs parcimonieux entre ces trois espèces et en choisir un commun aux trois espèces. Pour optimiser ce type d'algorithme Adam Siepel a développé un algorithme qui permet d'obtenir toutes les inversions triantes (c'est-à-dire qui diminuent la distance par inversions) [Siepel, 2002]. Cette étude a permis de noter qu'il existe un nombre conséquent de scénarios parcimonieux différents. Une étude plus approfondie a été menée par Anne Bergeron *et al.* dans [Bergeron et al., 2002a]. Le théorème principal statuant sur le nombre de scénarios évolutifs parcimonieux est le suivant :

Théorème 4.4 *[Bergeron et al., 2002a] En considérant qu'il n'existe pas de haie*[4] *dans les permutations, pour π^s une permutation signée de taille n avec n points de cassure, et ρ une inversion quelconque sur π^s, la probabilité que ρ ne trie pas π^s est de $O(1/n^2)$.*

Autrement dit, quasiment toutes les inversions appliquées sur des permutations sans point de cassure sont triantes.

Par exemple, le nombre de scénarios parcimonieux triant la permutation linéaire -4 +1 -3 +6 -7 -5 +2 de taille sept est de 200. Un autre exemple encore plus impressionnant est pour la permutation signée et linéaire de taille 31

+13 -19 -29 -23 +16 -31 -10 +7 -27 +12 -30 -14 +15 -20 -6 +3 +6 -2

-22 -28 -21 -5 -1 +11 +4 +8 +25 +9 +18 -17 -24

Pour cette permutation le nombre de scénarios évolutifs parcimonieux est de plus de 15×10^{12}. Bien sûr il existe des permutations pour lesquelles il n'existe qu'un seul scénario parcimonieux. Mais celles-ci ne sont bien souvent pas très mélangées.

Pour conclure, on peut avoir des doutes sur la véracité biologique des scénarios proposés. Si tous les scénarios sont parcimonieux pourquoi celui proposé est il meilleur qu'un autre ? De plus, on voit la nécessité d'apporter des contraintes supplémentaires au tri des permutations. Cela a été fait dans quelques rares cas, dont celui évoqué plus haut [Bergeron et al., 2002a] où les auteurs évoquent l'utilisation de différents critères. Une autre approche est exposée dans [Ajana et al., 2002, Lefebvre et al., 2003] ou on suppose que les inversions de petites tailles sont plus probables que les autres.

Autres travaux relatifs aux réarrangements

La présentation des réarrangements proposée dans ce chapitre est succincte. Nous avons volontairement omis d'autres travaux, fort intéressants, pour ne pas perdre le lecteur.

Il faut quand même citer le travail réalisé par Anne Bergeron dans [Bergeron, 2003]. Elle propose d'estimer les distances évolutives en comptant le nombre d'Intervalles conservés. Un

[4] Les haies sont des structures particulières et relativement rares.

Intervalle conservé est un Intervalle commun restreint au cas ou le plus petit et le plus grand élément délimitent ces bornes. Soit a le plus petit élément (en valeur absolue) et b le plus grand élément, alors les Intervalles communs du type $[a \ldots b]$ ou $[-b \ldots -a]$ sont des Intervalles conservés. Cette étude montre que le nombre d'Intervalles conservés est un meilleur estimateur d'évolution pour des génomes de chloroplastes. Dans une deuxième étude, [Bergeron et al., 2004], Bergeron et ses collègues proposent de reconstruire les génomes ancestraux en se basant sur les Intervalles conservés.

Des travaux plus directement liés aux calculs des distances parcimonieuses ont aussi été omis. On peut citer par exemple EDE [Moret et al., 2001], qui permet de corriger une distance d'édition pour obtenir le nombre correct d'inversions ; ou encore [Cosner et al., 2000], ou les données sur l'ordre des gènes sont encodées en séquences pour pouvoir utiliser les algorithmes classiques de reconstruction phylogénétique basés sur le maximum de parcimonie ; nous n'avons pas non plus parler du problème de la médiane [Siepel and Moret, 2001, Sankoff and Blanchette, 1998], où l'on infère une espèce ancestrale à trois autres espèces en utilisant les réarrangements ; Il y a aussi le travail de David Sankoff sur les distances parcimonieuses où leur calcul fait intervenir les familles de gènes [Sankoff, 1999] ; finalement nous devons aussi citer le travail fait par l'équipe du professeur Moret [Bader et al., 2001]. Ils proposent une suite logicielle permettant de calculer les principales variantes des distances basées sur les réarrangements.

4.5 Résumé et conclusions

Nous avons présenté les réarrangements génomiques et plus particulièrement les inversions de segments de gènes. Le problème d'inférer un scénario parcimonieux entre deux génomes ou le calcul d'une distance entre ces deux génomes posent des problèmes combinatoires intéressants. Les génomes sont alors représentés par des permutations signées ou non.

Nous avons exposé le calcul de la distance par inversions entre deux permutations signées. Nous avons montré le travail fait par [Hannenhalli and Pevzner, 1995] qui a donné lieu au premier algorithme polynomial pour ce problème. Dans un deuxième temps, nous avons montré comment trier des permutations non signées. L'algorithme proposé est un algorithme exact et de complexité exponentielle. Sous certaines conditions, si le nombre de singletons est en $O(\log n)$, l'algorithme proposé est de complexité polynomiale en temps. Finalement, nous avons clôturé ce chapitre par les limites des modèles présentés. Non seulement les algorithmes ne prennent pas en compte les duplications, mais ils obligent les génomes comparés à avoir le même contenu en gènes. De plus les algorithmes n'utilisent pas tous les réarrangements et bien souvent ne se limitent qu'aux inversions. Nous avons terminé en évoquant le problème du grand nombre de scénarios parcimonieux mettant en doute la véracité de ceux proposés par les algorithmes.

Les distances par inversions calculées par les algorithmes classiques, bien qu'efficaces du point de vue algorithmique, ne donnent pas toujours des résultats biologiques. Il est donc nécessaire de modifier les distances calculées en redéfinissant les problèmes.

chapitre 5

Scénario évolutif et contraintes simples

Pour un ensemble fixé d'opérations évolutives et deux génomes donnés, il existe une multitude de scénarios évolutifs possibles. Le principe couramment utilisé pour choisir certains scénarios plutôt que d'autres est le principe de parcimonie, aussi connu comme le principe du rasoir d'Ockham : les explications les plus courtes sont les meilleures. Inférer un scénario parcimonieux consiste donc à chercher le scénario avec le moins de pas évolutifs. Dans le cas des réarrangements, on cherchera le scénario évolutif avec le moins d'opérations de réarrangements.

Mais un scénario le plus parcimonieux n'est pas toujours le vrai scénario, ne serait-ce que parce qu'il en existe souvent une multitude. Que peut-on dire sur la véracité d'un scénario parcimonieux parmi un millier d'autres tout aussi parcimonieux ? Un scénario parcimonieux n'est pas non plus toujours le plus vraisemblable. Pour étayer cette proposition nous considérons l'exemple des scénarios évolutifs par inversions dans le cas particulier où l'on ne connaît pas le signe des gènes. On modélise les génomes sous la forme de permutations et inférer un scénario évolutif entre deux génomes revient à trier une permutation non signée. Il a été montré dans [Hannenhalli and Pevzner, 1996] que, parfois, les scénarios évolutifs les plus parcimonieux séparent des gènes qui sont adjacents dans les deux espèces. Deux gènes non côte à côte dans une espèce ancestrale ont-ils été réunis indépendamment par l'évolution dans deux de ses espèces filles ? Ce cas particulier expose ce que l'on peut appeler un cas d'homoplasie dans les scénarios évolutifs parcimonieux. Les scénarios évolutifs trouvés sont peut être les plus courts mais ne nous semblent pas toujours vraisemblables.

Il est communément admis que les réarrangements principaux chez les procaryotes sont les inversions de segments de gènes. Et par conséquent, nous considérons exclusivement les scénarios évolutifs par inversions. Nous nous attachons plus particulièrement dans ce chapitre à introduire plus de vraisemblance dans les scénarios parcimonieux par inversions. Nous nous sommes intéressés à une structure conservée et facilement identifiable dans les permutations représentant les génomes : les Intervalles communs. Ceux-ci étant des structures conservées, nous supposons que l'évolution les a maintenus. Nous considérons donc dans ce chapitre les scénarios parcimonieux respectant les Intervalles communs. Ce sont les scénarios les plus courts possibles et maintenant les Intervalles communs tout au long de leur histoire.

Dans un premier temps, nous introduisons le problème du tri des permutations signées respectant les Intervalles communs, que nous nommons MCSS. Nous montrons que les In-

tervalles communs permettent de détecter des homoplasies dans les scénarios évolutifs. Par conséquent, trier une permutation en respectant les Intervalles communs permet de supprimer ces homoplasies des scénarios évolutifs. Puis nous proposons un algorithme de complexité exponentielle en temps mais polynomial sous certaines conditions. Nous démontrons également que le problème posé est NP-complet. Dans un quatrième et dernier temps, nous calculons le diamètre du problème.

5.1 Définition des contraintes

Une permutation est toujours constituée d'au moins un Intervalle commun, lesquels forment des sous permutations de cette dernière. Un intervalle commun est composé de deux intervalles, un pour chaque permutation, contenant le même ensemble de gènes. Pour la définition formelle des Intervalles communs, nous renvoyons le lecteur à la section 1.2 page 26.

Les Intervalles communs sont des structures partagées par les génomes, et s'ils sont présents dans les espèces actuelles, on peut supposer qu'ils ont été conservés pendant l'évolution de leur génome. Les inversions modifient l'ordre des gènes et par conséquent cassent des intervalles communs. Les inversions peuvent bien sûr créer des Intervalles communs mais ce type d'évolution forme, ce que l'on peut appeler, une homoplasie. Une homoplasie, rappelons-le, se dit de sites présentant des états identiques mais ayant subi différentes étapes évolutives (réversions, mutations parallèles ou convergentes). Nous nous sommes permis d'adapter ce terme, lié aux mutations ponctuelles, aux réarrangements génomiques. Les homoplasies sont à éviter lorsque l'on reconstruit un scénario évolutif car supprimer un caractère pour le reformer par la suite n'est sans doute pas le scénario le plus probable.

Nous nous sommes naturellement intéressés à calculer des scénarios évolutifs en respectant les Intervalles communs existants dans les génomes actuels. Notre postulat est que ces Intervalles communs sont conservés pendant l'évolution et les scénarios évolutifs inférés doivent donc eux aussi les conserver.

Nous introduisons quelques termes pour caractériser les inversions en fonction des Intervalles communs :

- Une inversion $\rho(i,j)$ est *interne* à un intervalle $[k,l]$ si $k \leq i \leq j \leq l$.
- Une inversion $\rho(i,j)$ *incorpore* un intervalle $[k,l]$ si $i < k < l \leq j$ ou $i \leq k < l < j$.
- Une inversion $\rho(i,j)$ est *externe* à un intervalle $[k,l]$ si $i \leq j < k$ ou alors si $l < i \leq j$.
- Une inversion $\rho(i,j)$ *chevauche* un intervalle $[k,l]$ si $\rho(i,j)$ n'est ni externe, ni interne, ni n'incorpore l'intervalle $[k,l]$.

Définition 5.1 *Pour un ensemble d'intervalles fixé, une inversion qui ne chevauche aucun de ces Intervalles est dite une inversion* conforme.

Définition 5.2 *Pour π et σ deux permutations et C un ensemble d'Intervalles communs associé à ces deux permutations,* un scénario conforme, *$s = \rho_1 ... \rho_d$, est une suite d'inversions conformes à C telle que : $\pi.\rho_1 ... \rho_d = \sigma$.*

On appellera un scénario conforme un *CSS*, pour *Compliant Sorting Scenario*.

Définition 5.3 *Soit π et σ deux permutations, C un ensemble d'Intervalles communs associé à ces deux permutations et S l'ensemble des scénarios conformes à π, σ et C. On définit la distance conforme de π à σ par la fonction $d_C(\pi, \sigma)$:*

$$d_C(\pi, \sigma) = \min_{s \in S} |s|$$

Remarque 5.1 *Lorsque σ est la permutation identité, on parle du tri de π vers σ et on note $d_C(\pi)$ le nombre minimum d'inversions conformes nécessaires pour trier π.*

$$d_C(\pi, \sigma) = d_C(\pi)$$

Nous notons C un ensemble d'Intervalles communs, alors que l'ensemble de tous les Intervalles communs à deux permutations est noté C^*, $C \subseteq C^*$.

Nous montrons dans le paragraphe suivant que la fonction d_C est une bien une distance au sens mathématique du terme. Il est trivial que pour toute permutation π on a $d_C(\pi, \pi) = d_C(\sigma) = 0$ avec σ la permutation identité. De même pour deux permutations π et σ de taille n fixé, π et σ différentes, le chemin le plus court entre ces deux permutations nécessite au moins une inversion. On peut vérifier que pour toute permutation π et σ différentes, $d_C(\pi, \sigma) > 0$. Les Intervalles communs de π et σ sont les mêmes que ceux de σ et π. Les contraintes ajoutées étant les mêmes et comme si $\pi.\rho_1.....\rho_d = \sigma$ on a $\sigma.\rho_d.....\rho_1 = \pi$, la distance conforme est donc symétrique. Le test de l'inégalité triangulaire est plus délicat. Soit trois permutations π,σ et γ de taille n fixé et C un ensemble d'Intervalles communs (aux trois permutations), alors $d_C(\pi, \gamma) + d_C(\gamma, \sigma) < d_C(\pi, \sigma)$ est impossible car à ce moment là, les Intervalles communs étant les mêmes pour les trois permutations, le scénario conforme le plus court entre π et σ passe par γ et on obtient $d_C(\pi, \sigma) = d_C(\pi, \gamma) + d_C(\gamma, \sigma)$. Il est intéressant de noter que le C de d_C fait bien le lien avec l'ensemble C des Intervalles communs. Si pour chaque comparaison deux à deux nous redéfinissons C, il existe π, σ et γ trois permutations de taille n telles que $d_{C_1}(\pi, \sigma) \geq d_{C_2}(\pi, \gamma) + d_{C_3}(\gamma, \sigma)$. Il suffit de bien choisir π et σ avec des Intervalles communs et γ de telle sorte qu'elle ne partage pas ces Intervalles communs ($C_2 = C_3 = \{\emptyset\}$).

Nous pouvons maintenant définir formellement le problème :

Pour deux permutations π et σ, σ étant la permutation identité, et un ensemble d'Intervalles communs de π et σ nommé C, nous voulons calculer $d_C(\pi)$, le nombre minimum d'inversions conformes à C, nécessaires pour trier π.

Définition 5.4 *Pour C un ensemble d'Intervalles communs à π et σ deux permutations, un scénario qui transforme π en la permutation identité (σ), avec $d_C(\pi)$ inversions conformes est appelé un* MCSS*, pour Minimal Compliant Sorting Scenario.*

Nous montrons dans cette section qu'un MCSS est différent d'un scénario parcimonieux, que les permutations en jeu soient signées ou non. Non seulement tous les scénarios évolutifs

parcimonieux ne respectent pas les Intervalles communs, mais nous montrons que parfois il n'existe aucun scénario évolutif parcimonieux avec toutes ses inversions conformes aux Intervalles Communs.

Cas des permutations signées.

Nous noterons $d(\pi^s)$ le nombre minimum d'inversions nécessaires pour trier π^s une permutation signée, que nous opposons à $d_C(\pi^s)$ le nombre minimum d'inversions conformes nécessaires pour trier π^s en respectant C, un ensemble d'Intervalles communs donné. Il est évident que $d_C(\pi^s)$ est au moins aussi grande que $d(\pi^s)$ car cette dernière donne lieu à un scénario évolutif parcimonieux sans aucune autre contrainte. Nous savons d'après [Kececioglu and Sankoff, 1995], qu'il existe toujours un scénario parcimonieux qui ne crée pas de point de cassure. Donc si les Intervalles communs n'ont aucun point de cassure, aucune inversion de ce scénario ne chevauchera un Intervalle commun. Par conséquent, le scénario évolutif parcimonieux sera aussi un MCSS. Par exemple, nous pouvons prendre les deux permutations suivantes, $\pi^s =$ -5 +1 **-9** **-8** **-7** +4 +6 **-3** **-2** et σ la permutation identité. Ces permutations ont deux Intervalles communs autres que la permutation elle-même : $c_1 = [3, 5]$ avec $\pi^s([3, 5]) = \{7, 8, 9\}$ et $c_2 = [8, 9]$ avec $\pi^s([8, 9]) = \{2, 3\}$. Ces deux Intervalles communs sont représentés en gras sur les permutations. Nosu prenons $C = \{c_1, c_2\}$. Dans la représentation des scénarios parcimonieux nous annonçons une inversion d'éléments en soulignant ceux-ci. Un scénario parcimonieux est le suivant :

$$
\begin{array}{rcccccccccc}
\pi^s & = & \text{-5} & \text{+1} & \underline{\textbf{-9}} & \underline{\textbf{-8}} & \underline{\textbf{-7}} & \text{+4} & \text{+6} & \textbf{-3} & \textbf{-2} \\
 & \rightarrow & \text{-5} & \text{+1} & \textbf{+2} & \textbf{+3} & \underline{\text{-6}} & \underline{\text{-4}} & \textbf{+7} & \textbf{+8} & \textbf{+9} \\
 & \rightarrow & \text{-5} & \underline{\text{+1}} & \underline{\textbf{+2}} & \underline{\textbf{+3}} & \underline{\text{+4}} & \text{+6} & \textbf{+7} & \textbf{+8} & \textbf{+9} \\
 & \rightarrow & \underline{\text{-5}} & \underline{\text{-4}} & \underline{\textbf{-3}} & \underline{\textbf{-2}} & \underline{\text{-1}} & \text{+6} & \textbf{+7} & \textbf{+8} & \textbf{+9} \\
 & \rightarrow & \text{+1} & \text{+2} & \textbf{+3} & \textbf{+4} & \text{+5} & \text{+6} & \textbf{+7} & \textbf{+8} & \textbf{+9}
\end{array}
$$

Ce scénario respecte les Intervalles communs de C, ce qui montre qu'il existe des permutations signées π^s telles que $d_C(\pi^s) = d(\pi^s)$. Nous avons aussi montré qu'il n'existe pas de permutation signée π^s telle que $d_C(\pi^s) < d(\pi^s)$. Nous donnons maintenant un exemple qui montre qu'il existe des permutations signées π^s telles que $d_C(\pi^s) > d(\pi^s)$.

Prenons la permutation $\pi^s =$ -2 -3 +1, une permutation signée et linéaire. Soit $C = \{[1, 2], [1, 3]\}$ l'ensemble des Intervalles communs de π^s par rapport à la permutation identité. Ces deux Intervalles communs sont donc constitués des éléments $\{2, 3\}$ pour le premier et des éléments $\{1, 2, 3\}$ pour le deuxième. Il n'existe qu'un seul scénario évolutif parcimonieux qui tri π^s :

$$
\begin{array}{rcccc}
\pi^s = & \textbf{-2} & \underline{\textbf{-3}} & \underline{\text{+1}} \\
\rightarrow & \underline{\textbf{-2}} & \underline{\text{-1}} & \textbf{+3} \\
\rightarrow & \text{+1} & \textbf{+2} & \textbf{+3}
\end{array}
$$

Ce scénario nous donne $d(\pi^s)$ égal à deux. La première inversion n'est pas conforme à l'ensemble des Intervalles communs car elle sépare l'élément 3 de l'élément 2. Il existe un MCSS avec cinq inversions (conformes) :

$$\begin{aligned}
\pi^s = & \underline{\textbf{-2} \quad \textbf{-3} \quad +1} \\
\rightarrow & \underline{\text{-1}} \quad +\textbf{3} \quad +\textbf{2} \\
\rightarrow & +1 \quad \underline{+\textbf{3}} \quad +\textbf{2} \\
\rightarrow & +1 \quad \textbf{-3} \quad \underline{+\textbf{2}} \\
\rightarrow & +1 \quad \underline{\textbf{-3} \quad \textbf{-2}} \\
\rightarrow & +1 \quad +\textbf{2} \quad +\textbf{3}
\end{aligned}$$

Le théorème en découle :

Théorème 5.1 *Pour toute permutation signée π^s et C un ensemble d'Intervalles communs à π^s et à la permutation identité, $d_C(\pi^s) \geq d(\pi^s)$.*

Cas des permutations non signées.

D'après [Hannenhalli and Pevzner, 1996], nous savons que pour certaines permutations non signées, tous les scénarios évolutifs parcimonieux créent de nouveaux points de cassure. Comme toute sous-suite de gènes sans point de cassure est un Intervalle commun, nous savons qu'il existe des permutations non signées π^u telles que pour C un ensemble d'Intervalles communs à π^u et à la permutation identité, on a $d_C(\pi^u) > d(\pi^u)$. Or nous savons évidemment, comme pour les permutations signées, qu'il n'existe pas de permutation non signée π^u et des ensembles d'Intervalles communs C, tel que $d_C(\pi^u) < d(\pi^u)$. Le théorème en découle :

Théorème 5.2 *Pour toute permutation non signée π^u et C un ensemble d'Intervalles communs à π^u et à la permutation identité, $d_C(\pi^u) \geq d(\pi^u)$.*

5.2 Scénario évolutif minimal avec contraintes

Nous nous intéressons dorénavant exclusivement aux permutations signées. Nous avons décomposé la résolution du problème du MCSS en trois phases. Dans la première étape, nous limitons les Intervalles communs, en omettant certains d'entre eux, afin de présenter un algorithme trivial. Dans la deuxième étape, nous améliorons l'algorithme naïf proposé et nous amenons quelques modifications pour la phase suivante. La dernière étape expose la résolution du problème complet. Entre la première et la dernière étape nous avons inclus des optimisations permettant d'améliorer les calculs.

Pour distinguer les différents Intervalles communs, nous avons besoin de quelques définitions supplémentaires :

Définition 5.5 *Pour π une permutation avec $[i,j]$ et $[k,l]$ deux Intervalles communs de π et de la permutation identité, on dit que $[k,l]$* incorpore *$[i,j]$ et que $[i,j]$ est* inclus *dans $[k,l]$ si et seulement si $k \leq i \leq j \leq l$*

Définition 5.6 *Pour π une permutation avec $[i,j]$ et $[k,l]$ deux Intervalles communs de π et de la permutation identité, on dit que $[k,l]$* chevauche *$[i,j]$ et que $[i,j]$ chevauche $[k,l]$ si et seulement si $i < k \leq j < l$ ou $k < i \leq l < j$.*

Définition 5.7 *Pour π une permutation avec $[i,j]$ et $[k,l]$ deux Intervalles communs de π et de la permutation identité, on dit que $[k,l]$ et $[i,j]$ s'*intersectent *si l'un des deux chevauche l'autre ou si l'un inclut l'autre.*

Nous pouvons définir les mêmes relations pour des inversions entre elles :

Définition 5.8 *Pour deux inversions $\rho(i,j)$ et $\rho(k,l)$, ont dit que $\rho(i,j)$* incorpore *$\rho(k,l)$ et que $\rho(k,l)$ est* inclus *dans $\rho(i,j)$ si et seulement si $k \leq i \leq j \leq l$.*

Définition 5.9 *Pour deux inversions $\rho(i,j)$ et $\rho(k,l)$, ont dit que $\rho(i,j)$* chevauche *$\rho(k,l)$ et que $\rho(k,l)$ chevauche $\rho(i,j)$ si et seulement si $i < k \leq j < l$ ou $k < i \leq l < j$.*

Définition 5.10 *Pour deux inversions $\rho(i,j)$ et $\rho(k,l)$, ont dit que $\rho(i,j)$ et $\rho(k,l)$ s'intersectent si l'une des deux chevauche l'autre ou si l'une est incluse dans l'autre.*

Quand les Intervalles communs ne se chevauchent pas

Nous commençons l'analyse du MCSS avec un sous problème où nous considérons que les Intervalles communs ne se chevauchent pas. Nous omettons ces Intervalles communs pour limiter les dépendances entre eux pendant le calcul d'un MCSS. L'idée est de trier dans un premier temps les Intervalles communs pour éliminer leurs points de cassure, puis étant donné qu'il existe toujours un scénario parcimonieux pour les permutations signées qui ne crée pas de point de cassure, nous n'avons plus qu'à utiliser un de ces scénarios pour trier la permutation résultante. Les Intervalles communs étant déjà triés, ils ne seront pas cassés par des inversions non conformes.

Dans un premier temps, nous montrons que nous pouvons trier tous les Intervalles communs avant de trier la permutation résultante. Autrement dit, toutes les premières inversions seront internes aux Intervalles communs et les suivantes seront externes ou incluront ceux-ci. Bien sûr comme les Intervalles communs peuvent être inclus les uns dans les autres, il s'agit de trier dans un premier temps les Intervalles communs inclus dans d'autres. On admettra que quelque soit la permutation π^s et C un ensemble d'Intervalle commun, qu'il existe toujours un scénario triant π conforme à C.

Théorème 5.3 *Pour une permutation signée et un ensemble d'Intervalles communs ne se chevauchant pas, il existe toujours un MCSS qui trie d'abord les Intervalles communs avant d'appliquer des inversions qui les incorporent ou des inversions externes.*

Preuve : Soit π^s une permutation signée et $\rho_1. \ldots .\rho_k.\rho_{k+1}. \ldots .\rho_d$ un MCSS de π^s. On suppose que ρ_{k+1} est la première inversion interne à l'intervalle commun $[x, y]$. On note $\pi^{s\prime} = \pi^s.\rho_1. \ldots .\rho_{k-1}$. Nous montrons que $\pi^{s\prime}.\rho_k.\rho_{k+1} = \pi^{s\prime}.\rho_{k+1}.\rho_k$.

- *Si ρ_k et ρ_{k+1} se chevauchent, comme ρ_k n'est pas interne à $[x, y]$ c'est que ρ_k chevauche $[x, y]$, et donc ρ_k n'est pas conforme. Ce qui est impossible dans un MCSS.*
- *Sinon, c'est que ρ_k et ρ_{k+1} ne se chevauchent pas et nous avons $\pi^{s\prime}.\rho_k.\rho_{k+1} = \pi^{s\prime}.\rho_{k+1}.\rho_k$.*

En itérant ce processus, nous obtenons un MCSS qui trie tous les Intervalles communs avant d'appliquer des inversions qui les incorporent ou des inversions externes. ◀

Ce théorème montre que le pseudo algorithme décrit plus haut peut trouver un scénario conforme aux intervalles communs si ceux-ci ne se chevauchent pas. Cet algorithme est capable de trouver un scénario évolutif conforme (CSS) en triant d'abord les Intervalles communs mais pas forcément un MCSS. Les paragraphes suivants montrent que le scénario trouvé n'est pas toujours minimal. Cependant, en triant avec attention les Intervalles communs, il est possible de toujours trouver un MCSS.

Prenons la permutation signée linéaire suivante : $\pi^s = +3\ +2\ +5\ +1\ +4$, qui a deux Intervalles communs $[1, 2]$ et $[1, 5]$. L'Intervalle commun $[1, 2]$ correspond aux éléments $\{2, 3\}$ de π^s et l'Intervalle commun $[1, 5]$ correspond aux éléments $\{1, 2, 3, 4, 5\}$. En appliquant l'algorithme précédent, nous pouvons commencer par trier la sous-permutation $+3\ +2$ vers $+2\ +3$. Nous avons besoin d'au moins trois inversions pour trier cette sous-permutation. La permutation résultante est $\pi^{s\prime} = +2\ +3\ +5\ +1\ +4$, et il reste un seul Intervalle commun à trier, lequel est la permutation elle même. Trier cette permutation demande au moins cinq autres inversions. Soit un total de huit inversions.

$$
\begin{aligned}
\pi^s = & \underline{\mathbf{+3}\ \ \mathbf{+2}}\ +5\ +1\ +4 \\
\rightarrow & \underline{\mathbf{-2}}\ \mathbf{-3}\ +5\ +1\ +4 \\
\rightarrow & +\mathbf{2}\ \underline{\mathbf{-3}}\ +5\ +1\ +4 \\
\rightarrow & +\mathbf{2}\ +\mathbf{3}\ +5\ \underline{+1}\ +4 \\
\rightarrow & \underline{+\mathbf{2}\ +\mathbf{3}\ +5\ -1}\ +4 \\
\rightarrow & +1\ -5\ \underline{\mathbf{-3}\ \mathbf{-2}\ +4} \\
\rightarrow & +1\ \underline{-5\ -4\ +\mathbf{2}\ +\mathbf{3}} \\
\rightarrow & +1\ \underline{\mathbf{-3}\ \mathbf{-2}}\ +4\ +5 \\
\rightarrow & +1\ +\mathbf{2}\ +\mathbf{3}\ +4\ +5
\end{aligned}
$$

Au lieu de procéder de cette manière, nous aurions pu éliminer les points de cassure du premier Intervalle commun d'une autre manière. Recommençons en triant plutôt la sous permutation $+3\ +2$ vers -3 -2. Cela peut être fait avec seulement deux inversions et nous avons effectivement éliminé tous les points de cassure de la sous permutation. La permutation résultante, $\pi^{s\prime} = -3\ -2\ +5\ +1\ +4$, n'a besoin que de quatre nouvelles inversions pour être triée. Soit un total de six inversions.

$$
\begin{aligned}
\pi^s = & \underline{+\mathbf{3}} \quad +\mathbf{2} \quad +5 \quad +1 \quad +4 \\
\rightarrow & \text{ -}\mathbf{3} \quad \underline{+\mathbf{2}} \quad +5 \quad +1 \quad +4 \\
\rightarrow & \text{ -}\mathbf{3} \quad \text{ -}\mathbf{2} \quad +5 \quad \underline{+1} \quad +4 \\
\rightarrow & \underline{\text{ -}\mathbf{3} \quad \text{ -}\mathbf{2} \quad +5 \quad \text{ -}1} \quad +4 \\
\rightarrow & +1 \quad \text{ -}5 \quad \underline{+\mathbf{2} \quad +\mathbf{3} \quad +4} \\
\rightarrow & +1 \quad \underline{\text{ -}5 \quad \text{ -}4 \quad \text{ -}\mathbf{3} \quad \text{ -}\mathbf{2}} \\
\rightarrow & +1 \quad +\mathbf{2} \quad +\mathbf{3} \quad +4 \quad +5
\end{aligned}
$$

Dans les deux cas de figure, nous n'avons utilisé que des inversions conformes et nous avons respecté l'algorithme proposé précédemment. Le premier scénario proposé est en huit inversions alors que le deuxième n'en nécessite que six. Supprimer les points de cassure d'une permutation revient à la trier soit positivement, soit négativement. Cet exemple nous montre qu'il faut choisir avec attention si les Intervalles communs doivent être triés positivement et dans l'ordre croissant ou négativement et dans l'ordre décroissant.

Nous formalisons maintenant les propriétés illustrées ci-dessus. Nous réutilisons les notions de blocs et de strips définies dans [Hannenhalli and Pevzner, 1996] mais que nous adaptons aux permutations signées.

Définition 5.11 *Un* bloc d'une permutation signée π^s est un intervalle $[x, y]$, $x \leq y$, de π^s qui ne contient aucun point de cassure.

Définition 5.12 *Un* strip d'une permutation signée π^s est un bloc maximal, c'est-à-dire, un bloc $[x, y]$ tel qu'il existe un point de cassure entre ${\pi^s}_{x-1}$ et ${\pi^s}_x$ et entre ${\pi^s}_y$ et ${\pi^s}_{y+1}$.

Un bloc ou un strip $[x, y]$ est dit *croissant* si $||\pi_x|| < ||\pi_y||$ [5], sinon il est dit *décroissant*. Un strip croissant (respectivement décroissant) d'une permutation signée est dit *canonique* si tous ses éléments sont positifs (respectivement négatifs). Un strip canonique n'a aucun point de cassure et correspond donc à un intervalle trié, soit positivement, soit négativement.

Pour n arbitraire, nous définissons la permutation identité positive $\mathcal{I}^s_+ = +1 \ \ldots \ +n$ et la permutation identité négative $\mathcal{I}^s_- = \text{-}n \ \ldots \ \text{-}1$. La permutation identité est définie par rapport à une permutation signée quelconque de taille fixée n.

Pour un Intervalle commun $c = [x, y]$ relatif à une permutation signée π^s, nous définissons l et u tels que $l = \min(\pi^s([x, y]))$ et $u = \max(\pi^s([x, y]))$. Nous notons alors $+c = \mathcal{I}^s_+([l, u])$ l'Intervalle commun $[x, y]$ trié de manière à ce qu'il soit canonique croissant. Et de même nous notons $\text{-}c = \mathcal{I}^s_-([l, u])$ l'Intervalle commun $[x, y]$ trié de manière à ce qu'il soit canonique décroissant.

[5] Nous rappelons au lecteur que $||a||$ est la valeur absolue de a.

Par exemple, pour une permutation signée $\pi^s = {}_+5\ {}_+1\ \text{-}7\ \text{-}3\ {}_+2\ \text{-}4\ \text{-}6$ et l'Intervalle commun $c = \pi^s([4,6]) = \{2,3,4\}$, alors $l = \min(\{2,3,4\}) = 2$, $u = \max(\{2,3,4\}) = 4$, et ${}_+c = \mathcal{I}^s_+([2,4]) = {}_+2\ {}_+3\ {}_+4$ et $\text{-}c = \mathcal{I}^s_-([2,4]) = \text{-}4\ \text{-}3\ \text{-}2$.

Éliminer les points de cassure d'un Intervalle commun, revient à trier une sous permutation linéaire, soit vers l'identité positive, soit vers l'identité négative. Pour éliminer les points de cassure de c, un Intervalle commun, il faut soit le trier vers ${}_+c$, soit vers $\text{-}c$.

Lemme 5.1 *Pour π^s une permutation signée, $c = [x+1, y-1]$ un Intervalle commun à π^s et à la permutation identité et C un ensemble d'Intervalles communs à π^s et à la permutation identité avec $C = \{c\}$,*

$$d_C(\pi^s) = \min \begin{cases} d(c, {}_+c) + d(\pi^s([1,x]).{}_+c.\pi^s([y,n])) \\ d(c, \text{-}c) + d(\pi^s([1,x]).\text{-}c.\pi^s([y,n])) \end{cases}$$

Preuve : Soit s le scénario lié à l'obtention de la distance d_C. D'après le théorème 5.3, on peut faire remonter les inversions internes à c. Donc on peut modifier le scénario s pour qu'il soit de la forme $s = s_i.s_c$ avec d'abord les inversions internes, puis les inversions incorporante et externe. Donc s après réordonnement, passe soit par la permutation $\pi^s([1,x]).{}_+c.\pi^s([y,n])$, soit par la permutation $\pi^s([1,x]).\text{-}c.\pi^s([y,n])$. Donc $\min \begin{cases} d(c, {}_+c) + d(\pi^s([1,x]).{}_+c.\pi^s([y,n])) \\ d(c, \text{-}c) + d(\pi^s([1,x]).\text{-}c.\pi^s([y,n])) \end{cases}$ est minimal et ne casse pas c. ◄

Nous pouvons clore cette première partie en proposant un algorithme exact pour calculer le nombre minimum d'inversions conformes pour trier une permutation signée π^s étant donné un ensemble d'Intervalles communs ne se chevauchant pas. Nous avons nommé cet algorithme SRIC (algorithme 5.2) pour *Sorting by Reversals without any Interlacing Common intervals*.

entrée : π^s, C trié par taille croissante de ses éléments
début
1. **Si** $C = \{\emptyset\}$ **Alors**
2. retourne $d(\pi^s)$
3. **Sinon**
4. $c := C[0]$ /* $c := \pi^s([x+1, y-1])$ */
5. retourne $\min \begin{cases} d(c, {}_+c) + SRIC(\pi^s([1,x]).{}_+c.\pi^s([y,n]), C \setminus \{c\}) \\ d(c, \text{-}c) + SRIC(\pi^s([1,x]).\text{-}c.\pi^s([y,n]), C \setminus \{c\}) \end{cases}$
6. **FinSi**

Algorithme 6: Algorithme naïf (SRIC).

Théorème 5.4 *SRIC calcule $d_C(\pi^s)$ pour C un ensemble d'Intervalles communs non chevauchants trié par taille croissante de ses éléments.*

Preuve : Nous montrons que pour π^s, une permutation signée, et C, l'ensemble associé d'Intervalles communs non chevauchants, $\mathrm{SRIC}(\pi^s, C) = d_C(\pi^s)$. *Nous le montrons par un raisonnement par récurrence sur la taille de l'ensemble C. Soit une permutation signée π^s et C avec $|C| = k$ un ensemble d'Intervalles communs de π^s, en rapport avec la permutation identité, de taille au plus m. Notre hypothèse de récurrence est que* $d_C(\pi^s) = \mathrm{SRIC}(\pi^s, C)$. *Nous utiliserons la notation* $\mathrm{SRIC}(\pi^s \to \pi^{s\prime}, C)$ *et* $d_C(\pi^s \to \pi^{s\prime})$ *lorsque nous voulons préciser que la distance calculée à partir de π^s n'est pas vers la permutation identité mais vers $\pi^{s\prime}$. Cela ne modifie en rien l'algorithmique étant donné que nous pouvons renommer les permutations pour obtenir la distance vers la permutation identité.*

Pour $|C| = 0$, *il est évident que* $\mathrm{SRIC}(\pi^s, \{\emptyset\}) = d(\pi^s) = d_{\{\emptyset\}}(\pi^s)$.

Soit notre hypothèse de récurrence : $|C| = k$ *et* $SRIC(\pi^s, C) = d_C(\pi^s)$.

Nous construisons maintenant C' en ajoutant à C un Intervalle commun de taille supérieure ou égale à m, m étant la taille du plus grand Intervalle commun de C. Cet Intervalle est commun à π^s, la même permutation que précédemment, et à la permutation identité. Nous allons montrer que $\mathrm{SRIC}(\pi^s, C') = d_{C'}(\pi^s)$. *Revenons dans un premier temps sur l'hypothèse de récurrence. Nous pouvons écrire que*

$$\mathrm{SRIC}(\pi^s, C) = \min_{\pi \in \Pi} \left(\mathrm{SRIC}\left(\pi^s \to \pi, C\right) + d(\pi)\right).$$

Π *étant l'ensemble des* 2^k *permutations rencontrées après le tri des k Intervalles communs lors de l'application de l'algorithme* SRIC. *De même nous pouvons écrire :*

$$\mathrm{SRIC}(\pi^s, C') = \min_{\pi \in \Pi} \left(\mathrm{SRIC}\left(\pi^s \to \pi, C\right) + \mathrm{SRIC}\left(\pi, \{c_{k+1}\}\right)\right).$$

Or d'après l'hypothèse de récurrence, $\mathrm{SRIC}\left(\pi^s \to \pi, C\right) = d_C(\pi^s \to \pi)$, *et d'après le lemme 5.1,* $\mathrm{SRIC}\left(\pi, \{c_{k+1}\}\right) = d_{\{c_{k+1}\}}(\pi)$. *Donc*

$$\mathrm{SRIC}(\pi^s, C') = \min_{\pi \in \Pi} \left(d_C\left(\pi^s \to \pi\right) + d_{\{c_{k+1}\}}(\pi)\right).$$

Or d'après le théorème 5.3, il existe toujours un scénario conforme et minimal qui passe obligatoirement par une des permutations de Π, *et il est évident que le scénario inféré par* $SRIC(\pi^s, C')$ *est conforme, donc* $SRIC(\pi^s, C') = d_{C'}(\pi^s)$.

◀

L'algorithme est de complexité $O(2^p n)$ en temps avec n la taille de la permutation et p le nombre d'Intervalles communs non chevauchants. Dans le pire des cas, le nombre d'Intervalles communs non chevauchant est en $O(n)$, ce qui nous donne un algorithme en $O(2^n n)$. Mais en pratique, on peut s'attendre à avoir un petit nombre d'Intervalles communs non chevauchant.

Optimisation du sous problème

Nous avons présenté un algorithme naïf, SRIC, pour résoudre un sous problème du MCSS. Nous voulons maintenant améliorer sa complexité. Nous gardons les mêmes contraintes, c'est-à-dire qu'un scénario évolutif doit respecter les Intervalles communs, mais nous savons que ceux-ci ne se chevauchent pas. Nous nous intéressons exclusivement à la dernière étape

de l'algorithme SRIC, correspondant au double appel récursif pour un Intervalle commun $c = \pi^s([x+1, y-1])$:

$$\text{retourne } \min \begin{cases} d(c,{+}c) + SRIC(\pi^s([1,x]).{+}c.\pi^s([y,n]), C \setminus \{c\}) \\ d(c,{-}c) + SRIC(\pi^s([1,x]).{-}c.\pi^s([y,n]), C \setminus \{c\}) \end{cases}$$

$$\equiv$$

$$\text{retourne } \min \begin{cases} d(c,{+}c) + d_{C\setminus\{c\}}(\pi^s([1,x]).{+}c.\pi^s([y,n])) \\ d(c,{-}c) + d_{C\setminus\{c\}}(\pi^s([1,x]).{-}c.\pi^s([y,n])) \end{cases}$$

On notera $d_1 = d_{C\setminus\{c\}}(\pi([1,x]).{+}c.\pi([y,n])) + d(c,{+}c)$ et $d_2 = d_{C\setminus\{c\}}(\pi([1,x]).{-}c.\pi([y,n])) + d(c,{-}c)$. Nous voulons améliorer le calcul de d_C au moment de l'appel récursif : $d_C(\pi([1,x]).c.\pi([y,n])) = \min(d_1, d_2)$. Nous pouvons pour quelques Intervalles communs éliminer le double appel récursif. Certains Intervalles communs seront toujours triés vers un strip canonique croissant alors que d'autres seront toujours triés vers un strip canonique décroissant. Nous présentons d'abord des propriétés sur les Intervalles communs puis quelques modifications au niveau de l'approche présentée pour l'algorithme SRIC. L'élimination d'un des deux appels récursifs repose sur les remarques triviales suivantes :

Remarque 5.2 *Pour une permutation signée π^s de taille n et ρ une inversion quelconque sur π^s, on a $d(\pi^s.\rho) - d(\pi^s) \in \{-1, 0, 1\}$*

Cette remarque entraîne immédiatement la seconde remarque suivante :

Remarque 5.3 *Pour une permutation signée π^s de taille n, C un ensemble d'Intervalles communs et $c = \pi^s([x+1, y-1]) \in C$, on a $d_C\big(\pi^s([1,x]).{+}c.\pi^s([y,n])\big) - d_C(\pi^s([1,x]).{-}c.\pi^s([y,n]) \in \{-1,0,1\}$ si cette inversion est conforme à C.*

Lemme 5.2 *Soit une permutation signée π^s de taille n, C un ensemble d'Intervalles communs sans intersection et $c = \pi^s([x+1, y-1]) \in C$ n'incorporant aucun autre Intervalle commun. Si $d(c,{+}c) = d(c,{-}c)+1$, alors $d_C(\pi^s([1,x]).c.\pi^s([y,n])) = d(c,{-}c) + d_{C\setminus\{c\}}(\pi^s([1,x]).{-}c.\pi^s([y,n]))$.*

Preuve : Soit $C' = C \setminus \{c\}$, comme c n'incorpore aucun autre Intervalle commun, d'après le théorème 5.4, $d_C(\pi^s([1,x]).c.\pi^s([y,n])) = \min(d_1, d_2)$ avec

$$d_1 = d_{C'}(\pi^s([1,x]).{+}c.\pi^s([y,n])) + d(c,{+}c) \quad et$$

$$d_2 = d_{C'}(\pi^s([1,x]).{-}c.\pi^s([y,n])) + d(c,{-}c).$$

Nous montrons que $d_2 \leq d_1$.
Soit $k = d_{C'}(\pi^s([1,x]).{+}c.\pi^s([y,n]))$ et $l = d_{C'}(\pi^s([1,x]).{-}c.\pi^s([y,n]))$. Nous avons $d_1 = k + d(c,{+}c) = k + d(c,{-}c) + 1$ et $d_2 = l + d(c,{-}c)$, or d'après la remarque 5.3, $k = l + \varepsilon$, $\varepsilon \in \{-1, 0, 1\}$.

- *si $k = l - 1$ alors $d_1 = d(c,{-}c) + 1 + k = d(c,{-}c) + l = d_2$;*
- *si $k = l + 1$ alors $d_1 = d(c,{-}c) + 1 + k = d(c,{-}c) + l + 2 > d_2$;*

- *si* $k = l$ *alors* $d_1 = d(c, \text{-}c) + 1 + k = d(c, \text{-}c) + 1 + l > d_2$.

Nous avons montré que $d_2 \leq d_1$ *et donc que*
$d_C(\pi^s([1,x]).c.\pi^s([y,n])) = d(c, \text{-}c) + d_{C'}(\pi^s([1,x]).\text{-}c.\pi^s([y,n]))$. ◀

Nous avons symétriquement :

Lemme 5.3 *Soit une permutation signée* π^s *de taille* n, C *son ensemble d'Intervalles communs ne s'intersectant pas et* $c \in C$ *n'incorporant aucun autre Intervalle commun. Si* $d(c, {+}c) = d(c, \text{-}c) - 1$, *alors* $d_C(\pi^s([1,x]).c.\pi^s([y,n])) = d(c, {+}c) + d_C(\pi^s([1,x]).{+}c.\pi^s([y,n]))$

En utilisant les lemmes 5.2 et 5.3, nous pouvons caractériser les Intervalles communs :

Définition 5.13 *Un Intervalle commun* c *est dit* neutre *si* $d(c, {+}c) = d(c, \text{-}c)$.

Définition 5.14 *Un Intervalle commun* c *est dit* positif *si* $d(c, {+}c) < d(c, \text{-}c)$.

Définition 5.15 *Un Intervalle commun* c *est dit* négatif *si* $d(c, {+}c) > d(c, \text{-}c)$.

Pour π^s une permutation signée quelconque, le calcul de $d(\pi^s)$ demande un temps linéaire en fonction de la taille de π^s. Donc tester la neutralité d'un Intervalle commun peut être effectué en temps linéaire par rapport à sa taille. Pour tout Intervalle commun non neutre, nous savons, grâce aux lemmes 5.2 et 5.3 comment le signer de manière optimale pour ne pas appeler deux fois récursivement l'algorithme SRIC. Pour un Intervalle commun neutre, cela dépend du reste de la permutation. Nous utilisons cette propriété dans l'algorithme ASRIC pour diminuer la complexité de l'algorithme naïf, SRIC.

Théorème 5.5 *Pour une permutation signée* π^s *et* C *son ensemble d'Intervalles communs ne se chevauchant pas et trié par taille croissante de ses éléments, l'algorithme* ASRIC *calcule* $d_C(\pi^s)$.

Preuve :

Nous montrons que pour π^s, *une permutation signée, et* C, *l'ensemble associé d'Intervalles communs non chevauchant,* $\text{ASRIC}(\pi^s, C) = d_C(\pi^s)$. *Nous le montrons par un raisonnement par récurrence sur la taille de l'ensemble* C. *Soit une permutation signée* π^s *et* C *avec* $|C| = k$ *un ensemble d'Intervalles communs de* π^s, *en rapport avec la permutation identité, de taille au plus* m. *Notre hypothèse de récurrence est que* $d_C(\pi^s) = \text{ASRIC}(\pi^s, C)$. *Nous utiliserons la notation* $\text{ASRIC}(\pi^s \rightarrow \pi^{s\prime}, C)$ *et* $d_C(\pi^s \rightarrow \pi^{s\prime})$ *lorsque nous voulons préciser que la distance calculée à partir de* π^s *n'est pas vers la permutation identité mais vers* $\pi^{s\prime}$. *Cela ne modifie en rien l'algorithmique étant donné que nous pouvons renommer les permutations pour obtenir la distance vers la permutation identité.*

Pour $|C| = 0$, *il est évident que* $\text{ASRIC}(\pi^s, \{\emptyset\}) = d(\pi^s) = d_{\{\emptyset\}}(\pi^s)$.

Soit notre hypothèse de récurrence : $|C| = k$ *et* $\text{ASRIC}(\pi^s, C) = d_C(\pi^s)$.

entrée : π^s, C trié par taille croissante de ses éléments
début
1. **Si** $C = \{\emptyset\}$ **Alors**
2. retourne $d(\pi^s)$
3. **Sinon**
4. $c \coloneqq C[0]$ /* $c \coloneqq \pi^s([x+1, y-1])$ */
5. **Si** $d(c,{+}c) < d(c,{-}c)$ **Alors**
6. retourne $d(c, \mathcal{I}^s_+) + \text{ASRIC}(\pi^s([1,x])._+c.\pi^s([y,n]), C \setminus \{c\})$
7. **Sinon Si** $d(c,{+}c) > d(c,{-}c)$ **Alors**
8. retourne $d(c, \mathcal{I}^s_-) + \text{ASRIC}(\pi^s([1,x])._-c.\pi^s([y,n]), C \setminus \{c\})$
9. **Sinon**
10. retourne $\min \begin{cases} d(c,{+}c) + ASRIC(\pi^s([1,x])._+c.\pi^s([y,n]), C \setminus \{c\}) \\ d(c,{-}c) + ASRIC(\pi^s([1,x])._-c.\pi^s([y,n]), C \setminus \{c\}) \end{cases}$
11. **FinSi**
12. **FinSi**

Algorithme 7: L'algorithme ASRIC.

Nous construisons maintenant C' en ajoutant à C un Intervalle commun de taille supérieure ou égale à m, m étant la taille du plus grand Intervalle commun de C. Cet Intervalle est commun à π^s, la même permutation que précédemment, et à la permutation identité. Nous allons montrer que $\text{ASRIC}(\pi^s, C') = d_{C'}(\pi^s)$. *Revenons dans un premier temps sur l'hypothèse de récurrence. Nous pouvons écrire que*

$$\text{ASRIC}(\pi^s, C) = \min_{\pi \in \Pi} \left(\text{ASRIC}\left(\pi^s \to \pi, C\right) + d(\pi)\right).$$

Π *étant l'ensemble des* $2^{k'}$ *permutations rencontrées après le tri des* k *Intervalles communs lors de l'application de l'algorithme* ASRIC, $k' \leq k$. *De même nous pouvons écrire :*

$$\text{ASRIC}(\pi^s, C') = \min_{\pi \in \Pi} \left(\text{ASRIC}\left(\pi^s \to \pi, C\right) + \text{ASRIC}\left(\pi, \{c_{k+1}\}\right)\right).$$

Or d'après l'hypothèse de récurrence, $\text{ASRIC}\left(\pi^s \to \pi, C\right) = d_C(\pi^s \to \pi)$, *et d'après le lemme 5.1 et le lemme 5.2,* $\text{ASRIC}\left(\pi, \{c_{k+1}\}\right) = d_{\{c_{k+1}\}}(\pi)$. *Donc*

$$\text{ASRIC}(\pi^s, C') = \min_{\pi \in \Pi} \left(d_C\left(\pi^s \to \pi\right) + d_{\{c_{k+1}\}}(\pi)\right).$$

Or d'après le théorème 5.3 et le lemme 5.2, il existe toujours un scénario conforme et minimal passant par une des permutations de Π, *et il est évident que le scénario inféré par* $\text{ASRIC}(\pi^s, C')$ *est conforme, donc* $\text{ASRIC}(\pi^s, C') = d_{C'}(\pi^s)$.

◀

Estimation du nombre d'Intervalles communs neutres.

Nous faisons remarquer au lecteur que le nombre d'Intervalles communs neutres ne peut être pré-calculé parce que le type des Intervalles communs dépend du scénario de tri utilisé.

Un Intervalle commun non neutre au début du scénario, peut devenir neutre après le tri des Intervalles communs inclus dans celui-ci.

Nous estimons le nombre d'Intervalles communs neutres sans prendre en compte les sous Intervalles communs. Les Intervalles communs étant des sous-permutations, nous considérons le nombre de permutations neutres. Nous avons observé pour des données générées aléatoirement, que le nombre de permutations neutres croît avec la distance d'inversions de ces permutations. Nous avons aussi observé que pour toutes les permutations de taille $n \leq 100$, il y a pour chaque valeur de n moins de 34% de permutations neutres. Ces expériences sont décrites figures 5.1 et 5.2.

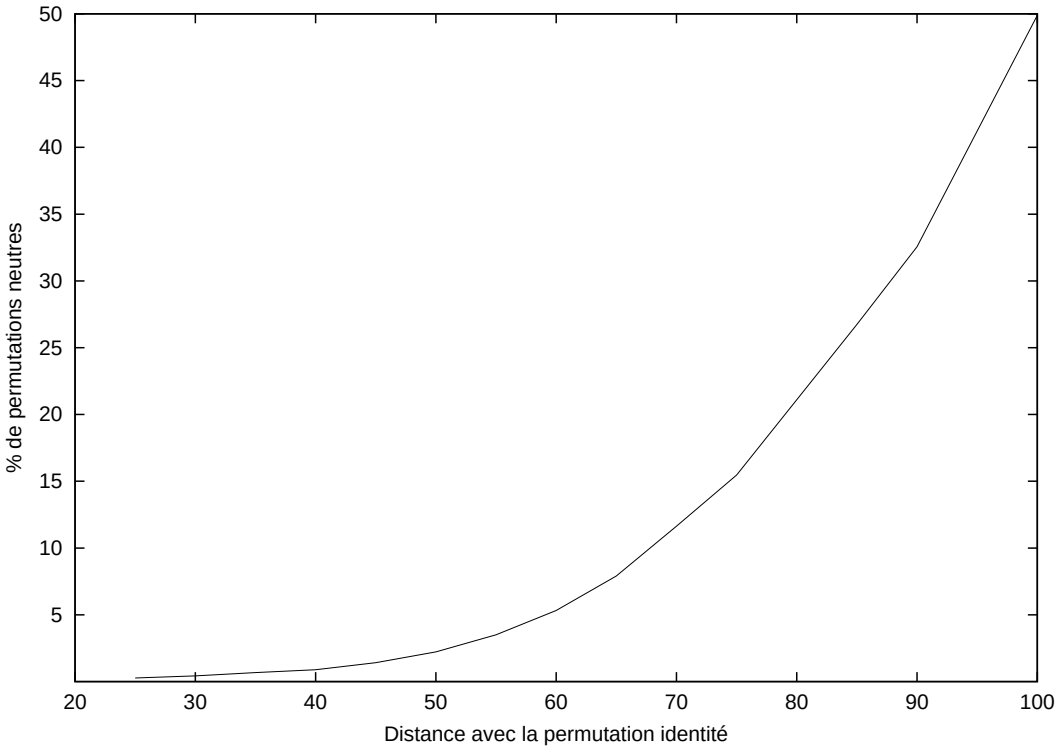


Figure 5.1: Pourcentage de permutations neutres pour des permutations générées aléatoirement. L'axe des abscisses donne la distance par inversions à l'identité positive pour des permutations de taille 100. L'axe des ordonnées, représente le pourcentage de permutations neutres. Pour chaque distance nous avons générés aléatoirement 10 000 permutations.

La résolution du problème complet

Nous traitons maintenant le problème complet où les Intervalles communs peuvent se chevaucher les uns les autres. Nous considérons ici que C, l'ensemble des Intervalles communs pris en compte, est égal à l'ensemble de tous les Intervalles communs entre les deux permutations comparées, c'est-à-dire $C = C^*$.

$\|\pi\|$	1	2	3	4	5	6	7	8	9	10	25	40	55	70	85	100
% neutres	0	0	29.2	27.1	29.4	29.2	29.7	29.9	30.2	30.3	32.5	32.1	32.6	33.0	32.9	33.1

Figure 5.2: Pourcentage de permutations neutres pour toutes les permutations de taille un à neuf. Pour des permutations de taille n strictement supérieure à 9, pourcentage de permutations neutres pour $n \times (n-1) \times (n-2)$ permutations aléatoires (en supposant l'équiprobabilité de toutes les permutations signées).

Nous traitons dans cette dernière partie le cas du calcul de la distance conforme lorsque nous prenons en compte l'ensemble de tous les Intervalles communs. Nous sommes limités à ce type de problème parce que lorsque nous avons des Intervalles communs chevauchants nous devons prendre en compte l'Intervalle commun englobant ainsi que ceux inclus. Or si ceux-ci ne sont pas présents dans C cela peut poser des problèmes. Mais on peut se demander ce que signifie de trier en respectant deux Intervalles communs chevauchants, sans prendre en compte ceux induits par ceux-ci. Prendre en compte deux Intervalles communs chevauchants signifie qu'il existe d'autres Intervalles communs reflétant les mêmes contraintes. S'il existe des contraintes liées à la conservation de deux Intervalles communs se chevauchant, il existe des contraintes liées à la conservation des Intervalles communs inclus et englobants.

Nous commençons par étudier les Intervalles communs issus de deux Intervalles communs se chevauchants :

Lemme 5.4 *Pour tout couple d'Intervalles communs $[i,j]$ et $[k,l]$ se chevauchant, il existe $[i,l]$ un troisième Intervalle commun.*

Preuve : Si nous renommons les éléments des Intervalles $[i,j]$ et $[k,l]$ en renumérotant leurs éléments dans l'intervalle des nombres de i à l. Nous obtenons alors l'ordre suivant sur ces Intervalles communs : $\min[i,j] < \min[k,l]$ *et* $max[i,j] < \max[k,l]$. *Or comme leurs éléments ont été renumérotés, nous avons* $\max[i,j] = j$ *et* $\min[i,j] = i$ *pour* $[i,j]$ *et* $\max[k,l] = l$ *et* $\min[k,l] = k$ *pour* $[k,l]$. *Nous avons alors* $\max[k,l] = \max[i,l]$ *parce que* $[k,l]$ *est inclus dans* $[i,l]$ *et* $\max[k,l] > \max[i,j]$. *De même, nous avons* $\min[i,j] = \min[i,l]$ *parce que* $[i,j]$ *est inclus dans* $[i,l]$ *et* $\min[i,j] < \min[k,l]$. *Finalement nous avons* $\max[i,l] - \min[i,l] = \max[k,l] - \min[i,j] = l - i$, *et donc d'après la propriété 1.1,* $[i,l]$ *est aussi un Intervalle commun.* ◀

Lemme 5.5 *Pour tout couple d'Intervalles communs $[i,j]$ et $[k,l]$ se chevauchant avec $k \leq j$, il existe, $[k,j]$, un troisième Intervalle commun.*

Preuve : Comme dans la preuve du lemme 5.4 nous avons : $\max[k,j] - \min[k,j] = \max[i,j] - \min[k,l] = j - k$ *et donc* $[k,j]$ *est un Intervalle commun.* ◀

Lemme 5.6 *Pour tout couple d'Intervalles communs, $[i,j]$ et $[k,l]$, se chevauchant, avec $k \leq j$, il existe trois autres Intervalles communs, $[i,k-1], [k,j]$ et $[j+1,l]$.*

Preuve : Nous savons par le lemme 5.5, qu'il existe $[k, j]$ un autre Intervalle commun. Si $[i, j]$ et $[k, j]$ sont deux Intervalles communs, alors nous avons aussi $[i, k-1]$ un Intervalle commun. De manière symétrique, nous avons aussi $[j+1, l]$ un Intervalle commun. ◀

On notera $u_i = \max ||b_i||$, l'élément le plus grand du bloc b_i en valeur absolue et $l_i = \min ||b_i||$ le plus petit.

Définition 5.16 *Un* bloc multiple *est une liste de blocs consécutifs $(b_i, \ldots, b_j)$ telle que pour tout $i \leq k < j$ $u_k + 1 = l_{k+1}$ ou $m_k - 1 = u_{k+1}$.*

Définition 5.17 *Pour $b_i, ..., b_j$ un bloc multiple, si pour tout $i \leq k < j$, $u_k + 1 = l_{k+1}$, nous l'appellerons un bloc multiple croissant, sinon nous l'appellerons un bloc multiple décroissant.*

Définition 5.18 *Un multi-strip, croissant ou décroissant, est un bloc multiple tel que ses blocs soient maximaux.*

Prenons par exemple la permutation signée $\pi^s = (+1\ +2\ +3\ -6\ -5\ -4\ -9\ -8\ -7)$, elle possède un multi-strip composé de trois blocs maximaux : $\pi^s([1,3]) = \{1,2,3\}, \pi^s([4,6]) = \{4,5,6\}$ et $\pi^s([7,9]) = \{7,8,9\}$.

Propriété 5.1 *Pour deux Intervalles communs s'imbriquant, $[i, j]$ et $[k, l]$, d'une permutation signée π^s, trier les trois Intervalles communs non intersectant $[i, k-1]$, $[k, j]$ et $[j+1, l]$ donne un multi-strip croissant ou décroissant. De même, le tri de $[i, k-1]$ et de $[k, j]$ donne un multi-strip croissant ou décroissant.*

Nous définissons maintenant l'opération COLLAPSE^{s}comme suit :

$$\text{COLLAPSE}^s : \Sigma_n \rightarrow \Sigma_{\leq n}$$

Σ_n est l'ensemble de toutes les permutations signées de taille n, et $\Sigma_{\leq n}$ est l'ensemble de toutes les permutations signées de taille n ou moins sans aucune adjacence. Les adjacences de plusieurs éléments sont comprimées en un seul élément et les éléments restant de la permutation sont renommés. Comme on peut trier optimalement les permutations signées sans casser d'adjacences, nous avons la propriété suivante : $d\big(\text{COLLAPSE}^s(\pi^s)\big) = d(\pi^s)$ [Bergeron et al., 2002a].

Lemme 5.7 *Trier un multi-strip croissant (respectivement décroissant) avec k blocs anti-canoniques (respectivement canoniques), en respectant les Intervalles communs, nécessite exactement k inversions retournant chacune un bloc anti-canonique (respectivement canonique).*

Preuve : Sans perdre en généralité nous supposons avoir un multi-strip m croissant constitué de deux blocs. Chaque bloc peut être comprimé en un seul élément. Nous obtenons quatre cas possibles suivant le signe de ces éléments :

[5] Pour rappel un bloc est un Intervalle commun sans point de cassure. Les blocs ont été introduits par les définitions 5.11 page 118.

- *si* $m = +1 \quad +2$, *il n'y a pas d'élément négatif, donc* $k = 0$ *et on n'a pas besoin d'inversion pour trier* m ;
- *si* $m = +1 \quad$ -2 *avec* $k = 1$, *alors* $d(+1 \quad$ -2$, +1 \quad +2) = 1$ *et* $d(+1 \quad$ -2$,$ -2 $\quad$ -1$) > 1$, *donc on peut trier* m *avec une inversion retournant le bloc anti-canonique ;*
- *si* $m =$ -1 $\quad +2$ *avec* $k = 1$, *on est dans la même situation que précédemment.*
- *si* $m =$ -1 $\quad$ -2 *avec* $k = 2$, *alors* $d($-1 $\quad$ -2$, +1 \quad +2) = 2$ *et* $d($-1 $\quad$ -2$,$ -2 $\quad$ -1$) > 2$, *donc on peut trier* m *avec deux inversion retournant les deux blocs anti-canoniques ;*

Dans aucun des cas nous n'avons déplacé les éléments les uns par rapport aux autres et nous avons donc respecté tous les Intervalles communs. ◀

Lemme 5.8 *Pour deux Intervalles communs* $[i, j]$ *et* $[k, l]$ *s'imbriquant avec* $k \leq j$, *les trier sans casser un seul Intervalle commun est optimalement réalisé en triant les trois Intervalles communs sans intersection :* $[i, k-1]$, $[k, j]$ *et* $[j+1, l]$, *et finalement en triant le multi-strip résultant.*

Preuve : Nous montrons d'abord que le scénario de tri proposé est conforme à tous les Intervalles communs. $[i, k-1]$ *est inclus dans* $[i, j]$ *et doit donc être trié avant* $[i, j]$. $[k, j]$ *est inclus dans* $[i, j]$ *et dans* $[k, l]$ *et, de même, doit être trié avant* $[i, j]$ *et avant* $[k, l]$. $[j+1, l]$ *est inclus dans* $[k, l]$ *et doit donc être trié avant* $[k, l]$. *Nous avons trois Intervalles communs sans intersection et qui doivent être triés en premier. Ce scénario ne casse aucun Intervalle commun et est donc conforme. La propriété 5.1 précise que le résultat de ces tris préliminaires est un multi-strip croissant ou décroissant. Le théorème 5.3 montre que ce scénario est optimal car nous avons respecté tous les niveaux d'inclusions sans avoir touché aux Intervalles communs se chevauchants.* ◀

Ces propriétés et ces lemmes permettent d'aboutir à l'algorithme SRAC(algorithme 5.2).

Théorème 5.6 *Pour une permutation signée* π^s *et tous ses Intervalles communs* C, SRAC *calcule* $d_C(\pi^s)$.

Preuve : Nous montrons que pour π^s, *une permutation signée, et* $C = C^*$, *l'ensemble associé d'Intervalles communs,* $\text{SRAC}(\pi^s, C) = d_C(\pi^s, C)$. *Nous le montrons avec un raisonnement par récurrence sur la taille de l'ensemble* C. *Soit une permutation signée* π^s *et* C *avec* $|C| = k$ *un ensemble d'Intervalles communs de* π^s, *en rapport avec la permutation identité, de taille au plus* m. *Notre hypothèse de récurrence est que* $d_C(\pi^s) = \text{SRAC}(\pi^s, C)$. *Nous utiliserons la notation* $\text{SRAC}(\pi^s \rightarrow \pi^{s\prime}, C)$ *et* $d_C(\pi^s \rightarrow \pi^{s\prime})$ *lorsque nous voulons préciser que la distance calculée à partir de* π^s *n'est pas vers la permutation identité mais vers* $\pi^{s\prime}$. *Cela ne modifie en rien l'algorithmique étant donné que nous pouvons renommer les permutations pour obtenir la distance vers la permutation identité.*

Pour $|C| = 0$, *il est évident que* $\text{SRAC}(\pi^s, \{\emptyset\}) = d(\pi^s) = d_{\{\emptyset\}}(\pi^s)$.

Soit notre hypothèse de récurrence : $|C| = k$ *et* $\text{SRAC}(\pi^s, C) = d_C(\pi^s)$.

Nous construisons maintenant C' *en ajoutant à* C *un Intervalle commun de taille supérieure ou égale à* m, m *étant la taille du plus grand Intervalle commun de* C. *Cet Intervalle est commun à* π^s, *la même permutation que précédemment, et à la permutation identité. Nous allons montrer que* $\text{SRAC}(\pi^s, C') = d_{C'}(\pi^s)$. *Revenons dans un premier temps sur l'hypothèse*

entrée : π^s et C trié par taille croissante de ses Intervalles communs
début
1. **Si** $C = \{\emptyset\}$ **Alors**
2. retourne $d(\pi^s)$
3. **Sinon**
4. $c \coloneqq C[0]$ $/* \; c \coloneqq \pi^s([x+1, y-1]) \; */$
5. **Si** c est un multi-strip croissant **Alors**
6. d :=nombre_de_blocs_négatifs (c)
7. retourne $d + \mathrm{SRAC}(\pi^{s1}.{+c}.\pi^{s2}, C \setminus \{c\})$
8. **Sinon Si** c est multi-strip décroissant **Alors**
9. d :=nombre_de_blocs_positifs (c)
10. retourne $d + \mathrm{SRAC}(\pi^{s1}.{-c}.\pi^{s2}, C \setminus \{c\})$
11. **Sinon Si** $d(c,{+c}) < d(c,{-c})$ **Alors**
12. retourne $d(c, \mathcal{I}^s_+) + \mathrm{SRAC}(\pi^{s1}.{+c}.\pi^{s2}), C \setminus \{c\}$
13. **Sinon Si** $d(c,{+c}) > d(c,{-c})$ **Alors**
14. retourne $d(c, \mathcal{I}^s_-) + \mathrm{SRAC}(\pi^{s1}.{-c}.\pi^{s2}, C \setminus \{c\})$
15. **Sinon**
16. retourne $\min \begin{cases} d(c,{+c}) + SRAC(\pi^{s1}.{+c}.\pi^{s2}, C \setminus \{c\}) \\ d(c,{-c}) + SRAC(\pi^{s1}.{-c}.\pi^{s2}, C \setminus \{c\}) \end{cases}$
17. **FinSi**
18. **FinSi**

Algorithme 8: L'algorithme SRAC.

de récurrence. Nous pouvons écrire que

$$\mathrm{SRAC}(\pi^s, C) = \min_{\pi \in \Pi} \left(\mathrm{SRAC}\left(\pi^s \to \pi, C\right) + d(\pi) \right).$$

Π *étant l'ensemble des* $2^{k'}$ *permutations rencontrées après le tri des* k *Intervalles communs lors de l'application de l'algorithme* SRAC, $k' \leq k$. *De même nous pouvons écrire :*

$$\mathrm{SRAC}(\pi^s, C') = \min_{\pi \in \Pi} \left(\mathrm{SRAC}\left(\pi^s \to \pi, C\right) + \mathrm{SRAC}\left(\pi, \{c_{k+1}\}\right) \right).$$

Or d'après l'hypothèse de récurrence, $\mathrm{SRAC}\left(\pi^s \to \pi, C\right) = d_C(\pi^s \to \pi)$.

Si c_{k+1} *ne chevauche aucun autre Intervalle commun, alors d'après le lemme 5.1 et le lemme 5.2,* $\mathrm{SRAC}\left(\pi, \{c_{k+1}\}\right) = d_{\{c_{k+1}\}}(\pi)$.

Si c_{k+1} *chevauche un autre Intervalle commun, alors il est inclus dans un multi-strip. Supposons que ce multi-strip soit composé des blocs* b_1, b_2 *et* b_3. *Nous supposerons que l'Intervalle commun ajouté est constitué des blocs* b_2 *et* b_3. *On remarque que comme tous les Intervalles communs de taille inférieure à* m *sont pris en compte dans* C, b_1 *et* b_2 *étaient des Intervalles communs de* C *et ces deux blocs formaient un mutli-strip. De même* b_3 *était un Intervalle commun de* C. *Le tri de L'intervalle commun ajouté ne casse aucun autre Intervalle commun et par le lemme 5.8 on obtient aussi* $\mathrm{SRAC}\left(\pi, \{c_{k+1}\}\right) = d_{\{c_{k+1}\}}(\pi)$.

Donc

$$\mathrm{SRAC}(\pi^s, C') = \min_{\pi \in \Pi} \left(d_C\left(\pi^s \to \pi\right) + d_{\{c_{k+1}\}}(\pi) \right).$$

Or d'après le théorème 5.3 et le lemme 5.2 et le lemme 5.8, un scénario conforme et minimal passe obligatoirement par une des permutations de Π*, et il est évident que le scénario inféré par* $\mathrm{SRAC}(\pi^s, C')$ *est conforme, donc* $\mathrm{SRAC}(\pi^s, C') = d_{C'}(\pi^s)$.

◀

Optimisation du problème complet

Trier un multi-strip en respectant les Intervalles communs demande un nombre d'inversions correspondant au nombre de blocs positifs ou négatifs. Or si dans un multi-strip, un strip provient d'un Intervalle commun neutre, il existe deux configurations pour ce multi-strip. Chacune des configurations vient d'un des deux appels récursifs nécessaires pour traiter les clusters neutres. Il y a une des configurations où cet Intervalle commun est un strip canonique croissant et une autre où il est un strip canonique décroissant. Une des deux configurations est inutile car suivant le multi-strip, au moment de le trier, il faudra rendre tous ses strips soit positifs, soit négatifs. Nous montrons maintenant que l'on peut éliminer la configuration inutile et par conséquent éliminer un des deux appels récursifs du cas des Intervalles communs neutres.

Nous adaptons maintenant les définitions de spins de permutations non signées, dues à [Hannenhalli and Pevzner, 1996], aux permutations partiellement signées. Une permutation signée π^s est un *spin* d'une permutation partiellement signée π^p, telle que

- si π^p_i est signé, alors $\pi^s_i = \pi^p_i$;
- si π^p_i est non signé, alors pour $1 \leq i \leq n$ soit $\pi^s_i = +\pi^p_i$ soit $\pi^s_i = -\pi^p_i$.

Pour une permutation partiellement signée arbitraire π^p de taille n avec k éléments non signés, nous définissons Π^{π^p} l'ensemble des 2^k spins de π^p.

Lemme 5.9 *Pour une permutation partiellement signée* π^p, $d(\pi^p) = \min_{\pi^s \in \Pi^{\pi^p}} d(\pi^s)$.

Preuve : Nous étendons les preuves de [Hannenhalli and Pevzner, 1996] pour les permutations non signées aux permutations signées. Pour tout spin π^s *de* π^p*, n'importe quel tri de* π^s *simule un tri de* π^p *et donc* $d(\pi^s) \geq d(\pi^p)$*. Soit* $\pi^p.\rho_1.\rho_2.\ldots.\rho_d = \mathcal{I}$ *un tri optimal de* π^p *avec* d *inversions. Considérons la permutation signée* $\pi^s = \mathcal{I}^s_+.\rho_d.\ldots.\rho_2.\rho_1$*, or* $\pi^s \in \Pi^{\pi^p}$ *et* $\pi^s.\rho_1.\rho_2.\ldots.\rho_d = \mathcal{I}^s_+$*,* $d(\pi^s) \leq d(\pi^p)$*, nous avons donc* $d(\pi^p) = \min_{\pi^s \in \Pi^{\pi^p}} d(\pi^s)$. ◀

Pareillement nous avons le lemme suivant :

Lemme 5.10 *Pour une permutation partiellement signée* π^p *et* C *l'ensemble de ses Intervalles communs,* $d_C(\pi^p) = \min_{\pi^s \in \Pi^{\pi^p}} d_C(\pi^s)$.

Nous adaptons maintenant l'opération COLLAPSE^{s}aux permutations partiellement signées. Soit l'opération COLLAPSE définie de la façon suivante :

$$\text{COLLAPSE} : \Sigma^p_n \rightarrow \Sigma^p_{\leq n}$$

Σ^p_n est l'ensemble de toutes les permutations partiellement signées de taille n et $\Sigma^p_{\leq n}$ est l'ensemble de toutes les permutations signées, de taille n ou moins, sans aucun strip. Les éléments non signés sont considérés comme possiblement positifs ou négatifs. Par exemple,

COLLAPSE(+3 +1 +2 +4) = +2 +1 +3 (la sous-permutation +1 +2 devient +1, la sous-permutation +3 devient +2 et la sous-permutation +4 devient +3). De même COLLAPSE(+3 ±1 +2 +4) = +2 +1 +3, COLLAPSE(+3 ±1 -2 +4) = +3 ±1 -2 +4 et COLLAPSE(-3 ±2 +1 +4) = -2 +1 +3.

Propriété 5.2 $d_C\big(\text{COLLAPSE}(\pi^p)\big) = d_C\left(\pi^p\right)$

Preuve : Comme [Hannenhalli and Pevzner, 1996] ont montré que les strips de taille au moins trois peuvent être comprimés et comme les strips de taille au moins deux forment un Intervalle commun, ce qui signifie que leurs éléments ne peuvent être séparés, nous avons que $d_C\big(\text{COLLAPSE}(\pi^p)\big) = d_C\left(\pi^p\right)$. ◄

Les Intervalles communs neutres doivent être triés vers l'identité positive ou négative. Autrement dit, il doivent être triés vers un strip canonique, croissant ou décroissant. Si on comprime ces strips, on obtient un seul élément dont le signe est soit positif, soit négatif. Pour prendre en compte ces deux cas, nous pouvons considérer cet élément comme non signé. Pour c un Intervalle commun neutre, nous nommerons $\pm c$ l'élément comprimé le représentant une fois trié. Son signe sera alors déterminé soit au final par le tri de la permutation résultante, soit par le tri d'un multi-strip l'englobant, soit par le tri d'un Intervalle commun l'incorporant.

Transformer les Intervalles communs en éléments non signées, donne lieu à une permutation partiellement signée. Cette permutation a les mêmes Intervalles communs et nous devons alors trier une permutation partiellement signée sans casser d'Intervalles communs. Les résultats démontrés précédemment dans ce chapitre restent corrects pour les permutations partiellement signées. Le lemme 5.2 s'adapte aux permutations partiellement signées car il dépend exclusivement de la remarque 5.3, laquelle est toujours vraie pour des permutations partiellement signées. Le théorème 5.3 s'adapte aussi aux permutations partiellement signées.

L'algorithme SRAC devient l'algorithme ASRAC en utilisant ces propriétés :

Théorème 5.7 *Pour une permutation partiellement signée* π^p *et* C *l'ensemble de tous ses Intervalles communs triés par taille croissante,* ASRAC *calcule* $d_C(\pi^s)$.

Preuve : Même raisonnement que pour la preuve de l'algorithme SRAC. ◄

Corollaire 5.1 *Pour* π^s *une permutation signée et* C *son ensemble d'Intervalles communs ne se chevauchant pas et trié par taille croissante de ses éléments, l'algorithme* ASRAC *calcule* $d_C(\pi^s)$.

L'algorithme ASRAC calcule $d_C(\pi^s)$ en $O(2^{k'}n + kn)$ en temps pour une permutation de taille n avec k Intervalles communs.k' est le nombre d'Intervalles communs n'étant ni positifs, ni négatifs, ni chevauchants. Dans le pire des cas k' est en $O(n)$.

La complexité de cet algorithme dépend de la complexité de l'algorithme permettant de trier une permutation partiellement signée. Dans le pire des cas, une permutation partiellement signée n'est pas du tout signée et le problème de trier une permutation non signée avec des inversions a été montré NP-dur [Caprara, 1997]. Mais si le nombre total d'éléments non

entrée : π^p et C trié par taille croissante de ses Intervalles communs
début
1. **Si** $C = \emptyset$ **Alors**
2. retourne $d(\pi^p)$
3. **Sinon**
4. $c \coloneqq C[0]$ $/* \; c \coloneqq \pi^s([x+1, y-1]) \; */$
5. **Si** c est un multi-strip croissant **Alors**
6. d :=nombre_de_blocs_négatifs (c)
7. retourne $d + \text{ASRAC}(\text{COLLAPSE}(\pi^p([1,x]).{+c}.\pi^p([y,n])), C \setminus \{c\})$
8. **Sinon Si** c est multi-strip décroissant **Alors**
9. d :=nombre_de_blocs_positifs (c)
10. retourne $d + \text{ASRAC}(\text{COLLAPSE}(\pi^p([1,x]).{-c}.\pi^p([y,n])), C \setminus \{c\})$
11. **Sinon Si** $d(c,{+c}) < d(c,{-c})$ **Alors**
12. retourne $d(c,{+c}) + \text{ASRAC}(\text{COLLAPSE}(\pi^p([1,x]).{+c}.\pi^p([y,n])))$
13. **Sinon Si** $d(c,{+c}) > d(c,{-c})$ **Alors**
14. retourne $d(c,{-c}) + \text{ASRAC}(\text{COLLAPSE}(\pi^p([1,x]).{-c}.\pi^p([y,n])))$
15. **Sinon**
16. retourne $d(c,{+c}) + \text{ASRAC}(\text{COLLAPSE}(\pi^p([1,x]).{\pm c}.\pi^p([y,n])), \text{COLLAPSE}(C \setminus \{c\}))$
17. **FinSi**
18. **FinSi**

Algorithme 9: L'algorithme ASRAC.

signés est en $O(\log n)$, alors le tri d'une permutation partiellement signée est un problème de complexité polynomial. Ce qui veut dire, que si le nombre total d'Intervalles communs neutres est en $O(\log n)$, alors l'algorithme proposé est polynomial. Dans le cas général, notre algorithme est de complexité $O(n2^{k_n})$ en temps avec k_n le nombre d'Intervalles communs neutres ne se chevauchants pas.

Remarques

On peut remarquer que la détection et le tri des multi-strips n'est pas nécessaire. En effet, le tri des Intervalles communs chevauchant est un peu particulier. Comme leurs sous Intervalles communs sont déjà triés, Ceux-ci forment soient des Intervalles communs positifs, soient des Intervalles communs négatifs. En appliquant la règle du tri des Intervalles communs positifs ou négatifs, les trier, revient à inverser ou non leur sous Intervalles communs ; ce qui correspond au tri des multi-strips. L'algorithme de tri ASRIC permet donc de calculer la distance conforme, même quand les Intervalles communs s'intersectent les uns les autres. Nous pouvons alors réecrire l'algorithme ASRAC en tenant compte de cette propriété : (voir algorithme 10).

entrée : π^p et C trié par taille croissante de ses Intervalles communs
début
1. **Si** $C = \emptyset$ **Alors**
2. retourne $d(\pi^p)$
3. **Sinon**
4. $c \coloneqq C[0]$ $/* \; c \coloneqq \pi^s([x+1, y-1]) \; */$
5. **Si** $d(c, {+}c) < d(c, {-}c)$ **Alors**
6. retourne $d(c, {+}c) + \text{ASRAC}(\text{COLLAPSE}(\pi^p([1,x]).{+}c.\pi^p([y,n])))$
7. **Sinon Si** $d(c, {+}c) > d(c, {-}c)$ **Alors**
8. retourne $d(c, {-}c) + \text{ASRAC}(\text{COLLAPSE}(\pi^p([1,x]).{-}c.\pi^p([y,n])))$
9. **Sinon**
10. retourne $d(c, {+}c) + \text{ASRAC}(\text{COLLAPSE}(\pi^p([1,x]).{\pm}c.\pi^p([y,n])), \text{COLLAPSE}(C \setminus \{c\}))$
11. **FinSi**
12. **FinSi**

Algorithme 10: L'algorithme ASRAC modifié.

5.3 Classe de complexité du problème

Nous discutons maintenant de la complexité du MCSS. Nous avons en effet proposé un algorithme de complexité exponentielle pour résoudre ce problème. Cette complexité est dans le pire des cas et on peut s'attendre en pratique à observer une complexité polynomiale en temps. Pour déterminer si nous pouvions trouver un algorithme polynomial dans le pire des cas, nous étudions la complexité du problème.

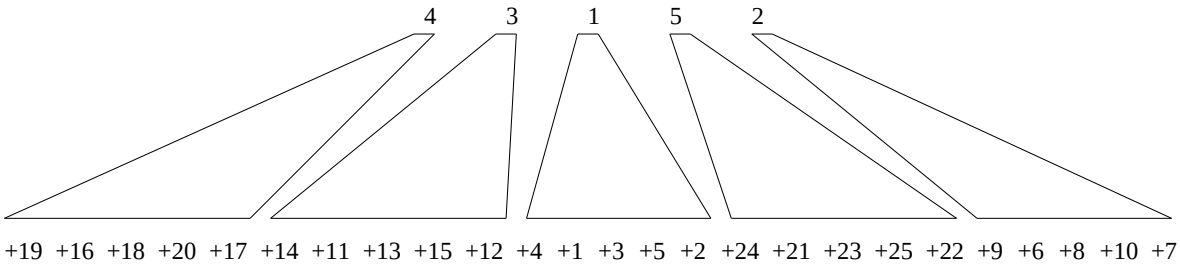
Nous étudions un cas particulier du MCSS, où les Intervalles communs ne s'intersectent pas les uns les autres. Ce problème est encore plus simplifié que celui résolu par l'algorithme SRIC. Non seulement les Intervalles communs ne se chevauchent pas mais ils ne sont pas inclus les uns dans les autres. Nous nommons ce sous problème SROC et nous montrons qu'il est NP-complet. Nous rappelons que le problème du calcul du nombre minimum d'inversions nécessaires pour trier une permutation signée est un problème dans P. D'ailleurs, on connaît un algorithme linéaire en temps [Bader et al., 2000].

Pour ce faire, nous comparons SROC avec le problème nommé USBR, qui consiste à trouver le nombre minimum d'inversions nécessaires pour trier une permutation non signée (*Unsigned Sorting By Reversals*). Ce problème a été montré NP-dur par Alberto Caprara [Caprara, 1997]. Le problème SROC quant à lui, consiste à trouver le nombre minimum d'inversions, conformes à un ensemble d'Intervalles communs sans intersection, pour trier une permutation signée.

Réduction de USBR dans SROC

Avant de présenter formellement la réduction des deux problèmes, nous montrons son fonctionnement par un exemple. Le principe de la réduction est de transformer une permutation non signée en permutation signée avec des contraintes liées à ses Intervalles communs.

Soit $\pi^u = 4 \quad 3 \quad 1 \quad 5 \quad 2$ une permutation non signée. On construit la permutation signée π^s à partir de π^u de la façon suivante :



On a remplacé chaque élément non signé de π^u par un Intervalle commun neutre. Un des ensembles d'Intervalles communs ne s'intersectant pas de la permutation signée est alors :

$$C = \{[1,5],[6,10],[11,15],[16,20],[21,25]\}$$

Nous nous intéressons maintenant au tri de cette permutation en respectant C. Comme les Intervalles communs ne s'intersectent pas, nous pouvons les trier indépendamment les uns des autres. Comme ce sont des clusters neutres, nous pouvons les trier vers l'identité positive et les remplacer par un élément non signé. Pour calculer la distance conforme il nous reste maintenant à trier la permutation résultante qui n'a plus de contrainte liées aux Intervalles communs. Cela revient à trier une permutation non signée :

$$\begin{aligned} d_C(\pi^s) = d(\pi^s([1,5,])) + d(\pi^s([6,10])) + d(\pi^s([11,15])) + d(\pi^s([16,20])) \\ + d(\pi^s([21,25])) + d(\pm 4 \quad \pm 3 \quad \pm 1 \quad \pm 5 \quad \pm 2) \\ d_C(\pi^s) = 5 * n + d(\pi^u) \\ d(\pi^u) = d_C(\pi^s) - 5 * n \end{aligned}$$

Lemme 5.11

$$USBR \leq_P SROC$$

Preuve : Soit $\pi^u = (\pi^u_1 \dots \pi^u_n)$, une permutation non signée et $c = (+4 \quad +1 \quad +3 \quad +5 \quad +2)$ une permutation signée de cinq éléments. Nous avons alors $d\big(c, (+1 \quad +2 \quad +3 \quad +4 \quad +5)\big) = d\big(c, (-5 \quad -4 \quad -3 \quad -2 \quad -1)\big) = 5$. Soit $\pi^s = (\pi^s{}_1 \dots \pi^s{}_{5n})$ une permutation signée obtenue en remplaçant chaque élément de π^u, noté π^u_i pour $1 \leq i \leq n$, par la sous-permutation neutre $\big(+(5(\pi^u{}_i - 1) + 4) \quad +(5(\pi^u{}_i - 1) + 1) \quad +(5(\pi^u{}_i - 1) + 3) \quad +(5(\pi^u{}_i - 1) + 5) \quad +(5(\pi^u{}_i - 1) + 2) \quad \big)$. Un ensemble d'Intervalles communs, sans intersection, de π^s est $C = \{[1,5],[6,10],\dots,\}$. Nous utilisons maintenant l'algorithme ASRAC. Après n appels récursifs et $5n$ inversions effectués en temps polynomial, nous obtenons une permutation non signée représentée par l'ensemble de ses 2^n spins (chaque élément est soit positif, soit négatif). Ce qui signifie que pour π^u, π^s et C l'ensemble d'Intervalles communs construit nous avons : $d_C(\pi^s) = 5n + d(\pi^u)$. Donc pour toute permutation π^u, nous pouvons construire π une permutation signée de taille $5n$ avec C Intervalles communs sans intersection telle que $d(\pi^u) = d_C(\pi^s) - 5n$. Nous venons de montrer que le problème USBR peut être réduit en temps polynomial dans le problème SROC. ◄

Réduction de SROC dans USBR

De nouveau nous mimons la réduction de π^s vers π^u avec un exemple. Soit π^s une permutation signée avec deux Intervalles communs sans intersection. π^s = -2 +10 +4 -3 -1 +8 +5 +7 +9 +6 et $C = \{[3,4],[6,10]\}$. Nous commençons par trier de manière optimale π^s.

Or $d(\pi^s([3,4]),\mathcal{I}^s_+) > d(\pi^s([3,4]),\mathcal{I}^s_-) = 1$, donc on peut transformer π^s en triant préalablement un premier Intervalle commun (lesquels sont indépendants l'un de l'autre).

$$\pi^s.\rho_1...\rho_a = \text{-2} \quad +10 \quad \text{-4} \quad \text{-3} \quad \text{-1} \quad +8 \quad +5 \quad +7 \quad +9 \quad +6 = \pi^{s1}$$

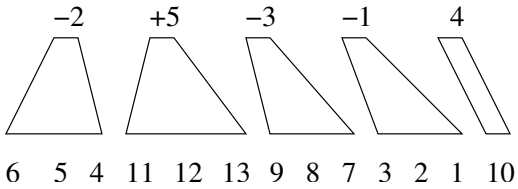
Nous pouvons maintenant comprimer π^{s1} sans modifier sa distance par inversions (conforme ou non). -4 -3 devient -3 et tous les éléments supérieurs à 4 sont décrémentés :

$$\text{COLLAPSE}(\pi^{s1}) = \text{-2} \quad +9 \quad \text{-3} \quad \text{-1} \quad +7 \quad +4 \quad +6 \quad +8 \quad +5 = \pi^{s2}$$

Or $d(\pi^s([6,10]),\mathcal{I}^s_+) = d(\pi^s([6,10]),\mathcal{I}^s_-) = 5$, donc en transformant π^{s2} en remplaçant cet Intervalle commun, une fois trié, par un élément non signé on obtient :

$$\pi^{s3} = \text{-2} \quad +5 \quad \text{-3} \quad \text{-1} \quad \pm 4$$

Et nous obtenons une permutation partiellement signée. Nous allons maintenant la transformer en permutation non signée. Pour cela nous remplaçons les éléments signés par des strips canoniques non signés. D'après [Hannenhalli and Pevzner, 1996], en effectuant cette modification, nous ne modifions pas la différence des distances par inversions entre permutations signées et non signées.



Nous notons π^u la permutation non signée obtenue ($d(\pi^{s3}) = d(\pi^u)$).

On obtient finalement :

$$d_C(\pi^s) = d(\pi^u) + d(+4 \quad \text{-3}, \text{-4} \quad \text{-3}) + d(+7 \quad +4 \quad +6 \quad +8 \quad +5, +4 \quad +5 \quad +6 \quad +7 \quad +8)$$

C'est-à-dire,

$$d_C(\pi^s) = 4 + 1 + 5 = 10.$$

Lemme 5.12

$$SROC \leq_P USBR$$

Preuve : Soit $\pi^s = (\pi^s_1 \dots \pi^s_n)$ *une permutation signée de* n *éléments et* C *l'ensemble de ses Intervalles communs sans intersection* $C = \{c_i | 1 \leq i \leq k\}$.

Soit une deuxième permutation signée $\pi^{s\prime}$ *une copie de* π^s. $\pi^{s\prime}$ *et* π^s *ont les même Intervalles communs. Nous allons d'abord transformer* $\pi^{s\prime}$ *en permutation partiellement signée sans Intervalle commun. Tous les Intervalles communs non neutres sont triés et deviennent un strip canonique croissant s'ils sont positifs et décroissant s'ils sont négatifs. Tous les Intervalles communs neutres sont quant à eux, remplacés par un élément non signé dans* $\pi^{s\prime}$. *Nous avons alors grâce à l'algorithme ASRAC que :*

$$d_C(\pi^s) = \sum_{\{c|c \in C \text{et } c \text{ positif}\}} d(c,+c) + \sum_{\{c|c \in C \text{et } c \text{ négatif}\}} d(c,\text{-}c) + \sum_{\{c|c \in C \text{et } c \text{ neutre}\}} d(c,+c) + d(\pi^{s\prime})$$

La permutation résultante $\pi^{s\prime}$ *est une permutation partiellement signée. Nous allons maintenant transformer* $\pi^{s\prime}$ *en permutation non signée. Tout élément positif de* $\pi^{s\prime}$ *est remplacé par un Intervalle commun non neutre. Si l'élément* $\pi^{s\prime}_j$ *est positif, alors on le remplace par la sous-permutation non signée* $\big(\pi_j \quad (\pi_j + 1) \quad (\pi_j + 2)\big)$. *De même on remplace tout élément* $\pi^{s\prime}_j$ *négatif de* $\pi^{s\prime}$ *par la sous permutation neutre non signée* $\big((\pi_j + 2) \quad (\pi_j + 1) \quad \pi_j\big)$. *Nous obtenons une nouvelle permutation* $\pi^{s\prime}$ *que nous nommons* π^u *et qui est une permutation non signée de* m *éléments. Nous avons maintenant d'après [Hannenhalli and Pevzner, 1996] que* $d(\pi^{s\prime}) = d(\pi^u)$. *Et nous avons finalement :*

$$d_C(\pi^s) = \sum_{\{c|c \in C \text{et } c \text{ positif}\}} d(c,+c) + \sum_{\{c|c \in C \text{et } c \text{ négatif}\}} d(c,\text{-}c) + \sum_{\{c|c \in C \text{et } c \text{ neutre}\}} d(c,+c) + d(\pi^u)$$

Le partitionnement de C *se fait en temps polynomial ainsi que la création de la permutation* π^u. *Nous avons montré que SROC se réduisait en USBR en temps polynomial.* ◀

Théorème 5.8 *Le problème SROC est NP-complet.*

Preuve : [Caprara, 1997] a démontré que USBR était NP-dur, et nous avons démontré que $SROC \leq_P SBR$ *et que* $SBR \leq_P SROC$, *donc SROC est NP-complet.* ◀

Corollaire 5.2 *Le problème MCSS est NP-complet.*

Preuve : SROC est NP-complet, or SROC est un sous problème du MCSS, donc MCSS est aussi NP-complet. ◀

5.4 Calcul du diamètre

Nous nous intéressons dans cette section au calcul du diamètre du MCSS. Nous voulons connaître le nombre maximum d'inversions conformes nécessaires pour résoudre le MCSS étant donné un ensemble d'Intervalles communs et une permutation signée.

Nous savons que le tri d'une permutation signée linéaire ou circulaire de n gènes nécessite au plus du nombre de points de cassure comme inversions. Nous notons $\hat{d}(\pi^s)$ le diamètre du problème du nombre minimum d'inversions pour trier une permutation signée (SBR).

Théorème 5.9 *[Meidanis et al., 2000]*

$$\hat{d}(\pi^s) = \begin{cases} n-1 & \textit{si } \pi^s \textit{ est circulaire et } n = 1, 2 \textit{ ou } 4 \\ n & \textit{si } \pi^s \textit{ est circulaire et } n \neq 1, 2, 4 \textit{ ou } \pi^s \textit{ linéaire et } n = 3 \\ n+1 & \textit{si } \pi^s \textit{ est linéaire et } n \neq 3 \end{cases}$$

Soit $b(\pi^s)$ une application retournant le nombre de points de cassure de π^s. On obtient alors

$$\hat{d}(\pi^s) \leq b(\pi^s)$$

Nous cherchons à caractériser $\hat{d_C}(\pi^s)$, le diamètre du problème du MCSS (le nombre minimum d'inversions pour trier une permutation signée en respectant C l'ensemble des Intervalles communs de celle-ci).

Nous avons divisé le calcul du diamètre en trois étapes. La première étape considère les Intervalles communs sans intersection. La deuxième étape suppose que les Intervalles communs sont sans chevauchement. Et la dernière étape, considère le problème complet.

Avec des Intervalles communs sans intersection

Soit une permutation signée π^s avec p Intervalles communs nommés c_i pour $1 \leq i \leq p$. Nous supposons dans un premier temps que les c_i ne s'intersectent pas, c'est-à-dire qu'ils ne sont pas inclus, ni ne sont imbriqués les uns les autres. Nous avons vu que nous pouvons résoudre ce problème en supprimant d'abord les points de cassure des Intervalles communs puis en triant la permutation résultante.

Supprimer tous les points de cassure d'un Intervalle commun c_i est équivalent à le trier soit positivement, soit négativement. Nous avons considéré que les Intervalles communs étaient des sous-permutations linéaires. On peut ainsi borner le nombre d'inversions nécessaires pour éliminer tous les points de cassure d'un Intervalle commun d'après le théorème 5.9 par $|c_i|+1$. Or éliminer les points de cassure d'un Intervalle commun ne revient pas exactement à trier une sous-permutation linéaire car l'Intervalle commun peut être trié soit vers l'identité positive, soit vers l'identité négative. Nous montrons que la borne $|c_i|+1$ n'est pas très précise et que $|c_i|$ est une meilleure borne.

Lemme 5.13 *Pour un Intervalle commun c d'une permutation signée π^s :* $\min\left(d(c, +c), d(c, -c)\right) \leq |c|$

Preuve : Nous démontrons par l'absurde que la borne $n+1$ ne peut pas être atteinte et que par conséquent n est une meilleure borne. Soit un Intervalle commun c avec $|c| = n$. On suppose que $d(c, +c) = n+1$ et que par conséquent il n'existe pas de scénario plus long. Pour un scénario de taille $n+1$, on peut distinguer deux cas de figure : soit le scénario passe par la permutation -c, soit il ne passe pas par cette permutation. Dans le premier cas, on a $d(c, -c) < n+1$. Sinon, on peut appliquer une inversion supplémentaire et obtenir $d(c, -c) = n+2$, ce qui est impossible par hypothèse. Par conséquent nous savons que si $d(c, +c) = n+1$, alors $d(c, -c) \leq n$. Et symétriquement nous savons que si $d(c, -c) = n+1$, alors $d(c, +c) \leq n$. Donc $\min\left(d(c, +c), d(c, -c)\right) \leq |c|$. ◀

Nous pouvons éliminer tous les points de cassure des Intervalles communs avec au plus $\sum_{i=1}^{i\leq p} |c_i|$ inversions. Ce nombre d'inversions correspond au nombre de points de cassure éliminés. La permutation résultante, $\pi^{s\prime}$, est alors triée avec au plus $b(\pi^{s\prime})$ inversions. Ce nombre de points de cassure dans $\pi^{s\prime}$ dépend du nombre initial de points de cassure de π^s et des points de cassure déjà éliminés. Nous avons donc $b(\pi^{s\prime}) \leq b(\pi^s) - \sum_{i=1}^{i\leq p}(|c_i| - 1)$.

Lemme 5.14 *Soit une permutation signée π^s avec p Intervalles communs nommés c_i pour $1 \leq i \leq p$. S'il n'y a pas d'intersection de ces Intervalles communs, alors $d_C(\pi^s) \leq b(\pi^s) + p$.*

Preuve : Nous avons besoin d'au plus $\sum_{i=1}^{i\leq p} |c_i|$ inversions pour éliminer les points de cassure des Intervalles communs (cf. lemme 5.13). La permutation résultante $\pi^{s\prime}$ peut être triée avec au plus $b(\pi^{s\prime})$ inversions, $b(\pi^{s\prime}) \leq b(\pi^s) - \sum_{i=1}^{i\leq p}(|c_i| - 1)$. C'est pourquoi $d_C(\pi^s) \leq \sum_{i=1}^{i\leq p} |c_i| + b(\pi^s) - \sum_{i=1}^{i\leq p}(|c_i| - 1) = b(\pi^s) + p$. ◀

Nous montrons maintenant à l'aide d'un exemple que cette borne supérieure peut être atteinte. Soit la permutation signée linéaire π^s = -2 -3 +1, qui possède un Intervalle commun (en omettant la permutation elle même) $[1, 2]$ correspondant aux éléments $\{2, 3\}$ de π^s. Un MCSS est le suivant :

$$\begin{aligned}
\pi^s &= \underline{\text{-2 -3 +1}} \\
&\rightarrow \underline{\text{-1}} \text{ +3 +2} \\
&\rightarrow \text{+1 } \underline{\text{+3}} \text{ +2} \\
&\rightarrow \text{+1 -3 } \underline{\text{+2}} \\
&\rightarrow \text{+1 } \underline{\text{-3 -2}} \\
&\rightarrow \text{+1 +2 +3}
\end{aligned}$$

La borne donnée par le lemme 5.14 est la suivante : $d_C(\pi^s) \leq b(\pi^s) + p = 4 + 1 = 5$; ce qui est exactement la distance minimum calculée précédemment.

Remarque 5.4 *L'exemple précédent nous permet par ailleurs de montrer que $d_C(\pi^s)$ n'est pas toujours inférieure ou égal à $d(\pi^s) + p$. Par contre, $d_C(\pi^s)$ est toujours inférieure ou égale à $b(\pi^s) + p$.*

Pour des Intervalles communs sans chevauchement

Nous nous intéressons maintenant à un problème légèrement plus compliqué où nous autorisons les Intervalles communs à être inclus les uns dans les autres. Nous n'autorisons toujours pas les Intervalles communs à s'intersecter. Nous montrons que la borne calculée précédemment est toujours valide.

Lemme 5.15 *Soit une permutation signée π^s avec p Intervalles communs nommés c_i pour $1 \leq i \leq p$. Les intervalles communs ne se chevauchent pas les uns les autres.*

$$\hat{d_C}(\pi^s) \leq b(\pi^s) + p$$

Preuve : Nous le montrons par récurrence sur les niveaux d'inclusions. Soit une permutation π^s avec p Intervalles communs et k niveaux d'inclusions de ceux-ci. Notre hypothèse de récurrence est que pour $k = m-1$, $\hat{d_C}(\pi^s) \leq b(\pi^s) + p$.

- *pour $k = 1$, le lemme 5.14 nous donne $\hat{d_C}(\pi^s) \leq b(\pi^s) + p$;*
- *pour $k = m-1$, c'est notre hypothèse de récurrence ;*
- *pour $k = m$, nous trions d'abord les m_0 Intervalles communs sans aucun autre Intervalle commun inclus $(c_1, ..., c_{m_0})$. Nous pouvons le faire avec au plus $\sum_{i=0}^{i \leq m_0} |c_i|$ inversions (cf. lemme 5.13).*
  La permutation résultante $\pi^{s\prime}$ possède $k = m-1$ niveaux d'inclusions et en appliquant l'hypothèse de récurrence, nous obtenons $\hat{d_C}(\pi^{s\prime}) \leq b(\pi^{s\prime}) + p - m_0$.
  Le nombre total d'inversions est alors : $\hat{d_C}(\pi^s) \leq \sum_{i=0}^{i \leq m_0} |c_i| + b(\pi^{s\prime}) + p - m_0$ avec $b(\pi^{s\prime}) = b(\pi^s) - \sum_{i=0}^{i \leq m_0} (|c_i| - 1)$. Ce qui donne

$$\hat{d_C}(\pi^s) \leq \sum_{i=0}^{i \leq m_0} |c_i| + b(\pi^s) - \sum_{i=0}^{i \leq m_0} |c_i| + m_0 + p - m_0 = b(\pi^s) + p$$

◄

Nous pouvons utiliser le même exemple que précédemment, où il n'y avait aucun niveau d'inclusion des Intervalles communs, pour montrer que la borne peut être atteinte.

Avec tous les Intervalles communs

Nous nous intéressons dans cette section au problème complet sans restriction sur les Intervalles communs. Nous caractérisons le diamètre du problème général du MCSS.

Lemme 5.16 *Pour π^s une permutation signée et C l'ensemble de tous ses Intervalles communs. Soit deux Intervalles communs $[i,j]$ et $[k,l]$ de C tel qu'ils se chevauchent l'un dans l'autre. Soit s un MCSS respectant C. Prenons maintenant $C' = C \setminus \{[k,l],[i,j]\}$, alors s est un MCSS respectant C'.*

Preuve : Prenons deux Intervalles communs $[i,j]$ et $[k,l]$ d'une permutation signée π^s. Sans perdre en généralité, on suppose que $i < k$ et que $k < j$. Si $[i,j]$ et $[k,l]$ se chevauchent, alors il existe six Intervalles communs relatifs à $[i,j]$ et $[k,l]$: $[i,k-1],[k,j],[j+1,l],[i,j],[k,l]$ et $[i,l]$. Ces Intervalles communs sont représentés figure 5.3. Supposons sans perdre en généralité que dans un MCSS, l'ordre de tri de ces Intervalles communs est le suivant :$[i,k-1],[k,j],[j+1,l],[i,j],[k,l]$ et finalement $[i,l]$. Soit $\pi^{s\prime}$ la permutation résultante après que les trois premiers Intervalles communs soient triés. Une fois les éléments de $\pi^{s\prime}$ comprimés, l'intervalle $[i,l]$ est soit du type $\pm 1 \quad \pm 2 \quad \pm 3$, soit du type $\pm 3 \quad \pm 2 \quad \pm 1$. Il n'est alors plus nécessaire de prendre en compte les sous Intervalles communs chevauchants car $[i,l]$ forme un multi-strip. Il sera trié de manière conforme à tous les Intervalles communs. ◄

Le théorème suivant en découle :

Théorème 5.10 *Pour π^s une permutation signée et C son ensemble d'Intervalles communs. Il existe $C' = \{[i,j] | [i,j] \in C \text{ et } \nexists [k,l] \in C \text{ chevauchant } [i,j]\}$, tel que $d_{C'}(\pi^s) = d_C(\pi^s)$.*

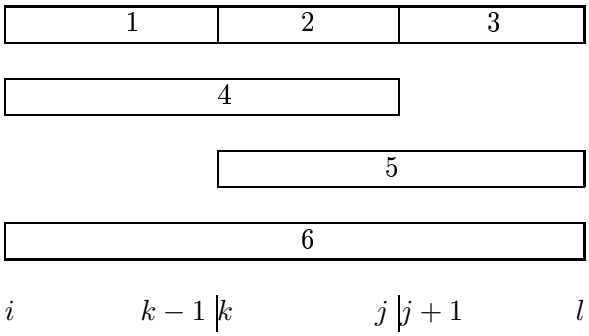


Figure 5.3: Les Intervalles communs $[i, j]$ et $[k, l]$ se chevauchent. Cela donne lieu à six Intervalles communs : $[i, k-1], [k, j], [j+1, l], [i, j], [k, l]$ et $[i, l]$. Les deux Intervalles communs se chevauchant peuvent être éliminés des contraintes évolutives sans modifier le MCSS.

Lemme 5.17 *Soit une permutation signée π^s et son ensemble d'Intervalles communs $C = \{c_i | 1 \leq i \leq p\}$. Soit p' le nombre maximal d'intervalles communs de C sans que ces p' Intervalles ne s'intersectent. Le diamètre du MCSS, $\hat{d_C}$, est*

$$\hat{d_C}(\pi^s) \leq b(\pi^s) + p'$$

Preuve : D'après le théorème 5.10, on peut retirer un ensemble d'Intervalles communs chevauchant sans modifier le MCSS. On arrive ainsi à un ensemble d'Intervalles communs ne se chevauchant pas. Il existe plusieurs sous-ensembles C' de C tels que leurs Intervalles communs ne se chevauchant pas. Prenons C' de taille p', tel que C' soit de taille maximum. Donc d'après le lemme 5.15, le diamètre du MCSS, $\hat{d_C}$, est tel que $\hat{d_C}(\pi^s) \leq b(\pi^s) + p'$. ◀

Théorème 5.11 *Pour une permutation signée π^s de taille n et son ensemble d'Intervalles commun C, le diamètre du MCSS, $\hat{d_C}$, est tel que*

$$\hat{d_C}(\pi^s) \leq \begin{cases} 2n-3 & \text{si } \pi^s \text{ est circulaire} \\ 2n-1 & \text{si } \pi^s \text{ est linéaire} \end{cases}$$

Preuve : D'après le lemme 5.17, $\hat{d_C}(\pi^s) \leq b(\pi^s)+p'$, avec p' le nombre maximal d'Intervalles communs de C sans qu'ils ne s'imbriquent. Or le nombre maximum d'Intervalles communs ne se chevauchant pas dans une permutation linéaire de taille n est $n-1$. A ces $n-1$ Intervalles communs, nous pouvons éliminer l'Intervalle commun correspondant à la permutation elle même. Donc $\hat{d_C}(\pi^s) \leq 2n-1$ si π^s est linéaire. Dans le cas des permutations circulaires, le nombre maximum d'Intervalles communs ne se chevauchant pas est de $n-3$. Donc $\hat{d_C}(\pi^s) \leq 2n-3$ si π^s est linéaire.

◀

Théorème 5.12 *Pour une permutation signée linéaire π^s de taille n quelconque et son ensemble d'Intervalles communs C, le diamètre du MCSS, $\hat{d_C}$, est tel que*

$$\hat{d_C}(\pi^s) = 2n-1$$

Preuve : Nous savons que $\hat{d_C}(\pi^s) \leq 2n-1$ *pour* π^s *une permutation linéaire de taille* n. *Nous montrons que pour toute permutation de taille* $n = 2k$ *et* $n = 2k+1$ *pour* $k \in \mathbb{N}$ *cette borne peut être atteinte.*

Soit π^s *une permutation de taille* $n = 2k$ *pour* $k \in \mathbb{N}$ *de la forme*

$$\pi^s = +n \quad +(n-2) \quad \dots \quad 2 \quad 1 \quad -3 \quad -5 \quad \dots \quad -(n-1)$$

Les $n-1$ *Intervalles communs sont les suivants :* $\{1,2\}$,$\{1,2,3\}$,...,$\{1,...,n\}$. *On commence par trier le premier Intervalle commun qui est négatif. On le trie donc vers -2 -1 avec deux inversions. L'Intervalle commun suivant est -2 -1 -3 qui lui est positif. On le trie de même en deux inversions pour obtenir* $+1$ $+2$ $+3$. *L'Intervalle commun suivant,* $+4$ $+1$ $+2$ $+3$, *est identique au premier si on comprime le strip* $+1$ $+2$ $+3$. *L'Intervalle suivant sera par le même procédé identique au deuxième Intervalle commun traité. En réitérant le procédé, deux inversions par Intervalle commun rencontré sont nécessaires. Après avoir traité les* $n-1$ *Intervalles communs on obtient l'identité négative qui nécessite une inversion supplémentaire pour être transformée en permutation identité positive. Le nombre total d'inversions est donc de* $2(n-1)+1$ *ce qui est équivalent à la borne du diamètre.*

Soit π^s *une permutation de taille* $n = 2k+1$ *pour* $k \in \mathbb{N}$ *de la forme*

$$\pi^s = -2 \quad -4 \quad \dots \quad -(n-1) \quad -n \quad (n-2) \quad (n-4) \quad \dots \quad 1$$

Les $n-1$ *Intervalles communs sont les suivants :* $\{n-1,n\}$,$\{n-2,n-1,n\}$,...,$\{1,...,n\}$. *En suivant le même procédé que pour la permutation précédente on obtient une distance de* $2(n-1)+1$ *laquelle est équivalente à la borne du diamètre.* ◄

Théorème 5.13 *Pour une permutation signée circulaire* π^s *de taille* n *quelconque et son ensemble d'Intervalles commun* C, *le diamètre du MCSS,* $\hat{d_C}$, *est tel que*

$$\hat{d_C}(\pi^s) = 2n-3$$

Preuve : Nous savons que $\hat{d_C}(\pi^s) \leq 2n-3$ *pour* π^s *une permutation circulaire de taille* n. *Nous montrons qu'il existe une permutation de taille* $n = 2k$ *et* $n = 2k+1$ *pour* $k \in \mathbb{N}$ *tel que* $d_C = 2n-3$.

Soit π^s *une permutation de taille* $n = 2k$ *pour* $k \in \mathbb{N}$ *de la forme*

$$\pi^s = -2 \quad -4 \quad \dots \quad -(n-2) \quad -(n-1) \quad (n-3) \quad (n-5) \quad \dots \quad 3 \quad 1 \quad n$$

Cette permutation à $n-1$ *Intervalles communs :* $C = \{[n-2,n-1],[n-3,n-1],\dots,[1,n-1],[1,n]\}$. *On commence par trier le premier Intervalle commun qui est négatif. On le trie donc vers -*$(n-2)$ *-*$(n-1)$ *avec deux inversions. L'Intervalle commun suivant est -*$(n-2)$ *-*$(n-1)$ *-*$(n-3)$ *qui lui est positif. On le trie de même en deux inversions pour obtenir* $+(n-1)$ $+(n-2)$ $+(n-3)$. *L'Intervalle commun suivant,* $+(n-4)$ $+(n-1)$ $+(n-2)$ $+(n-3)$,

est identique au premier si on comprime le strip $_{+}(n-1)$ $_{+}(n-2)$ $_{+}(n-3)$. *L'Intervalle suivant cette étape sera par le même procédé identique au deuxième Intervalle commun traité. En continuant ce procédé, qui est optimal, on a besoin de deux inversions par Intervalle commun. Après avoir traité les* $n-2$ *premiers Intervalles communs avec* $2n-4$ *inversions, on obtient la permutation suivante :* -$(n-1)$ -$(n-2)$ ... -1 $_{+}n$. *Laquelle avec une inversion supplémentaire est triée. On obtient finalement une distance de* $2n-3$ *inversions pour cette permutation circulaire.*

Soit π^s *une permutation de taille* $n+1=2k+1$ *pour* $k \in I\!N$ *de la forme*

$$\pi^s = {}_{+}(n-1) \quad \dots \quad {}_{+}4 \quad {}_{+}2 \quad {}_{+}1 \quad -3 \quad \dots \quad -(n-2) \quad n$$

En suivant le même procédé que pour la permutation précédente on obtient la permutation -$(n-1)$ ... -1 $_{+}n$ *après* $2\times(n-3)$ *inversions. Il reste à effectuer 3 inversions pour trier cette permutation, ce qui donne une distance de* $2n-3$. *Laquelle est équivalente à la borne du diamètre pour une permutation de taille* n. ◄

5.5 Résumé et conclusions

Pour mesurer l'évolution entre des espèces, on peut analyser l'ordre des gènes de leurs génomes. Une des méthodes la plus utilisée est le calcul d'un scénario parcimonieux par inversions de segments de gènes. Une inversion retourne un segment de gènes et change leur signe.

Alors que dans le chapitre précédent nous avons montré les lacunes de cette méthode, nous avons étudié dans celui-ci une des façons d'améliorer les scénarios évolutifs inférés. Après avoir étudié les scénarios parcimonieux par inversions, nous avons mis en avant leur manque de vraisemblance. Nous sommes sortis des chantiers battus en proposant une nouvelle distance. Nous n'avons pas rajouté d'événements évolutifs mais nous avons contraint les distances calculées. Nous avons montré que les scénarios parcimonieux par inversions ne respectent pas les Intervalles communs, des groupes de gènes toujours conservés dans le même voisinage. Ces Intervalles communs sont cassés par les scénarios parcimonieux pour être reconstruits par la suite. Un tel scénario ne nous semble pas vraisemblable.

Pour palier à ce non sens, nous avons posé un nouveau problème, le MCSS (*Minimal Compliant Sorting Scenario*). Pour deux permutations π et σ, σ étant la permutation identité, avec C un ensemble d'Intervalles communs de π et σ, nous avons calculé $d_C(\pi)$, le nombre minimum d'inversions conformes à C, nécessaires pour trier π. d_C est la distance conforme à C.

Non seulement les distances évolutives calculées sont plus vraisemblables, mais c'est la première fois que des travaux sur les inversions proposent d'augmenter la distance parcimonieuse. Les travaux précédents, liés aux problèmes des scénarios parcimonieux, consistaient à choisir un scénario parmi tous les scénarios parcimonieux. Nous avons montré que parfois, tous les scénarios parcimonieux ne sont pas conformes aux Intervalles communs et donc qu'aucun d'eux ne reflète l'évolution réelle entre les espèces.

Nous avons proposé des algorithmes exacts pour résoudre le problème du calcul de la distance conforme. Le dernier est de complexité $O(2^{k'}n+kn)$ en temps pour une permutation

de taille n avec k Intervalles communs.k' est le nombre d'Intervalles communs n'étant ni positifs, ni négatifs, ni imbriqués. Dans le pire des cas k' est en $O(n)$. Mais on peut s'attendre à ce qu'il soit bien inférieur.

Nous avons aussi montré que ce nouveau problème est NP-complet. Et pour clôturer la présentation du problème du MCSS, nous avons donné son diamètre. Le tableau suivant résume la présentation du MCSS que nous avons fait :

Théorème 5.1 *Pour toute permutation signée π^s et C un ensemble d'Intervalles communs à π^s et à la permutation identité, $d_C(\pi^s) \geq d(\pi^s)$.*

Théorème 5.2 *Pour toute permutation non signée π^u et C un ensemble d'Intervalles communs à π^u et à la permutation identité, $d_C(\pi^u) \geq d(\pi^u)$.*

Théorème 5.6 *Pour une permutation signée π^s et tous ses Intervalles communs C,* SRAC *calcule $d_C(\pi^s)$.*

Théorème 5.7 *Pour une permutation signée π^s et tous ses Intervalles communs C,* ASRAC *calcule $d_C(\pi^s)$.*

Théorème 5.2 *Le problème MCSS est NP-complet.*

Théorème 5.12 *Pour une permutation signée linéaire π^s de taille n quelconque et son ensemble d'Intervalles commun C, le diamètre du MCSS, $\hat{d_C}$, est tel que*

$$\hat{d_C}(\pi^s) = 2n - 1$$

Théorème 5.13 *Pour une permutation signée circulaire π^s de taille n quelconque et son ensemble d'Intervalles commun C, le diamètre du MCSS, $\hat{d_C}$, est tel que*

$$\hat{d_C}(\pi^s) = 2n - 3$$

chapitre 6

Expérimentations

Les travaux antérieurs sur le calcul de scénarios parcimonieux par inversions soit permettent simplement de calculer la distance parcimonieuse ou d'inférer un scénario parcimonieux, soit parmi tous les scénarios parcimonieux, ils cherchent un scénario spécifique. Nous avons exposé dans le chapitre précédent une lacune commune à tous les scénarios parcimonieux : ils ne respectent pas toujours les Intervalles communs. Il ne s'agit donc plus de choisir un scénario parmi les scénarios parcimonieux mais d'en trouver un plus long qui respecte les Intervalles communs. Nous avons montré dans le chapitre précédent comment calculer une distance évolutive respectant ces contraintes. Nous avons appelé cette distance, la distance conforme.

Nous présentons dans ce chapitre des expérimentations sur le calcul de la distance parcimonieuse et sur la distance conforme. Pour mieux mettre en avant les différences de ces deux distances, nous les comparons sur des données générées. Nous proposons différents modèles d'évolution permettant de générer des permutations avec une certaine distance évolutive par rapport à la permutation identité.

Pour chacun de ces modèles d'évolution, nous évaluons la distance parcimonieuse et la distance conforme. Pour la distance conforme nous l'évaluons dans le cas où nous connaissons les Intervalles communs réels, s'il y en a, ainsi que dans le cas où nous devons préalablement les détecter.

Dans un premier temps, nous montrons l'adéquation entre notre méthode, la distance conforme, et la mesure d'une distance évolutive basée sur le nombre d'inversions. Cette étude utilise des modèles d'évolution basiques. Dans un deuxième temps, nous proposons des modèles d'évolution basés sur les observations faites sur des génomes réels. Puis, nous utilisons la distance parcimonieuse et la distance conforme pour mesurer l'évolution générée. Nous montrons que pour ces modèles d'évolution, la distance conforme est un meilleur estimateur que la distance parcimonieuse simple. Finalement, nous comparons les temps d'exécutions des algorithmes de la distance conforme et de la distance parcimonieuse simple.

6.1 Évaluation sur des données générées

Pour évaluer le bien fondé des contraintes apportées par les Intervalles communs, nous étudions dans cette section, dans quelles mesures ils modifient la distance par inversions. On fera référence au calcul de la distance par inversions dite simple par l'algorithme HP. L'algorithme et son implémentation ont été proposées par Bader *et al* [Bader et al., 2000, Bader et al., 2001]. Nous l'avons nommé HP, pour faire référence à la première version de cet algorithme due à Hannenhalli et Pevzner [Hannenhalli and Pevzner, 1995]. Pour le calcul de la distance conforme, on utilisera l'algorithme SRAC présenté dans le chapitre précédent.

Adéquation entre le modèle et l'algorithmique

Nous faisons l'hypothèse que l'évolution de l'ordre des gènes respecte des Intervalles communs. Nous voulons vérifier que ce modèle d'évolution est bien pris en compte par l'algorithmique mise en place.

Pour cela nous générons aléatoirement des permutations suivant le procédé suivant : pour une distance évolutive d, un nombre de gènes n donné et un nombre d'Intervalles communs c, nous construisons la permutation identité positive de taille n, nous choisissons alors c Intervalles représentant des Intervalles communs de manière aléatoire et nous appliquons d inversions conformes à cette permutation. Un Intervalle commun est choisi en sélectionnant de manière équiprobable deux points de cassure parmi les $n+1$ possibles. Les Intervalles communs définis peuvent s'intersecter et être inclus les uns dans les autres. Une inversion est choisie en sélectionnant de manière équiprobable deux points de cassure parmi les $n+1$ possibles. Si cette inversion n'est pas conforme aux Intervalles communs, alors on en choisit une autre. Une même inversion peut-être sélectionnée deux fois et ainsi s'annuler. Il faut noter qu'une même inversion appliquée deux fois ne s'annihile pas toujours.

Nous avons fixé n, la taille des permutations, à 100 gènes. Le nombre d'Intervalles communs prédéfinis, c, est de 5, 10 et 15. Pour les différentes valeurs de nombre d'inversions, d, nous avons répété l'expérience 5000 fois.

Nous calculons maintenant le nombre minimum d'inversions conformes à C nécessaires pour trier les permutations générées. C l'ensemble des intervalles communs n'est pas calculé mais est donné en entrée à l'algorithme de tri. Il correspond aux c Intervalles communs générés. Comme nous ne prenons plus en compte tous les Intervalles communs mais exclusivement ceux spécifiquement générés, l'algorithme SRAC ne garantit plus de trouver un MCSS. En effet, lorsque deux Intervalles communs s'intersectent, on suppose avant de les trier, que leurs sous Intervalles communs sont déjà triés. En donnant à l'algorithme SRAC un ensemble d'Intervalles communs incluant des Intervalles communs qui s'intersectent, sans que leurs sous Intervalles communs soient inclus, celui-ci n'est plus correct car les sous Intervalles communs ne seront pas préalablement triés. Le tri d'un des Intervalles communs qui intersecte un autre ne donnera plus forcément un multi-strip.

Prenons l'exemple suivant constitué des deux Intervalles communs avec une intersection $c_1 = \{1,2,3,4,5,6\}$ et $c_2 = \{4,5,6,7,8,9\}$ de la permutation $\pi^s =$ -3 -1 +2 -5 +4 -6 -9 -8 -7. D'après l'algorithme de tri, on peut par exemple commencer par trier c_1 qui est un Intervalle commun négatif. On le trie donc

vers la permutation identité négative pour obtenir la permutation résultante : π^s = -6 -5 -4 -3 -2 -1 -9 -8 -7. L'Intervalle commun c_2 a été cassé et le scénario proposé n'est pas conforme à C. Pour que le scénario généré soit conforme et minimal nous rajoutons à l'ensemble C les sous Intervalles communs dus aux Intervalles communs qui s'intersectent. Il n'est pas nécessaire de rajouter l'Intervalle commun les englobants car celui-ci est déjà trié par le tri des deux Intervalles communs qui s'intersectent.

Pour le calcul de la distance conforme, nous rajoutons les sous Intervalles communs liés aux Intervalles communs qui s'intersectent.

Nous comparons la distance conforme calculée par rapport à la distance usuelle (le nombre minimum d'inversions). Pour chacune des deux distances nous calculons l'erreur absolue moyenne relative à la distance d générée par le modèle d'évolution. La figure 6.1 représente la différence entre les erreurs absolues calculées pour les trois valeurs du paramètre c.

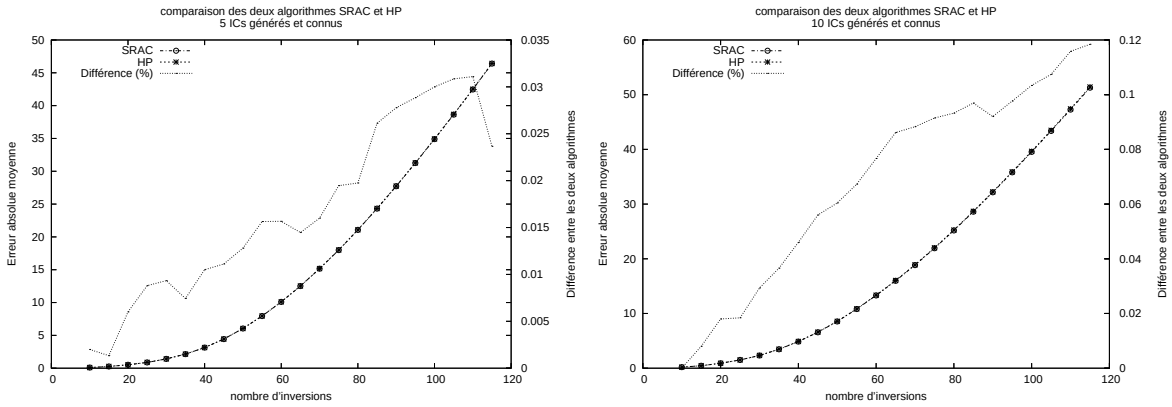
Les deux courbes des erreurs absolues se confondent l'une et l'autre. Nous avons donc aussi représenté la différence entre les deux algorithmes comme le pourcentage de gain obtenu par l'algorithme SRAC par rapport à l'algorithme HP. Ce gain est calculé par rapport à la distance générée. Que la différence soit minime entre les deux algorithmes est normal étant donné que par ce protocole expérimental, la distance évolutive ne peut être modifiée que par au plus le nombre d'Intervalles communs. D'ailleurs, tous les Intervalles communs ne permettent pas de modifier la distance évolutive. Ces expériences nous montrent que les Intervalles communs modifiant la distance évolutive sont rares. Nous avons vu, en introduction du chapitre quatre, un exemple où un Intervalle commun augmente de trois inversions la distance évolutive, mais ceci est un cas extrêmement rare. Lorsque nous avons étudié le diamètre du problème, nous avons vu que dans le meilleur des cas, chaque Intervalle commun pouvait augmenter de une inversion la distance évolutive.

On remarque que les deux algorithmes ne sont pas de bons estimateurs de la distance évolutive réelle dès que le nombre d'inversions augmente. Cela signifie que dès que le nombre d'inversions dépasse un certain seuil, les permutations sont saturées en inversions. Le signal évolutif est en grande partie perdu. Les Intervalles communs permettent de mieux conserver ce signal, mais la saturation est tellement importante que l'ajout de ceux-ci, n'améliore pas significativement l'estimation de la vraie distance.

Plus le nombre d'Intervalles communs augmente, plus le gain obtenu par l'algorithme SRAC par rapport à HP est important. Celui-ci reste minime et atteint au plus 0.2 pour-cent d'erreur absolue moyenne dans le cas de 15 Intervalles communs générés.

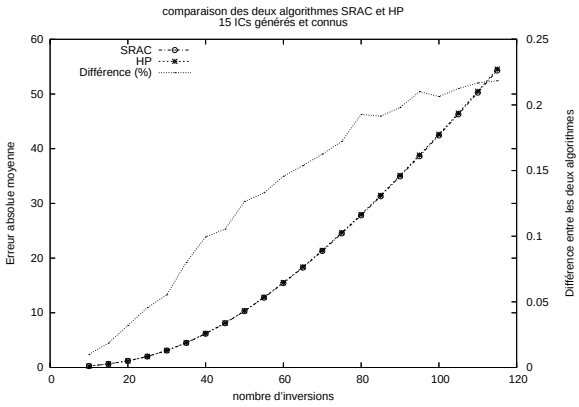
Évaluation de la distance conforme sans Intervalle commun

Nous avons fait l'hypothèse que l'évolution de l'ordre des gènes chez certaines familles d'espèces conservait des Intervalles communs. À partir de cette hypothèse, nous avons proposé un nouveau problème (le MCSS) et par conséquent une nouvelle distance évolutive. Nous voulons connaître maintenant l'erreur induite par notre distance dans le cas où il n'existe pas d'Intervalle commun expressément conservé. Si l'erreur est importante, alors notre méthode ne peut servir à comparer conjointement des génomes avec et sans Intervalles communs. Dans le cas contraire, l'utilisation de notre méthode pour de telles comparaisons incluant beaucoup de



(a)

(b)



(c)

Figure 6.1: Modèle d'évolution avec Intervalles communs. Erreur absolue et pourcentage de différence entre celles-ci calculés pour 5, 10 et 15 Intervalles communs (ICs) générés. L'algorithme de calcul de la distance conforme (SRAC) connaît les Intervalles communs générés. Bien que la distance conforme donne une meilleure estimation de la distance réelle, la différence entre les deux algorithmes est minime et ils ont tous les deux une erreur absolue importante.

génomes est possible. Nous faisons remarquer au lecteur que dans ce cas, la distance conforme ne respecte plus l'inégalité triangulaire.

Pour répondre à cette interrogation, nous avons généré aléatoirement des permutations suivant le procédé suivant : pour une distance évolutive d et un nombre de gènes n donnés, nous construisons la permutation identité positive de taille n et nous lui appliquons d inversions. Une inversion est choisie en sélectionnant de manière équiprobable deux points de cassure parmi les $n+1$ possibles. Nous avons fixé n, la taille des permutations, à 100 gènes. Pour les différentes valeurs de d nous avons répété l'expérience 5000 fois.

La figure 6.2 donne pour des distances évolutives allant de 10 à 115 par pas de 5, l'erreur absolue moyenne de la distance évolutive calculée par l'algorithme HP et celle calculée par l'algorithme SRAC. Les Intervalles communs sont préalablement calculés avec l'algorithme

proposée par Heber et Stoye [Heber and Stoye, 2001a]. L'erreur absolue est calculée en prenant comme référence la distance évolutive utilisée pour générer les permutations.

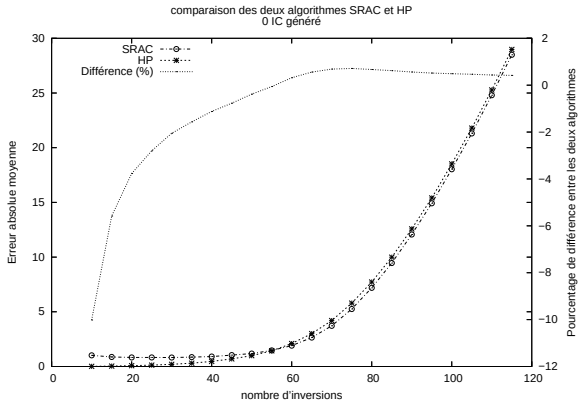


Figure 6.2: Modèle d'évolution usuel, équiprobabilité de toutes les points de cassure. Erreur absolue moyenne entre HP et SRAC. Calculs effectués sur 5000 permutations de taille 100 pour chaque distance évolutive fixée (10,15,...,115). La courbe avec les étoiles est l'erreur absolue moyenne donnée par l'algorithme HP et la courbe avec les ronds, l'erreur absolue moyenne pour l'algorithme SRAC. La courbe en pointillés est le pourcentage de différence entre les deux algorithmes par rapport à la distance évolutive réelle. Lorsque celle-ci est positive, l'algorithme SRAC est un meilleur estimateur de l'évolution que l'algorithme HP. On observe que pour des permutations générées avec moins de 55 inversions, l'algorithme HP donne une erreur absolue moyenne inférieure à celle de SRAC. Et inversement, mais dans une moindre mesure, pour des permutations générées avec plus de 55 inversions, l'algorithme SRAC donne une erreur absolue inférieure à celle donnée par l'algorithme HP.

Dans l'expérimentation précédente où SRAC correspondait au modèle d'évolution, nous avons observé que celui-ci ne permettait pas d'estimer correctement les distances évolutives. On observe avec cette nouvelle expérience que de manière similaire à l'expérimentation précédente, l'algorithme HP ne permet pas de trouver la distance évolutive réelle. Et cela malgré qu'il concorde exactement avec le modèle qui a généré les permutations. Plus la distance évolutive augmente plus l'erreur est importante. A partir d'un certain nombre d'inversions, on dira que les permutations sont saturées en inversions.

L'algorithme SRAC, quant à lui, donne une erreur absolue moyenne supérieure à celle de HP pour des distances évolutives inférieures à 55 inversions. Pour des distances supérieures à 55 inversions, l'algorithme SRAC donne une erreur légèrement inférieure à celle de HP et est donc meilleur que celui-ci.

On peut expliquer ce phénomène par le nombre d'Intervalles communs dus aux circonstances. Pour des distances évolutives petites, les permutations générées comportent des Intervalles communs dus au peu de mélange de celles-ci. Leur prise en compte pour le calcul de la distance évolutive la modifie sans raison réelle. L'algorithme SRAC donne alors une erreur supérieure à celle de HP. Ce point est débattu en détail dans ce chapitre page 150. Pour des distances évolutives plus importantes, ce nombre d'Intervalles communs non définis diminue jusqu'à ce qu'il n'en reste plus que quelques uns. Ceux persistants sont bien réels même si

on ne les a pas expressément générés. Ils représentent un signal évolutif, ce qui permet à l'algorithme SRAC de donner une erreur moindre que celle générée par HP.

On remarque que la distance conforme devient meilleure que la distance simple dès que les permutations sont saturées en inversions. La distance conforme est alors un meilleur estimateur d'évolution, mais comme son erreur absolue moyenne croit exponentiellement, son utilisation est limitée.

Avec des Intervalles communs non communiqués à SRAC

Nous avons vu précédemment que SRAC n'était pas forcément moins bon que HP pour des permutations générées sans Intervalle commun. Nous comparons maintenant les deux algorithmes pour des permutations générées avec des Intervalles communs. Ceux-ci ne sont pas connus à l'avance lors du calcul de la distance conforme. Nous devons les détecter en utilisant les algorithmes de Heber et Stoye.

Nous générons aléatoirement des permutations par le procédé suivant : pour une distance évolutive d, un nombre d'Intervalles communs c et un nombre de gènes n donné, nous construisons la permutation identité positive de taille n ; nous générons aléatoirement et de manière équiprobable c intervalles ; nous appliquons alors à la permutation d inversions. Une inversion est choisie en sélectionnant de manière équiprobable deux points de cassure parmi les $n+1$ possibles. Tant que cette inversion n'est pas conforme aux intervalles prédéfinis, on recommence sa sélection. Nous avons fixé n, la taille des permutations, à 100 gènes. Pour chaque valeur de d et de c, nous avons répété l'expérience 5000 fois. Les intervalles présélectionnés deviennent les seuls Intervalles communs au fur et à mesure que des inversions mélangent les permutations. De même que précédemment, une inversion peut-être sélectionnée deux fois et ainsi s'annihiler.

La figure 6.3 donne pour des distances évolutives allant de 10 à 115 et un nombre d'Intervalles communs présélectionnés allant de 0 à 15, l'erreur absolue moyenne de la distance évolutive calculée par l'algorithme HP et celle calculée par l'algorithme SRAC. L'erreur absolue est calculée en prenant comme référence la distance évolutive utilisée pour générer les permutations.

On observe, comme pour le cas sans Intervalle commun, que l'algorithme HP donne une erreur moindre pour des permutations peu mélangées. Pour des permutations plus mélangées, c'est l'algorithme SRAC qui donne une erreur moindre que celle de HP. D'ailleurs, plus le nombre d'intervalles prédéfinis augmente, plus l'intervalle de distances où l'algorithme HP est meilleur est petit. Par exemple, sans intervalle prédéfini, la bascule entre les deux algorithmes s'effectue un peu avant 55 inversions. Pour 15 intervalles prédéfinis, la bascule a lieu pour 32 inversions.

L'algorithme SRAC ne permet pas d'améliorer de beaucoup le calcul des distances évolutives par inversions. L'erreur absolue de SRAC, bien que plus petite que celle de HP, est importante. SRAC malgré le fait qu'il soit conforme au modèle qui a généré les données ne donne pas de bien meilleurs résultats. Ceci peut s'expliquer par le fait que HP lui-même a une erreur absolue importante pour des distances élevées lorsque le modèle générant les données n'a pas d'Intervalles communs. De plus, SRAC devient meilleur lorsque le nombre d'inversions augmente, or au même moment, le phénomène de saturation des permutations a lieu.

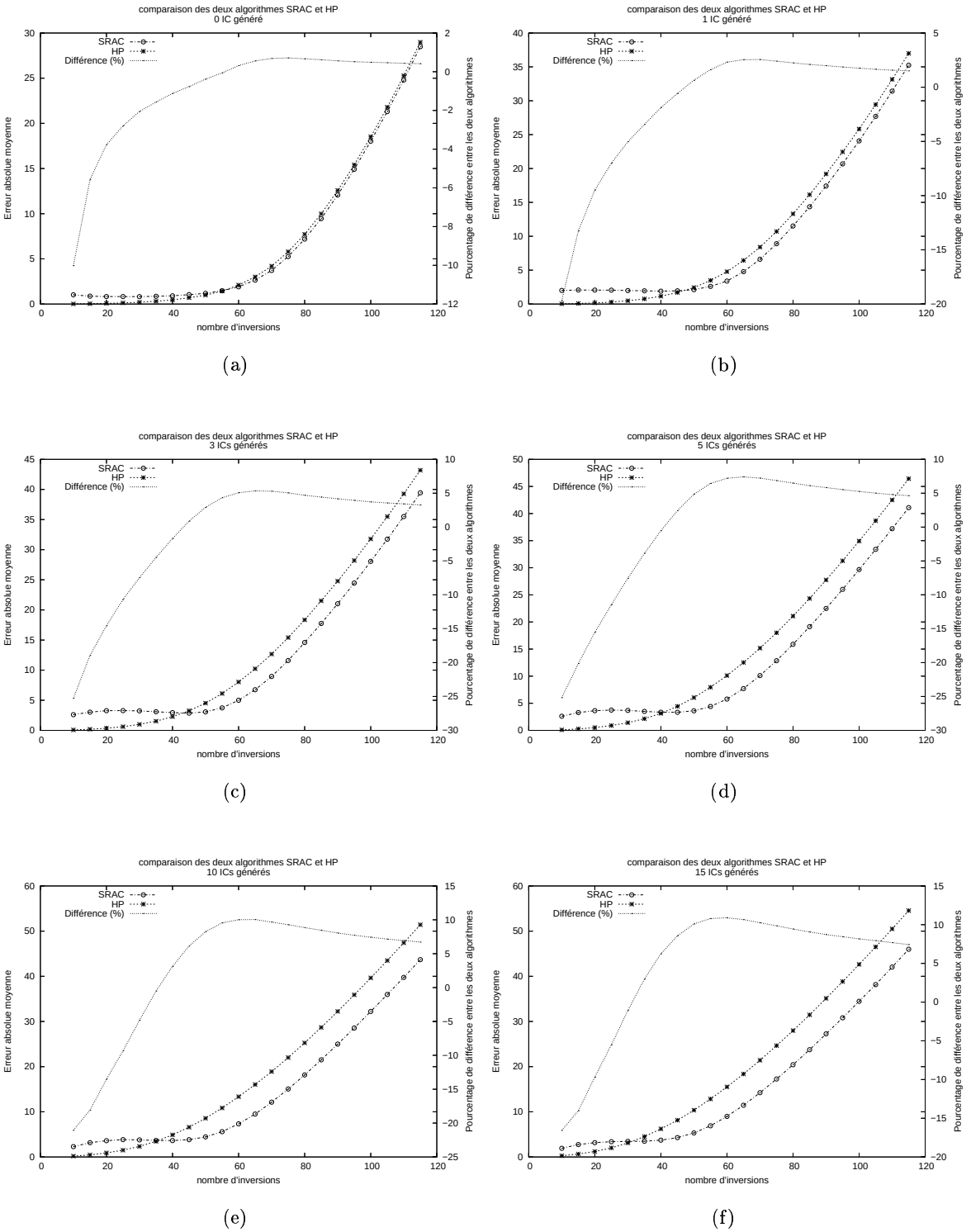


Figure 6.3: Modèle d'évolution avec Intervalles communs (non connus). Erreur absolue moyenne entre HP et SRAC. Calculs effectués sur 5000 permutations de taille 100 pour chaque distance évolutive fixée (10,15,...,115) et pour 1,3, 5, 10 et 15 intervalles préfixés. La courbe avec les étoiles donne l'erreur absolue moyenne de l'algorithme HP et la courbe avec les ronds, l'erreur absolue moyenne de l'algorithme SRAC. La courbe en pointillés représente le pourcentage de différence entre les deux algorithmes par rapport à la distance évolutive réelle. Quand celle-ci est positive, l'algorithme SRAC est meilleur que l'algorithme HP.

C'est pourquoi, pour ce modèle d'évolution, SRAC ne permet d'améliorer que légèrement les distances évolutives calculées.

Il est intéressant de noter le très mauvais comportement de l'algorithme HP sur ce modèle d'évolution. Avec 15 Intervalles communs, celui-ci génère une erreur absolue moyenne de plus de 5 inversions dès une distance réelle de 36 inversions. Alors que dans le cas où nous ne générons pas d'Intervalles communs, HP génère une erreur de plus de 5 inversions à partir de seulement 75 inversions.

Nous avons remarqué précédemment que la distance conforme devient meilleure que la distance simple dès que les permutations sont saturées en inversions. La distance conforme est alors un meilleur estimateur d'évolution, mais comme son erreur absolue moyenne croit exponentiellement, son utilisation est limitée. Le phénomène a toujours lieu lorsque nous générons des Intervalles communs. De plus les Intervalles communs morcellent la permutation en sous-permutations. Lesquelles de taille plus petites sont plus vite soumises au phénomène de saturation.

Étude des artefacts de SRAC

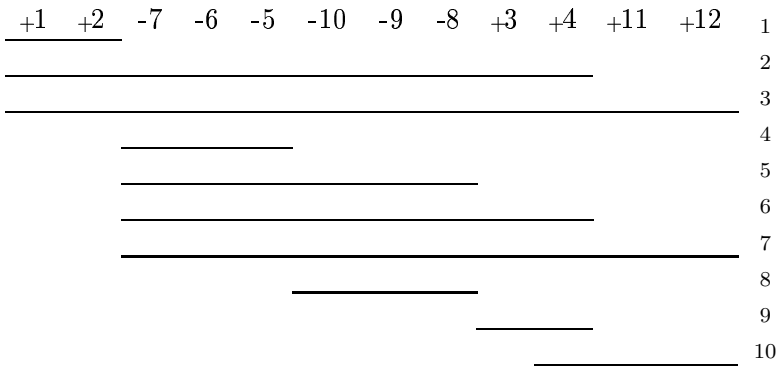
L'algorithme SRAC donne de meilleures estimations pour des distances évolutives élevées. Par contre, pour des distances moindres, l'algorithme HP devient meilleur bien qu'il ne prenne pas en compte les Intervalles communs. Ceci provient du trop grand nombre d'Intervalles communs existant dans des permutations peu mélangées. Les Intervalles communs ne représentent plus une contrainte mais du hasard.

Nous exposons une des raisons expliquant l'erreur absolue importante de SRAC quand la distance évolutive réelle est petite. Il existe un biais important induit par les inversions qui s'intersectent. Soit deux inversions avec une intersection :

$$\begin{array}{rcccccccccccc}
\pi^s = & +1 & +2 & \underline{+3} & \underline{+4} & \underline{+5} & \underline{+6} & \underline{+7} & +8 & +9 & +10 & +11 & +12 \\
 & +1 & +2 & -7 & -6 & -5 & \underline{-4} & \underline{-3} & \underline{+8} & \underline{+9} & \underline{+10} & +11 & +12 \\
 & +1 & +2 & -7 & -6 & -5 & -10 & -9 & -8 & +3 & +4 & +11 & +12
\end{array}$$

Avec deux inversions nous avons créé un grand nombre d'Intervalles communs. Pour simplifier, nous ne prenons pas en compte les Intervalles communs sans point de cassure ni à droite ni à gauche [6]. Nous obtenons un ensemble de 10 Intervalles communs :

[6] Ceux-ci ne modifient pas les distances par inversions.



Le tri de cette permutation avec l'algorithme SRAC et tous les Intervalles communs donne une distance évolutive de 5 inversions, 3 de plus que la vraie distance qui est d'ailleurs celle donnée par l'algorithme HP. Le problème est dû au trop grand nombre d'Intervalles communs ne reflétant pas des contraintes évolutives. L'Intervalle commun qui nous pose principalement problème ici, est le numéro cinq. Il regroupe deux sous Intervalles communs qui ont été séparés pendant l'évolution.

Nous pouvons refaire les tests précédents en éliminant ce type de structure. Les Intervalles communs issus de ce type d'évolution ne seront pas pris en compte dans le calcul de la distance conforme. On obtient les résultats exprimés figure 6.4.

Par rapport aux expériences précédentes, quelque soit le nombre d'Intervalles communs, l'erreur absolue est moins importante pour des permutations générées avec un petit nombre d'inversions. Par contre, comme précédemment, la distance conforme donne de moins bons résultats que la distance de HP pour un petit nombre d'inversions. Mais cette différence est relativement plus petite. Ce qui montre que le traitement proposé supprime le biais étudié ici. De plus, pour un nombre d'inversions plus important, la distance conforme reste un meilleur estimateur de la distance évolutive. Le gain apporté par la distance conforme est légèrement inférieur en éliminant ces structures.

Le problème de la saturation

Nous venons d'éliminer un des biais de la distance conforme lorsque le nombre d'inversions est petit. Nous nous intéressons maintenant aux problèmes liés à la saturation en inversions des permutations. C'est-à-dire, quand le nombre d'inversions est important.

La distance conforme ne peut donner des résultats bien meilleurs qu'une distance plus simple car elle nécessite l'existence d'Intervalles communs mélangés. Or dès que ceux-ci sont suffisamment mélangés pour être identifiés comme des Intervalles communs, les permutations sont saturées en inversions.

Nous avons précédemment proposé un modèle d'évolution simpliste. pour une distance évolutive d, un nombre d'Intervalles communs c et un nombre de gènes n donné, nous construisons la permutation identité positive de taille n ; nous générons aléatoirement et de manière équiprobable c intervalles ; nous appliquons alors à la permutation d inversions. Une inversion est choisie en sélectionnant de manière équiprobable deux points de cassure. Tant que cette inversion n'est pas conforme aux intervalles sélectionnées, on recommence sa sélection.

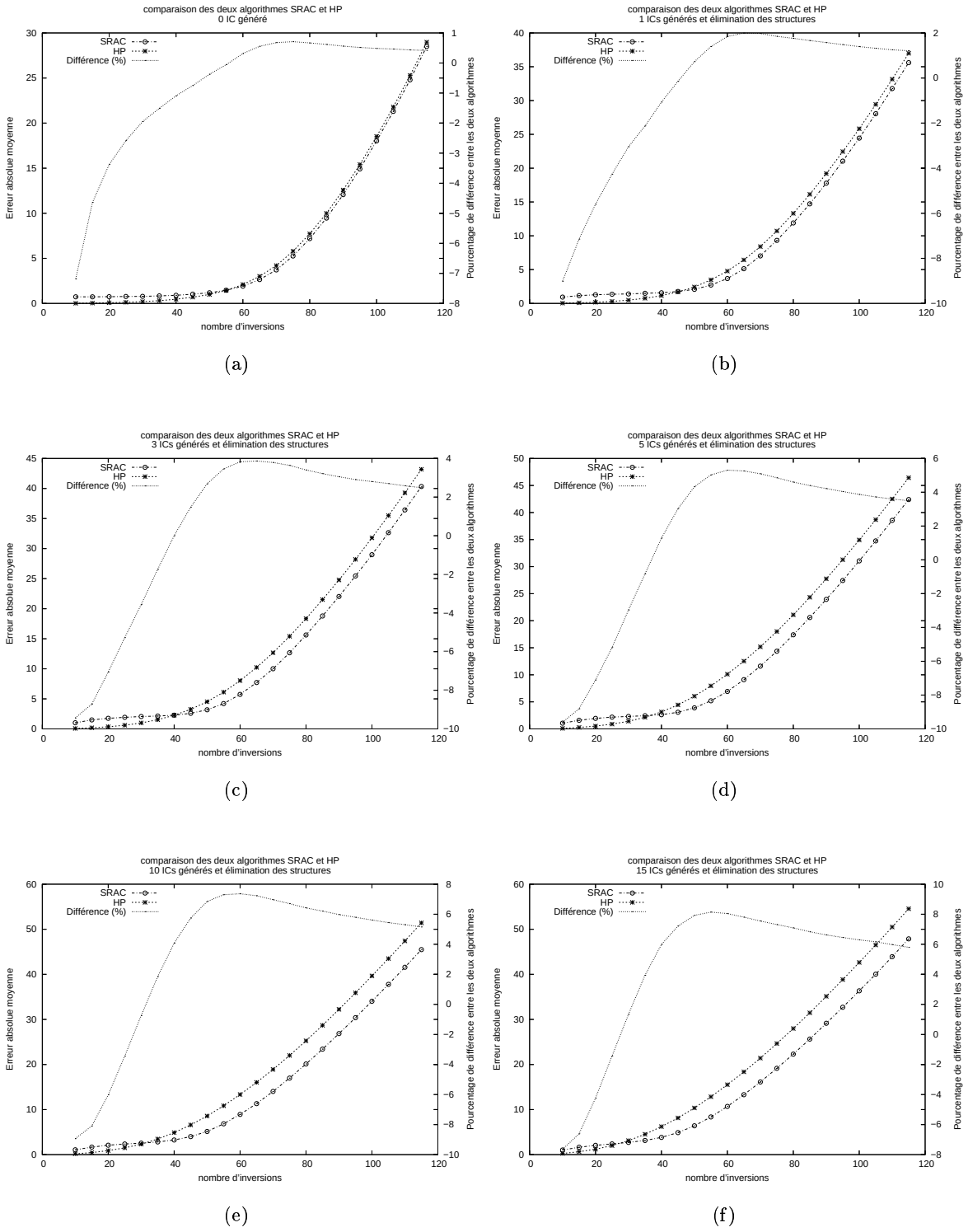


Figure 6.4: Modèle d'évolution avec Intervalles communs (non connus et élimination de certains). Erreur absolue moyenne entre HP et SRAC. Calculs effectués sur 5000 permutations de taille 100 pour chaque distance évolutive fixée (10,15,...,115) et pour 0,1,3, 5, 10 et 15 intervalles préfixés. La courbe avec les étoiles donne l'erreur absolue moyenne de l'algorithme HP et la courbe avec les ronds, l'erreur absolue moyenne de l'algorithme SRAC. Nous avons éliminé des Intervalles communs détectés ceux provenant des inversions avec intersection.

Par ce procédé, les inversions à l'intérieur des petits Intervalles communs sont moins probables que les autres. Par conséquent, les petits Intervalles communs ne sont que peu mélangés. Pour qu'ils soient mélangés, il faut que le nombre d'inversions soit important et donc que les permutations soient saturées. Le mélange de la permutation n'est pas homogène. Pour remédier au biais de ce modèle d'évolution, nous en proposons un autre.

Soit la permutation identité de taille n. Soit une distance évolutive donnée et représentée par un pourcentage d d'inversions. Ce pourcentage est exprimé en fonction du nombre maximum d'inversions nécessaires pour trier parcimonieusement une permutation, c'est-à-dire $n+1$ inversions. Soit un nombre d'Intervalles communs c chacun de taille au plus $\frac{c}{n}$. Ces Intervalles communs ne peuvent s'intersecter, ni être inclus les uns dans les autres. Nous sélectionnons aléatoirement et de manière équiprobable c Intervalles communs ; nous appliquons alors à chaque Intervalle commun un nombre d'inversions conformes relatif au pourcentage d. Nous faisons de même avec la permutation elle-même. Au final, le pourcentage d'inversions subi par la permutation est d. Une inversion est choisie en sélectionnant de manière équiprobable deux points de cassure dans l'intervalle autorisé. Tant que cette inversion n'est pas conforme aux intervalles prédéfinis, on recommence sa sélection. Nous avons appelé ce modèle d'évolution, le modèle d'évolution homogène.

La figure 6.5 donne pour des distances évolutives allant de 10 à 115 suivant le modèle évolutif homogène et un nombre d'Intervalles communs présélectionnés allant de 5 à 15, l'erreur absolue moyenne de la distance évolutive calculée par l'algorithme HP et celle calculée par l'algorithme SRAC. La moyenne est effectuée sur 5000 expériences aléatoires. L'erreur absolue est calculée en prenant comme référence le pourcentage de distance évolutive utilisé pour générer les permutations. Les permutations générées sont de taille 100. Pour un petit nombre d'inversions, les résultats sont sujets aux erreurs d'arrondi. Nous avons donné en entrée à l'algorithme SRAC les Intervalles communs générés. La figure 6.6 reprend les mêmes expériences sans donner à SRAC les Intervalles communs. Ceux-ci sont détectés par l'algorithme proposé par Heber et Stoye. La figure 6.7, quant à elle, représente les expériences où l'on ne connaît pas au préalable les Intervalles communs et où on n'élimine pas les structures dues aux inversions qui s'intersectent.

La première expérience donne des résultats similaires pour les deux algorithmes HP et SRAC. Il faut noter que pour cette expérience, l'algorithme SRAC est toujours meilleur que l'algorithme HP, même quand les Intervalles communs ne sont pas préalablement connus. Il est vrai que cette différence entre les deux algorithmes est minime mais les expériences montrent que l'on a toujours intérêt à utiliser l'algorithme SRAC. Les résultats de HP sont similaires à ceux issus du premier modèle d'évolution pour un grand nombre d'inversions ; par contre, la gain de SRAC est plus important avec ce modèle. Pour 15 Intervalles communs, le gain de SRAC passe de 0.2% à 0.45% environ.

Les deux dernières expériences, où l'on ne connaît pas préalablement les Intervalles communs, donnent de bons résultats. Plus on ajoute de contraintes en augmentant le nombre d'Intervalles communs, plus les résultats calculés par SRAC sont meilleurs par rapport à ceux de HP. Si on génère 15 Intervalles communs, l'algorithme SRAC donne à peu près la même erreur absolue moyenne que l'algorithme HP pour des permutations peu mélangées (jusqu'à environ 40 inversions pour des permutations de taille 100). Pour le même nombre d'Intervalles communs et plus de 40 inversions, les distances calculées par l'algorithme SRAC sont 10% meilleures que celles calculées par HP. L'algorithme SRAC garde une erreur absolue moyenne

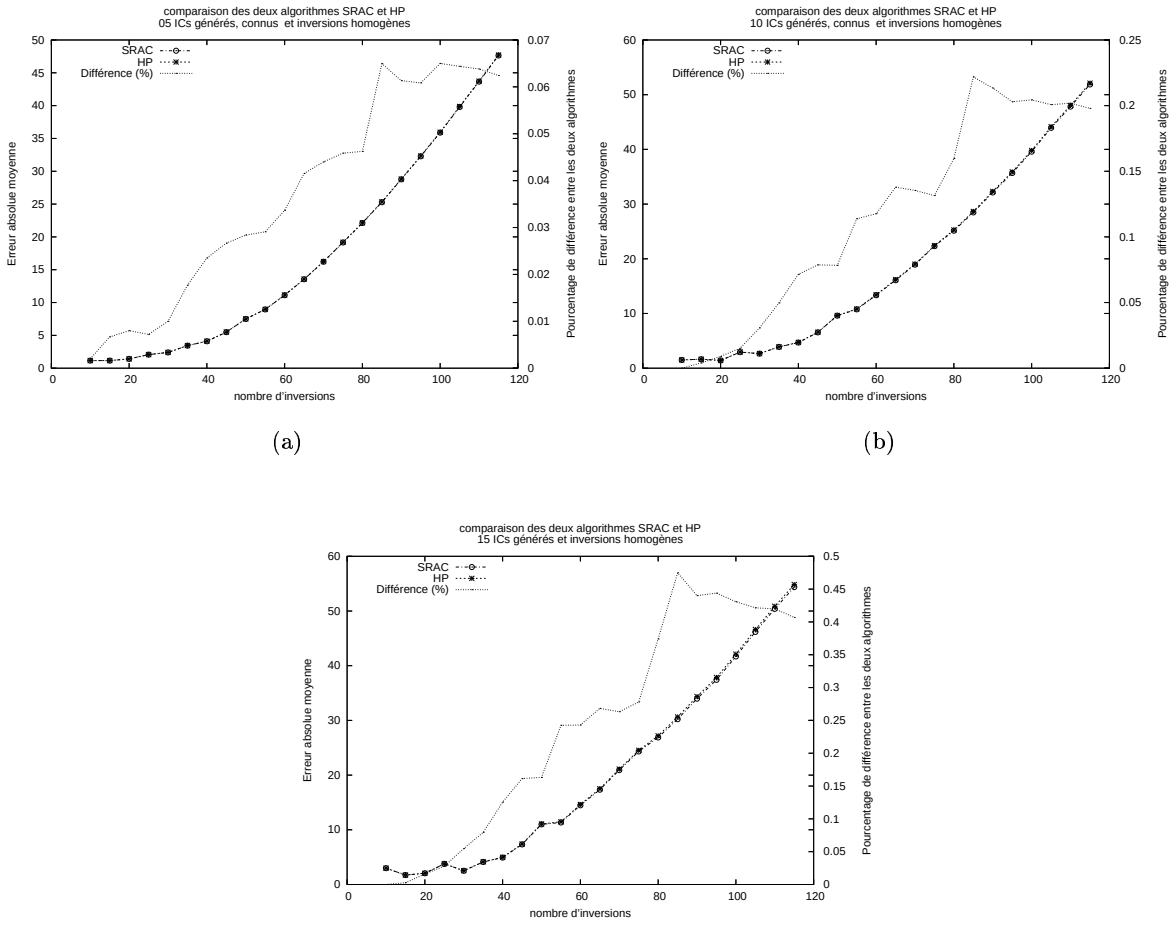


Figure 6.5: Modèle d'évolution homogène. Erreur absolue obtenue pour les deux algorithmes SRAC et HP sur des permutations générées suivant le modèle d'évolution homogène. L'erreur absolue est une moyenne sur 5000 permutations. Les Intervalles communs prédéfinis sont connus par l'algorithme SRAC.

inférieure à 10 inversions pour des taux d'évolution allant jusqu'à 65 inversions pour 100 gènes. Pour ce même taux d'inversions, HP donne une erreur absolue moyenne de 17.5 inversions. Jusqu'au taux de 45 inversions, SRAC reste en dessous de 5 inversions d'erreur absolue en moyenne. Alors que pour les mêmes taux d'inversions, HP donne une erreur absolue moyenne dans le pire des cas de 7.4 inversions.

Pour un petit nombre d'Intervalles communs et un petit nombre d'inversions, l'élimination des structures dues aux inversions qui s'intersectent permet de mieux évaluer les distances évolutives. Mais dans tous les autres cas, ne pas les éliminer, permet de prendre en compte tous les Intervalles communs et d'obtenir des meilleures distances évolutives. Sachant qu'avec ce modèle d'évolution les permutations sont mieux mélangées, ce qui élimine d'office les Intervalles communs dus au peu de mélange, il est logique d'obtenir de meilleurs résultats.

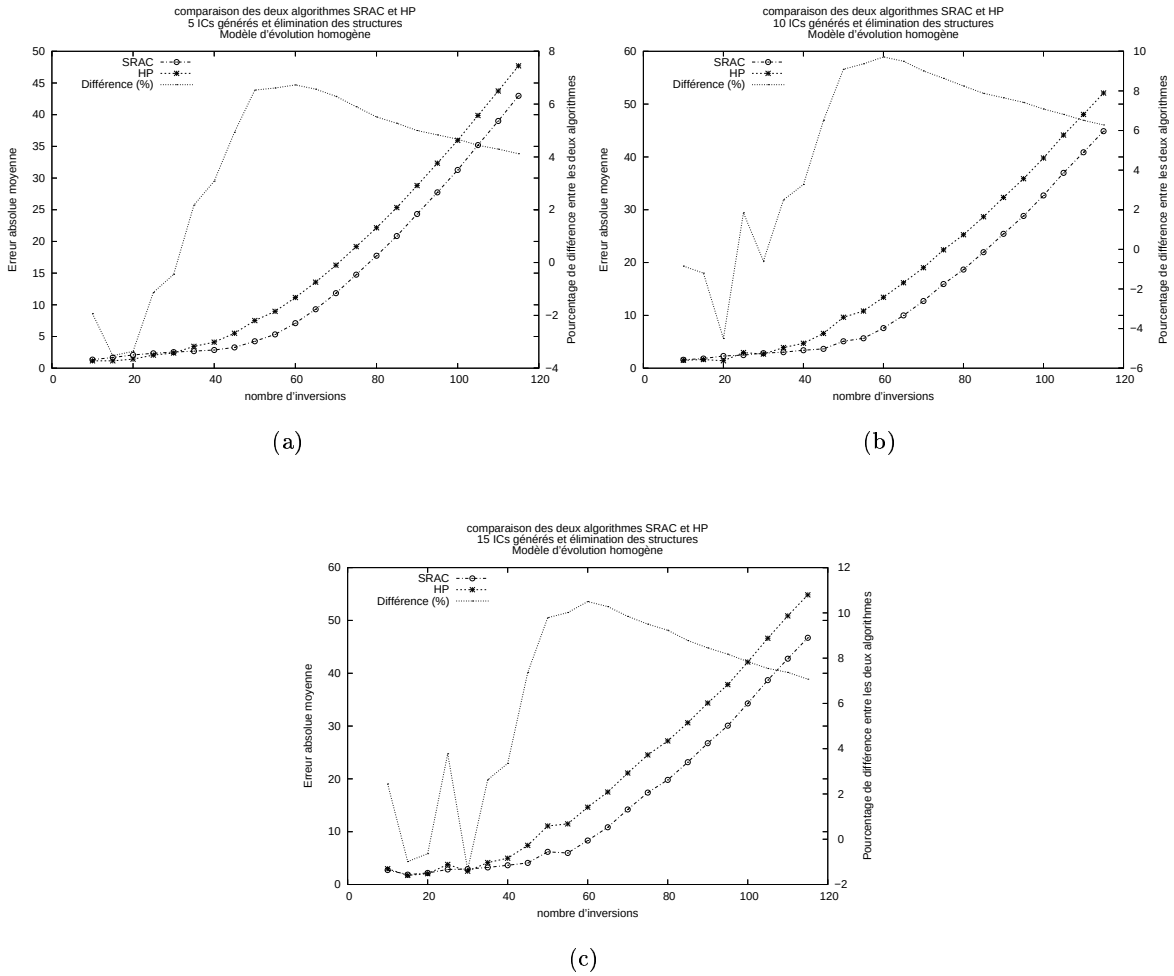


Figure 6.6: Modèle d'évolution homogène (ICs non connus et élimination des structures). Erreur absolue obtenue pour les deux algorithmes SRAC et HP sur des permutations générées suivant le modèle d'évolution homogène. L'erreur absolue est une moyenne sur 5000 permutations. Les Intervalles communs prédéfinis ne sont pas connus par l'algorithme SRAC. Les Intervalles communs sont préalablement calculés et les structures spéciales dues aux inversions qui s'intersectent sont éliminées.

6.2 Des modèles d'évolution plus réalistes

Nous essayons maintenant de générer des permutations avec des modèles d'évolutions basés sur des observations biologiques. De tels modèles n'existent pas encore et ceux que nous présentons sont des propositions.

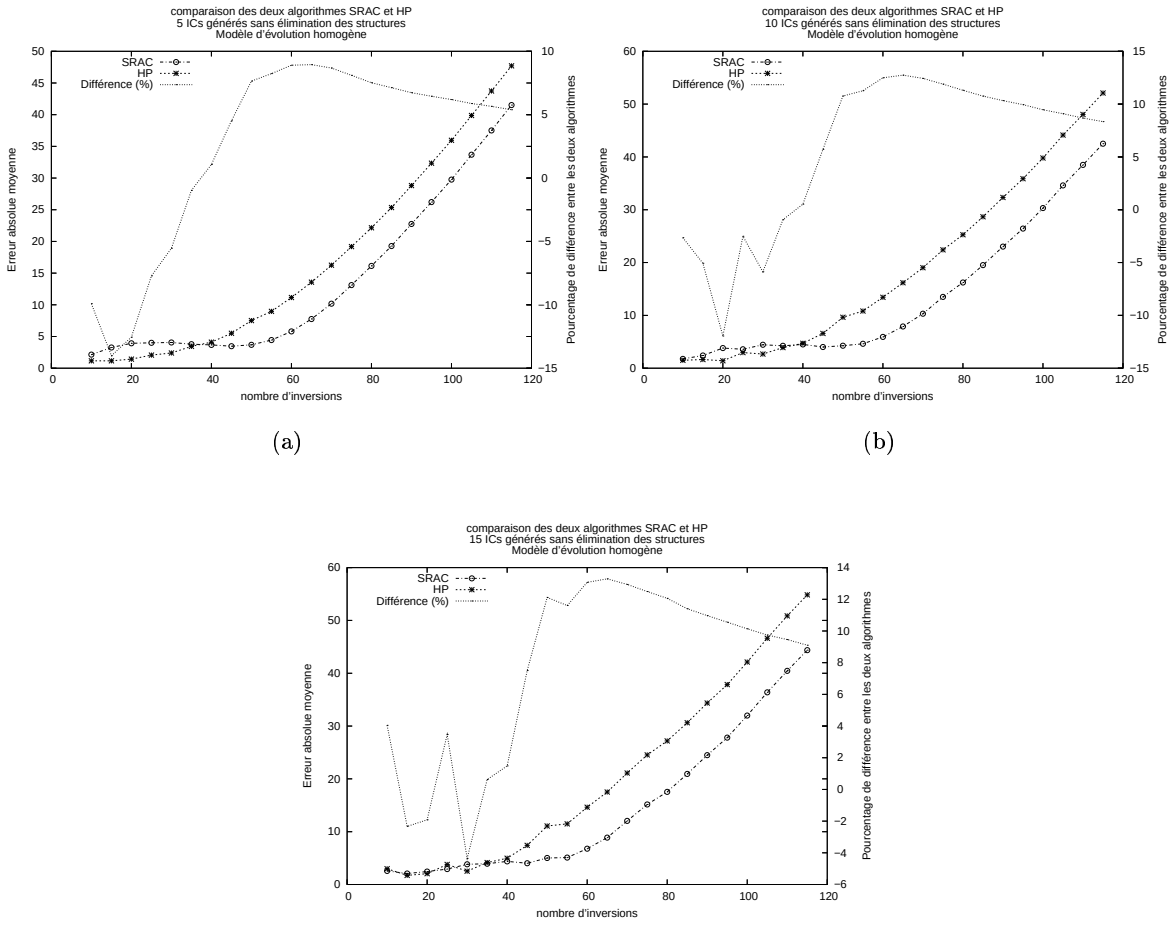


Figure 6.7: Modèle d'évolution homogène (ICs non connus). Erreur absolue moyenne obtenue pour les deux algorithmes SRAC et HP sur des permutations générées suivant le modèle d'évolution homogène. L'erreur absolue est une moyenne sur 5000 permutations. Les Intervalles communs prédéfinis ne sont pas connus par l'algorithme SRAC. Les Intervalles communs sont préalablement calculés et les structures spéciales dues aux inversions qui s'intersectent ne sont pas éliminées.

Avec des Inversions centrées

Nous avons présenté dans le chapitre 0.3 page 17 le modèle d'évolution centrée. Les seules inversions autorisées sont alors centrées sur l'origine de réplication. Nous avons comme précédemment généré des permutations suivant ce modèle d'évolution.

Pour une distance évolutive d et un nombre de gènes n donné, nous construisons la permutation identité positive de taille n ; nous appliquons alors à la permutation d inversions centrées sur le gène $\frac{n}{2}$. Une inversion est choisie en sélectionnant de manière équiprobable un point de cassure parmi $\frac{n}{2}$. L'autre point de cassure, est obtenu par symétrie.

Nous avons effectué les mêmes expériences que précédemment avec 5000 permutations générées pour chaque distance évolutive. Le résultat est sans appel. Il n'y a aucune différence entre HP et SRAC pour des données générées avec des inversions toutes centrées sur le même point. Pourtant, les inversions centrées génèrent des Intervalles communs. Une inversion de ce type ne cassera jamais un Intervalle commun et donc, dans le pire des cas, ne modifie pas le nombre d'Intervalles communs. Autrement dit, si les Intervalles communs sont générés par des inversions centrées, alors les distances calculées par l'algorithme HP sont conformes aux Intervalles communs. L'algorithme HP permet de trouver un scénario parcimonieux et conforme.

Nous obtenons ce résultat pour des données parfaites. C'est-à-dire que les inversions sont toutes exactement centrées sur la même origine. On observe dans les données réelles (voir section 0.3 page 17) que toutes les inversions ne sont pas parfaitement centrées sinon la forme de X serait précise et toujours parfaitement présente.

Inversions centrées et inversions d'un seul gène

Il est évident que le modèle proposé précédemment n'est pas valide du point de vue biologique. Il est trop "parfait" et n'autorise aucune erreur. On observe effectivement des inversions centrées sur l'origine de la réplication mais aussi d'autres inversions plus petites. Il existe dans les génomes une préférence pour les inversions de petites taille et plus particulièrement des inversions d'un seul gène [Lefebvre et al., 2003, Sankoff, 2002].

Nous proposons donc un autre modèle d'évolution ou nous autorisons en plus des inversions centrées sur l'origine de la réplication, les inversions d'un seul gène. Les résultats sont donnés figure 6.8.

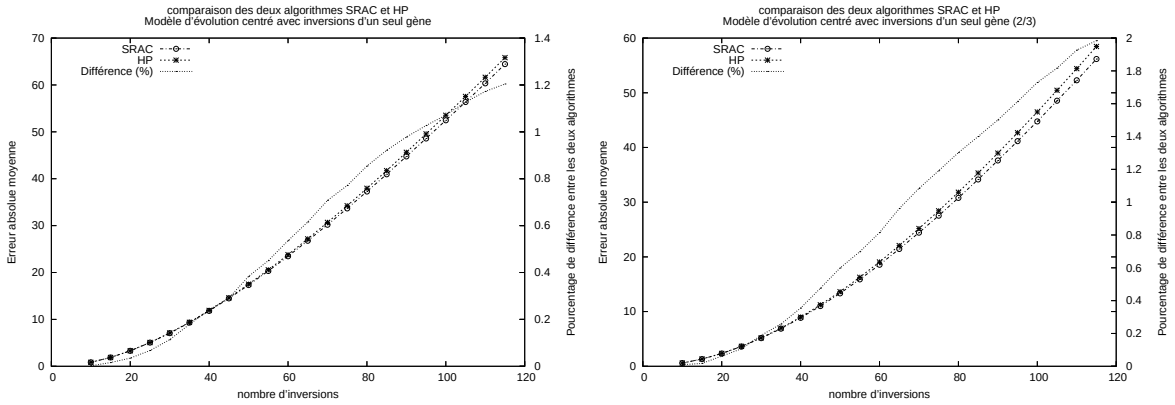
Il y a maintenant une légère différence entre les deux algorithmes. Encore une fois l'algorithme SRAC est toujours meilleur que l'algorithme HP. L'algorithme SRAC est d'autant meilleur que le taux d'inversions d'un seul gène augmente. Les inversions d'un seul gène ne modifient pas les Intervalles communs car ceux-ci, dans la version utilisée, ne sont pas sensibles au signe des gènes. Cela signifie que les mêmes Intervalles communs que dans l'expérience précédente ne sont plus respectés par l'algorithme HP.

Petites inversions

Non seulement les inversions d'un seul gène sont prépondérantes mais il semblerait que les petites inversions le soient aussi [Sankoff, 2002, Dalevi et al., 2002]. Pour simuler ce processus nous proposons un modèle d'évolution basé sur la longueur des inversions.

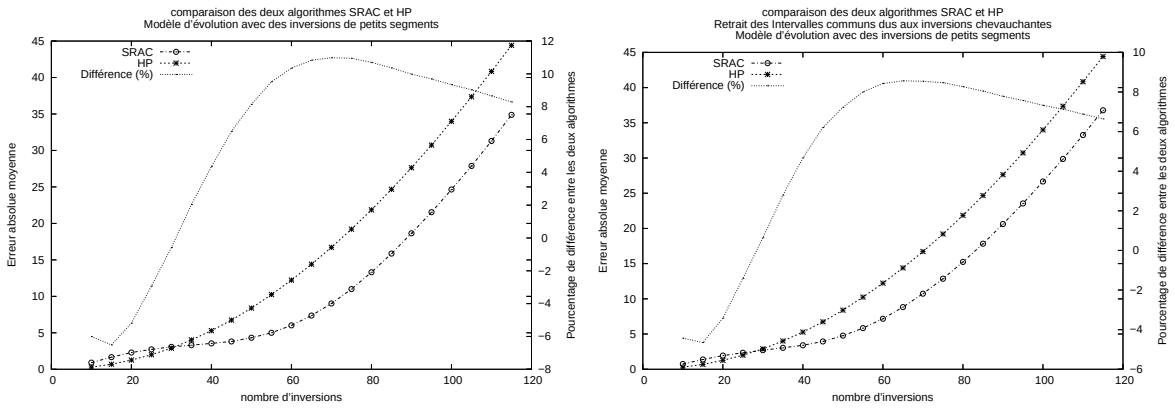
Pour une distance évolutive d et un nombre de gènes n donné, nous construisons la permutation identité positive de taille n ; nous appliquons alors à la permutation d inversions. Une inversion est choisie en sélectionnant de manière équiprobable un point de cassure parmi n ; puis le nombre de gènes inversés suit une loi de probabilité exponentielle p^{x-1}. p étant un paramètre fixé.

Nous avons effectué les mêmes expériences que précédemment avec 5000 permutations générées pour chaque distance évolutive. Nous avons arbitrairement pris $p = 0.5$. Les résultats sont exprimés figure 6.9.



(a) (b)

Figure 6.8: Modèle d'évolution centrée et inversions d'un seul gène. Erreur absolue obtenue pour les deux algorithmes SRAC et HP sur des permutations générées suivant le modèle d'évolution centrée tout en autorisant les inversions d'un seul gène. L'erreur absolue est une moyenne sur 5000 permutations. Les Intervalles communs prédéfinis ne sont pas connus par l'algorithme SRAC. Les Intervalles communs sont préalablement calculés et les structures spéciales dues aux inversions qui s'intersectent ne sont pas éliminées. Les courbes de gauche sont pour un rapport de $\frac{1}{3}$ d'inversions d'un seul gène. Les courbes de droite sont pour un rapport de $\frac{2}{3}$ d'inversions d'un seul gène.



(a) (b)

Figure 6.9: Modèle d'évolution avec de petites inversions. Erreur absolue moyenne obtenue pour les deux algorithmes SRAC et HP sur des permutations générées suivant le modèle d'évolution basé sur la longueur des inversions. L'erreur absolue est une moyenne sur 5000 permutations générées suivant le modèle d'évolution. Les Intervalles communs sont préalablement calculés et les structures spéciales dues aux inversions qui s'intersectent ne sont pas éliminées pour la courbe de gauche. Elles le sont pour la courbe de droite.

Pour ce modèle d'évolution réaliste, on observe que les distances calculées par l'algorithme SRAC sont des meilleurs estimateurs que celles calculées par l'algorithme HP. SRAC permet de gagner jusqu'à 11% d'inversions sur le nombre réel par rapport à HP. On observe toujours le meilleur comportement de HP pour des petites distances évolutives. En éliminant les structures dues aux inversions qui s'intersectent, la différence entre HP et SRAC pour des petites distances s'amoindrit. Bien que SRAC, pour des distances plus grandes, reste bien meilleur que HP, son comportement, dans ce cas de figure où le nombre d'inversions est important et où l'on a supprimé les structures dues aux inversions qui s'intersectent, est moins bon que sans leur élimination.

L'algorithme SRAC donne une erreur moindre avec des petites inversions que lorsque les inversions sont toutes équiprobables. On retrouve les résultats de David Sankoff dans [Sankoff, 2002] qui montre que des petites inversions permettent de générer des clusters de gènes.

Nous pouvons nous assurer que ces bons résultats ne sont pas dû à un phénomène de saturation moins important quand on a des petites inversions. Alors que lorsque les inversions ne sont pas contraintes, HP donne moins de 5 inversions d'erreur en moyenne pour des distances allant jusqu'à 75 inversions pour des permutations de 100 gènes, dans le cas des inversions contraintes par leur taille, dès 40 inversions HP génère plus de 5 inversions d'erreur en moyenne. Avec des petites inversions, l'algorithme HP sature plus vite parce que cela génère des Intervalles communs. Il sature d'autant plus que les inversions d'un seul gène, qui sont prépondérantes pour ce modèle d'évolution, ne lui font pas respecter les Intervalles communs.

Inversions centrées et petites inversions

Nous couplons maintenant les deux modèles précédents. Les inversions autorisées sont petites et leur taille suit la loi de probabilité de paramètre x : p^{x-1}. Et nous autorisons aussi, sans réglementer leur longueur, les inversions centrées sur l'origine de la réplication. Le rapport entre ces deux types d'inversions est défini par le paramètre τ.

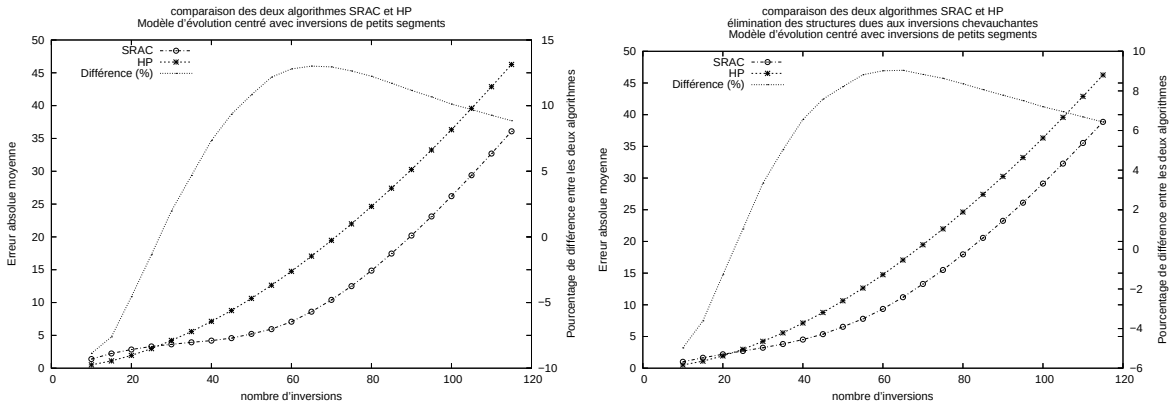
Nous avons expérimentalement comparé des permutations construites suivant ce modèles pour $\tau = 0.5$. Ces expériences ont été répétées 5000 fois pour chaque distance évolutive comprise entre 10 et 115, le tout pour des permutations de taille 100. La figure 6.10 donne l'erreur absolue moyenne des algorithmes HP et SRAC sur ces données.

On observe les mêmes comportements que précédemment où l'on n'autorisait pas les inversions centrées. Par contre avec ce modèle d'évolution, le gain obtenu par SRAC par rapport à HP atteint les 13%. D'ailleurs SRAC reste longtemps sous les 5 inversions d'erreur : jusqu'à 48 inversions alors que l'erreur de HP est déjà de 10 inversions en moyenne.

En éliminant les structures dues aux inversions qui s'intersectent, on diminue un peu les performances de l'algorithme SRAC mais celui-ci devient meilleur que HP dès 28 inversions.

6.3 Temps de calcul

Nous comparons les temps de calcul des algorithmes SRAC et HP. Rappelons que HP est un algorithme de complexité linéaire en temps, alors que SRAC est de complexité exponentielle



(a) (b)

Figure 6.10: Modèle d'évolution centrée et petites inversions. Erreur absolue moyenne obtenue pour les deux algorithmes SRAC et HP sur des permutations générées suivant le modèle d'évolution basé sur la longueur des inversions et autorisant les inversions centrées. Le rapport entre ces deux types d'inversions est de $\frac{1}{2}$. L'erreur absolue est une moyenne sur 5000 permutations générées suivant ce modèle d'évolution. Les Intervalles communs sont préalablement calculés et les structures spéciales dues aux inversions qui s'intersectent ne sont pas éliminées pour la courbe de gauche. Elles le sont pour la courbe de droite.

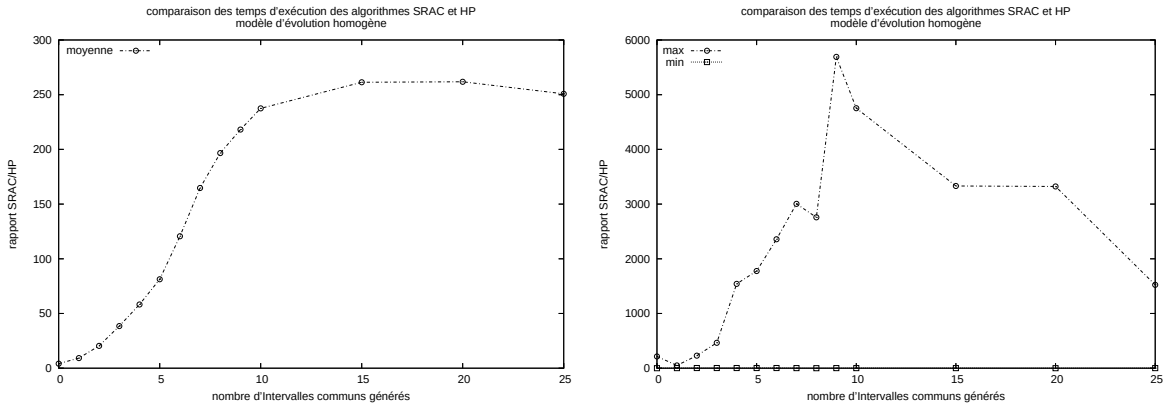
en temps. Mais ces deux algorithmes ne calculent pas la même distance évolutive. Plutôt que de comparer directement le temps de calcul de ces deux algorithmes, comme SRAC fait appel plusieurs fois à l'algorithme HP, et que ses temps de calculs dépendent du temps d'exécution de l'algorithme HP, nous analysons le rapport du temps de calcul de SRAC sur celui de HP.

Dans un premier temps, nous comparons ces deux algorithmes sur le modèle d'évolution homogène. Pour différents nombres d'Intervalles communs, nous calculons le rapport moyen des temps de calcul de SRAC sur celui de HP. Les résultats sont exprimés figure 6.11.

Le temps de calcul de HP ne dépend pas du nombre d'Intervalles communs et reste donc constant. Le temps de calcul de SRAC augmente avec le nombre d'Intervalles communs, pour stagner à partir de 15 Intervalles communs générés. Malgré la complexité exponentielle en temps dans le pire des cas de l'algorithme SRAC, on observe en pratique que celui-ci se comporte très bien. Cela est dû au grand nombre d'Intervalles communs positifs et négatifs.

Pour information, le temps réel de calcul de SRAC est en moyenne de 9 769 micro secondes pour 10 Intervalles communs générés. Le temps minimum est de 61 micro secondes, alors que le temps maximum qui représente le pire des cas rencontré pendant nos expérimentations est de 185 332 micro secondes. Ces calculs ont été effectués sur un PC cadencé à 3 GHz.

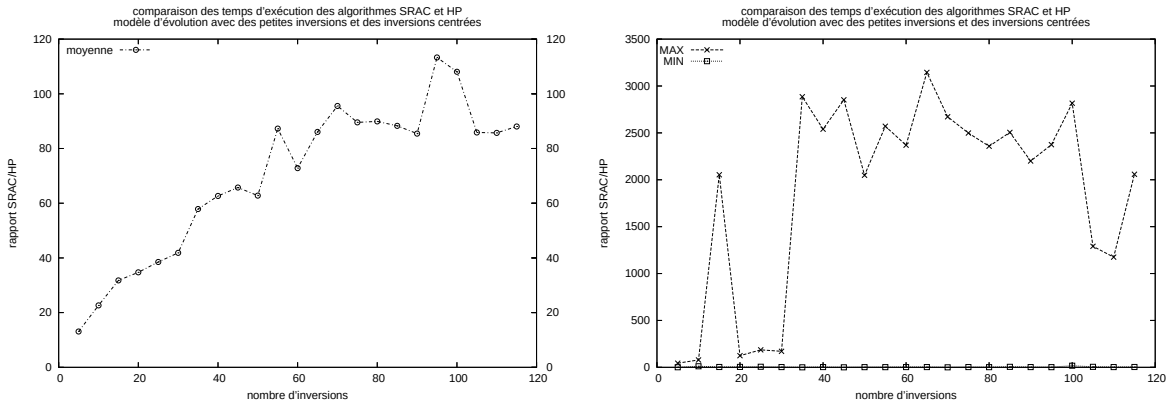
Nous comparons maintenant les deux algorithmes sur les jeux de données générés par le modèle d'évolution autorisant les inversions centrées et les inversions suivant la loi exponentielle p^{x-1}, avec $p = 0.5$, le tout dans un rapport de $\frac{1}{2}$. Nous ne générons aucun Intervalle commun, c'est l'évolution qui les crée. Nous avons choisi ce modèle d'évolution car non seulement c'est le plus réaliste mais c'est aussi lui qui donne le plus d'écarts entre les distances



(a) (b)

Figure 6.11: Rapport du temps d'exécution de l'algorithme SRAC et de l'algorithme HP. Les données ont été générées avec le modèle d'évolution homogène. A gauche la courbe représentant la moyenne suivant le nombre d'Intervalles communs générés. A droite, le minimum et le maximum du rapport du temps de calcul pour chaque nombre d'Intervalles communs générés.

calculées avec HP et celles calculées avec SRAC. La figure 6.12 donne le rapport des temps d'exécution entre HP et SRAC.



(a) (b)

Figure 6.12: Rapport du temps d'exécution de l'algorithme SRAC et de l'algorithme HP. Les données ont été générées avec le modèle d'évolution générant des petites inversions et des inversions centrées. A gauche la courbe représentant la moyenne pour chaque distance générée. A droite, le minimum et le maximum du rapport du temps de calcul pour chaque distance générée.

Le temps de calcul de HP ne dépend pas du nombre d'inversions et reste donc constant. Le temps de calcul de SRAC augmente avec la distance évolutive, pour stagner à partir de 80

inversions. Le maximum du temps de calcul de SRAC reste à peu près constant quelque soit la distance évolutive. Le temps de calcul de l'algorithme SRAC ne dépend pas non plus du nombre d'inversions, mais du nombre d'Intervalles communs. Et ce dernier dépend du nombre d'inversions ayant générées les permutations.

De même que dans l'expérimentation précédente, malgré la complexité exponentielle en temps dans le pire des cas de l'algorithme SRAC, on observe, en pratique, que celui-ci se comporte très bien. Cela est encore une fois dû au grand nombre d'Intervalles communs positifs et négatifs.

6.4 Discussion

Nous avons montré les possibilités et les limites de notre approche. Pour cela nous avons généré des génomes en simulant l'évolution. Nous avons proposé plusieurs modèles d'évolution. Le premier modèle, simpliste, applique un certain nombre d'inversions, toutes équiprobables, à un génome. Des Intervalles communs sont générés en fixant des intervalles pour lesquels les inversions doivent être conformes. Le deuxième modèle supprime la contrainte des intervalles. Le troisième modèle, le modèle d'évolution homogène, sélectionne, de même que le premier modèle, des intervalles mais applique à chacun d'eux un taux d'évolution fixe. Pour simplifier, on peut considérer qu'un des intervalles est la permutation elle-même. Le quatrième modèle d'évolution n'autorise, quant à lui, que les inversions centrées sur l'origine de réplication (simulée par un point quelconque de la permutation). Dans une deuxième version, nous autorisons en plus les inversions d'un seul gène. Le cinquième modèle d'évolution génère des inversions dont la taille suit une loi exponentielle. Finalement, le sixième et dernier modèle d'évolution, génère des inversions centrées ou non centrées mais alors leur taille suit une loi exponentielle dans ce dernier cas.

Nous pouvons séparer les résultats en trois catégories : celle pour laquelle nous calculons la distance conforme en connaissant les intervalles qui permettent de générer les Intervalles communs ; celle pour laquelle nous calculons la distance conforme après avoir calculé les Intervalles communs sans information préalable ; et finalement, celle pour laquelle nous n'avons pas explicitement généré d'Intervalle commun.

Pour la première catégorie, la distance conforme, bien que meilleure que la distance parcimonieuse simple, n'améliore pas de beaucoup les distances évolutives calculées. Pour chaque Intervalle commun, le nombre maximum d'inversions supplémentaires est de trois. Mais cela n'arrive que dans de rares cas. En moyenne, on n'observe pas de différence sensible entre les deux mesures évolutives. On peut noter une petite amélioration avec le modèle d'évolution homogène, qui permet de mieux mélanger les permutations générées.

Pour la deuxième catégorie, la distance conforme estime moins bien les distances évolutives que la distance parcimonieuse simple dans le cas de petites distances. Un petit nombre d'inversions génère des Intervalles communs qui ne sont pas dû à une contrainte et ne sont pas conservés par l'évolution. Mais surtout, nous avons montré comment deux inversions qui s'intersectent peuvent complètement biaiser le calcul de la distance conforme. Nous avons recalculé la distance conforme en éliminant les structures dues aux inversions qui s'intersectent et cela a permis de diminuer l'erreur donnée par la distance conforme. Mais dans la plupart des cas, la distance conforme reste un moins bon estimateur que la distance classique pour

des petites distances évolutives. Pour des distances évolutives plus importantes, la distance conforme estime mieux les distances évolutives que la distance parcimonieuse simple. On peut noter que dans le cas du modèle d'évolution homogène, la distance conforme est aussi bonne que la distance parcimonieuse pour un petit nombre d'inversions, et bien meilleure que celle-ci pour un plus grand nombre d'inversions.

Lorsque l'on ne génère pas expressément d'Intervalles communs, on remarque que curieusement, la distance conforme est un meilleur estimateur que la distance parcimonieuse pour un grand nombre d'inversions. En effet, il se crée des Intervalles communs et ceux-ci sont pris en compte par la distance conforme. Les expériences avec le modèle évolution centrée sont intéressantes. Il n'y a aucune différence entre les distances conformes et les distances parcimonieuses. Ce qui signifie que les Intervalles communs générés, en grand nombre, par les inversions centrées ne sont pas cassés par les scénarios évolutifs basés sur la distance parcimonieuse simple. Pour tous les autres modèles (inversions centrées et inversions d'un seul gène, petites inversions, etc) la distance conforme est un bon estimateur de l'évolution. Et plus particulièrement quand les distances évolutives sont importantes. La distance conforme donne de très bons résultats avec le modèle d'évolution générant des inversions centrées et des inversions dont la taille suit une loi exponentielle.

6.5 Résumé et conclusions

Les algorithmes proposés sont dans le pire des cas de complexité exponentielle en temps. Nous avons postulé dans le chapitre précédent qu'en pratique nos algorithmes ont un bon comportement. Les expérimentations de ce chapitre ont permis de le montrer. En effet, en comparant les temps de calculs de la distance parcimonieuse classique (de complexité linéaire en fonction de la taille des permutations) avec ceux de la distance conforme, le comportement en moyenne de cette dernière est sous-exponentiel. Et dans tous les cas rencontrés, le calcul de la distance conforme est largement praticable.

Nous avons montré dans quels cas la distance conforme est un meilleur estimateur de distance évolutive que la distance parcimonieuse. Par exemple, avec le modèle d'évolution homogène ou avec des modèles d'évolution basés sur l'observation des génomes connus. Nous avons aussi montré les cas limites où la distance conforme génère un taux d'erreur important. Par exemple, lorsque les Intervalles communs sont calculés et qu'il y a peu d'évolution. Les inversions qui s'intersectent crées des Intervalles communs ne représentant pas une structure conservée par l'évolution. Nous avons aussi montré comment éliminer ces structures pour améliorer la distance conforme dans cette configuration.

Bien que les résultats obtenus avec la distance conforme soient intéressants, il faut prendre en compte l'erreur absolue moyenne importante qu'elle génère. La distance parcimonieuse, elle aussi, génère une erreur absolue moyenne importante et qui croit de manière exponentielle avec le nombre d'inversions appliquées. Même si la distance conforme permet de diminuer cette erreur absolue, celle-ci reste relativement importante.

Nous noterons tout de même les très bons résultats obtenus par la distance conforme pour des données générés suivant des modèles d'évolutions construit à partir d'observations biologiques.

Conclusion

Avec le nombre grandissant d'espèces connues et leur séquençage à venir, les biologistes accumulent toujours plus de données. Sans des méthodes informatiques pour les comparer, ces données resteront sous-utilisées. Un des grands challenges des années à venir est la détection de structures conservées entre les génomes, dont font partis les clusters de gènes. On recherche des ensembles de gènes dont leur produit forme un complexe ou a une fonction apparentée. Un deuxième challenge important est la compréhension des évènements évolutifs survenus entre les espèces. Il s'agit de retracer les liens phylogénétiques des espèces. On essaye alors d'inférer un scénario évolutif à deux espèces représentées par leur information génétique.

La détection de clusters de gènes basée sur les réarrangements génomiques est un champ relativement nouveau de la bioinformatique. Les méthodes actuelles sont peu adaptées aux données réelles ou ne sont pas très rigoureuses. Soit elles simplifient trop leur représentation des génomes ; elles n'autorisent, par exemple, ni les duplications ni les insertions délétions ; mais surtout, les groupes de gènes détectés doivent être strictement présents dans tous les génomes comparés. Soit elles s'apparentent à une suite de filtres sans être bien définies ; alors qu'elles autorisent les duplications et d'autres particularités, qu'elles autorisent aussi les groupes des gènes à ne pas être présents dans tous les génomes, on ne peut avoir d'information sur la pertinence des objets détectés.

Notre point de vue a été d'améliorer les méthodes actuelles en nous efforçant de faire un pas vers les données biologiques réelles. Dans un premier temps, avec les clusters de gènes où notre méthode — HUGO — permet de prendre en compte, les gènes dupliqués, les gènes non communs à tous les génomes et les gènes insérés ou délétés ; mais aussi et surtout, notre méthode permet de détecter des clusters de gènes communs à un sous-ensemble des espèces comparées. HUGO est basé sur le concept des über-operons : des structures partiellement conservées dans les génomes de procaryotes. Nous nous sommes efforcés de mettre au point une méthode traitant des données réelles tout en évitant le piège des méthodes basées sur des filtres. Nous avons voulu garder un modèle mathématique simple. Les groupes de gènes que nous détectons sont facilement caractérisés et on peut d'ailleurs estimer leur probabilité d'apparition.

Les problèmes, liés aux méthodes actuelles qui consistent à inférer l'évolution à partir de l'ordre des gènes, sont du même ordre. Elles simplifient les données réelles et ne permettent pas de mesurer efficacement l'évolution. L'hypothèse sur laquelle repose principalement ces méthodes est le principe de parcimonie.

Nous avons analysé les scénarios évolutifs inférés et nous avons découvert qu'ils ne respectaient pas les groupes de gènes conservés. Les scénarios évolutifs calculés par les méthodes actuelles ne respectent pas les données réelles et par conséquent ne nous semblent pas vraisemblables. Le principe de parcimonie sur lequel ils sont basés n'est pas optimal. Pour palier à ces problèmes, nous avons proposé une nouvelle mesure de l'évolution : la distance conforme. La distance conforme permet de conserver les Intervalles communs, une caractérisation simple des structures conservées par l'évolution. Non seulement nous avons proposé cette nouvelle distance plus adapté aux modèles biologiques de l'évolution des espèces, mais nous avons aussi fait un pas important vers sa caractérisation. Nous avons calculé son diamètre, montré que ce problème appartenait à la classe des problèmes NP-complets, et proposé différents algorithmes pour le résoudre. Nous avons aussi montré que cette nouvelle distance mesure mieux l'évolution réelle, entre des génomes construits à partir de modèles biologiques d'évolution, que la distance parcimonieuse usuelle.

Nous sommes bien conscients que notre travail dans ces domaines n'est pas fini, le rapprochement entre les modèles mathématiques et les données biologiques est encore loin d'être parfait. Notre travail basé sur les über-operons, permet de palier aux limitations des *Connected Gene Neighborhoods* ou des *Gene Teams*, mais lui aussi peut être encore amélioré. Pour éviter que les über-operons soient des objets trop éparpillés dans les génomes, il serait intéressant de pouvoir préciser le nombre minimum de gènes qui doivent être réellement dans les mêmes opérons. De même, il faudrait pouvoir définir dans combien d'espèces au minimum ces gènes doivent être dans les mêmes opérons.

Nous avons fait le tour du problème de la distance conforme pour les Intervalles communs, mais ceux-ci ne sont pas les seules contraintes à ajouter aux scénarios parcimonieux. Ne serait-ce que parce qu'ils représentent une structure parfaitement conservée, c'est-à-dire sans insertion ni délétion de gènes. Une deuxième limitation des Intervalles communs est qu'ils doivent être exactement présents dans tous les génomes comparés. Bien sûr, il manque aussi la prise en compte des gènes dupliqués. Il serait intéressant de reformuler le problème de la distance conforme avec d'autres groupes de gènes conservés. L'idéal serait de contraindre les scénarios parcimonieux avec des groupes de gènes tels que les über-operons qui correspondent à une réalité biologique basée sur les opérons.

Annexe

Classification des Procaryotes

Bacteria	Proteobacteria	gamma	Escherichia coli K-12 MG1655	eco
			Escherichia coli K-12 W3110	ecj
			Escherichia coli O157 EDL933	ece
			Escherichia coli O157 Sakai	ecs
			Escherichia coli CFT073	ecc
			Salmonella typhi CT18	sty
			Salmonella typhi Ty2	stt
			Salmonella typhimurium LT2	stm
			Yersinia pestis CO92	ype
			Yersinia pestis KIM	ypk
			Yersinia pestis bv. Mediaevails	ypm
			Yersinia pseudotuberculosis	yps
			Shigella flexneri 301 (serotype 2a)	sfl
			Shigella flexneri 2457T (serotype 2a)	sfx
			Erwinia carotovora	eca
			Photorhabdus luminescens	plu
			Buchnera aphidicola APS	buc
			Buchnera aphidicola Sg	bas
			Buchnera aphidicola Bp	bab
			Wigglesworthia brevipalpis	wbr
			Blochmannia floridanus	bfl
			Haemophilus influenzae	hin
			Haemophilus ducreyi	hdu
			Pasteurella multocida	pmu
			Mannheimia succiniciproducens	msu
			Xylella fastidiosa 9a5c	xfa
			Xylella fastidiosa Temecula1	xft
			Xanthomonas campestris	xcc
			Xanthomonas axonopodis	xac
			Vibrio cholerae	vch
			Vibrio vulnificus CMCP6	vvu
			Vibrio vulnificus YJ016	vvy
			Vibrio parahaemolyticus	vpa
			Photobacterium profundum	ppr
			Pseudomonas aeruginosa	pae
			Pseudomonas putida	ppu
			Pseudomonas syringae pv. tomato	pst
			Acinetobacter sp. ADP1	aci
			Shewanella oneidensis	son
			Coxiella burnetii	cbu
			Legionella pneumophila	lpn
			Methylococcus capsulatus	mca
		beta	Neisseria meningitidis MC58 (serogroup B)	nme
			Neisseria meningitidis Z2491 (serogroup A)	nma
			Chromobacterium violaceum	cvi
			Ralstonia solanacearum	rso
			Burkholderia mallei	bma
			Burkholderia pseudomallei	bps
			Bordetella pertussis	bpe
			Bordetella parapertussis	bpa
			Bordetella bronchiseptica	bbr
			Nitrosomonas europaea	neu
		epsilon/delta	Helicobacter pylori 26695	hpy
			Helicobacter pylori J99	hpj
			Helicobacter hepaticus	hhe
			Wolinella succinogenes	wsu
			Campylobacter jejuni	cje
			Geobacter sulfurreducens	gsu
			Desulfovibrio vulgaris	dvu
			Bdellovibrio bacteriovorus	bba
			Desulfotalea psychrophila	dps
		alpha	Rickettsia prowazekii	rpr
			Rickettsia typhi	rty
			Rickettsia conorii	rco
			Wolbachia wMel	wol
			Mesorhizobium loti	mlo
			Sinorhizobium meliloti	sme
			Agrobacterium tumefaciens C58 (UWash/Dupont)	atu
			Agrobacterium tumefaciens C58 (Cereon)	atc
			Brucella melitensis	bme
			Brucella suis	bms
			Bradyrhizobium japonicum	bja
			Rhodopseudomonas palustris	rpa
			Bartonella henselae	bhe
			Bartonella quintana	bqu
			Caulobacter crescentus	ccr
	Firmicutes	Bacillales	Bacillus subtilis	bsu
			Bacillus halodurans	bha
			Bacillus anthracis Ames	ban
			Bacillus anthracis A2012	baa
			Bacillus anthracis Sterne	bat
			Bacillus cereus ATCC 14579	bce
			Bacillus cereus ATCC 10987	bca
			Bacillus cereus ZK	bcz
			Bacillus thuringiensis	bth
			Bacillus licheniformis ATCC 14580	bli
			Bacillus licheniformis DSM 13	bld
			Oceanobacillus iheyensis	oih
			Staphylococcus aureus N315 (MRSA)	sau
			Staphylococcus aureus Mu50 (VRSA)	sav
			Staphylococcus aureus MW2	sam
			Staphylococcus aureus MRSA252	sar
			Staphylococcus aureus MSSA476	sas
			Staphylococcus epidermidis	sep
			Listeria monocytogenes EGD-e (serotype 1/2a)	lmo
			Listeria monocytogenes F2365 (serotype 4b)	lmf
			Listeria innocua	lin

Bacteria	Firmicutes	Lactobacillales	Lactococcus lactis	lla
			Streptococcus pyogenes SF370 (serotype M1)	spy
			Streptococcus pyogenes MGAS8232 (serotype M18)	spm
			Streptococcus pyogenes MGAS315 (serotype M3)	spg
			Streptococcus pyogenes SSI-1 (serotype M3)	sps
			Streptococcus pyogenes MGAS10394 (serotype M6)	spa
			Streptococcus pneumoniae TIGR4	spn
			Streptococcus pneumoniae R6	spr
			Streptococcus agalactiae 2603 (serotype V)	sag
			Streptococcus agalactiae NEM316 (serotype III)	san
			Streptococcus mutans	smu
			Lactobacillus plantarum	lpl
			Lactobacillus johnsonii	ljo
			Enterococcus faecalis	efa
		Clostridia	Clostridium acetobutylicum	cac
			Clostridium perfringens	cpe
			Clostridium tetani	ctc
			Thermoanaerobacter tengcongensis	tte
		Mollicutes	Mycoplasma genitalium	mge
			Mycoplasma pneumoniae	mpn
			Mycoplasma pulmonis	mpu
			Mycoplasma penetrans	mpe
			Mycoplasma gallisepticum	mga
			Mycoplasma mycoides	mmy
			Mycoplasma mobile	mmo
			Ureaplasma urealyticum	uur
			Phytoplasma sp. onion yellows	poy
			Mesoplasma florum	mfl
	Actinobacteria		Mycobacterium tuberculosis H37Rv (lab strain)	mtu
			Mycobacterium tuberculosis CDC1551	mtc
			Mycobacterium bovis	mbo
			Mycobacterium leprae	mle
			Mycobacterium avium paratuberculosis	mpa
			Corynebacterium glutamicum	cgl
			Corynebacterium efficiens	cef
			Corynebacterium diphtheriae	cdi
			Streptomyces coelicolor	sco
			Streptomyces avermitilis	sma
			Tropheryma whipplei Twist	twh
			Tropheryma whipplei TW08/27	tws
			Leifsonia xyli xyli	lxx
			Propionibacterium acnes	pac
			Bifidobacterium longum	blo
			Symbiobacterium thermophilum	sth
	Fusobacteria		Fusobacterium nucleatum	fnu
	Planctomyces		Rhodopirellula baltica (Pirellula sp.)	rba
	Chlamydia		Chlamydia trachomatis	ctr
			Chlamydia muridarum	cmu
			Chlamydophila pneumoniae CWL029	cpn
			Chlamydophila pneumoniae AR39	cpa
			Chlamydophila pneumoniae J138	cpj
			Chlamydophila pneumoniae TW-183	cpt
			Chlamydophila caviae	cca
			Parachlamydia UWE25	pcu
	Spirochete		Borrelia burgdorferi	bbu
			Borrelia garinii	bga
			Treponema pallidum	tpa
			Treponema denticola	tde
			Leptospira interrogans 56601 (serover lai)	lil
			Leptospira interrogans Fiocruz L1-130 (serover Copenhageni)	lic
	Bacteroid		Bacteroides thetaiotaomicron	bth
			Bacteroides fragilis	bfr
			Porphyromonas gingivalis	pgi
	Cyanobacteria		Synechocystis sp. PCC6803	syn
			Synechococcus sp. WH8102	syw
			Synechococcus sp. PCC6301	syc
			Thermosynechococcus elongatus	tel
			Gloeobacter violaceus	gvi
			Anabaena sp. PCC7120 (Nostoc sp. PCC7120)	ana
			Prochlorococcus marinus SS120 (CCMP1375)	pma
			Prochlorococcus marinus MED4 (CCMP1378)	pmm
			Prochlorococcus marinus MIT9313	pmt
	Green sulfur bacteria		Chlorobium tepidum	cte
	Deinococcus-Thermus		Deinococcus radiodurans	dra
			Thermus thermophilus	tth
	Hyperthermophilic bacteria		Aquifex aeolicus	aae
			Thermotoga maritima	tma
Archaea	Euryarchaeota		Methanococcus jannaschii	mja
			Methanococcus maripaludis	mmp
			Methanosarcina acetivorans	mac
			Methanosarcina mazei	mma
			Methanobacterium thermoautotrophicum	mth
			Methanopyrus kandleri	mka
			Archaeoglobus fulgidus	afu
			Halobacterium sp. NRC-1	hal
			Thermoplasma acidophilum	tac
			Thermoplasma volcanium	tvo
			Picrophilus torridus	pto
			Pyrococcus horikoshii	pho
			Pyrococcus abyssi	pab
			Pyrococcus furiosus	pfu
	Crenarchaeota		Aeropyrum pernix	ape
			Sulfolobus solfataricus	sso
			Sulfolobus tokodaii	sto
			Pyrobaculum aerophilum	pai
	Nanoarchaeota		Nanoarchaeum equitans	neq

D'après KEGG : http ://www.genome.jp/kegg/kegg2.html

alpha	4	5	6	7	8	9	10	11	12
P(|C|>= 2) ≤	1	1	1	1	1	1	1	1	1
P(|C|>= 3) ≤	1	1	1	1	1	1	1	1	1
P(|C|>= 4) ≤	1	1	0.721006	0.0833104	0.0670142	0.0661456	0.0661013	0.0660992	0.0660991
P(|C|>= 5) ≤	1	1	0.0536325	0.0035739	0.00274966	0.00270761	0.0027055	0.0027054	0.0027054
P(|C|>= 6) ≤	1	1	0.00434351	0.000168452	0.000124108	0.000121931	0.000121823	0.000121818	0.000121818
P(|C|>= 7) ≤	1	1	0.000338985	8.1439e-06	5.79021e-06	5.67835e-06	5.67288e-06	5.67263e-06	5.67262e-06
P(|C|>= 8) ≤	1	1	2.7835e-05	4.14247e-07	2.84222e-07	2.78228e-07	2.77938e-07	2.77925e-07	2.77924e-07
P(|C|>= 9) ≤	1	1	2.23878e-06	2.13599e-08	1.41979e-08	1.38767e-08	1.38612e-08	1.38605e-08	1.38605e-08
P(|C|>=10) ≤	1	1	1.80285e-07	1.12135e-09	7.2341e-10	7.06005e-10	7.05175e-10	7.05139e-10	7.05138e-10
P(|C|>=11) ≤	1	1	1.46659e-08	5.99622e-11	3.75766e-11	3.66207e-11	3.65754e-11	3.65734e-11	3.65734e-11
P(|C|>=12) ≤	1	1	1.07098e-09	3.09016e-12	1.89436e-12	1.8443e-12	1.84194e-12	1.84184e-12	1.84184e-12
P(|C|>=13) ≤	1	0.722636	7.73779e-11	1.59926e-13	9.60362e-14	9.34099e-14	9.32871e-14	9.32818e-14	9.32816e-14
P(|C|>=14) ≤	1	0.302587	5.67235e-12	8.39785e-15	4.93987e-15	4.80025e-15	4.79376e-15	4.79348e-15	4.79347e-15
P(|C|>=15) ≤	1	0.122157	4.09637e-13	4.40788e-16	2.54313e-16	2.4691e-16	2.46567e-16	2.46553e-16	2.46552e-16
P(|C|>=16) ≤	1	0.0462849	2.87128e-14	2.29367e-17	1.30032e-17	1.26148e-17	1.2597e-17	1.25962e-17	1.25962e-17
P(|C|>=17) ≤	1	0.0177048	2.0318e-15	1.20493e-18	6.71214e-19	6.50662e-19	6.4972e-19	6.49681e-19	6.49679e-19
P(|C|>=18) ≤	1	0.00587084	1.3391e-16	6.13576e-20	3.36934e-20	3.26418e-20	3.25938e-20	3.25918e-20	3.25917e-20
P(|C|>=19) ≤	1	0.00172439	8.33766e-18	3.04392e-21	1.65154e-21	1.5992e-21	1.59682e-21	1.59672e-21	1.59671e-21
P(|C|>=20) ≤	1	0.000509882	5.22607e-19	1.52019e-22	8.1495e-23	7.88732e-23	7.87544e-23	7.87494e-23	7.87492e-23
P(|C|>=21) ≤	1	0.000151674	3.29545e-20	7.63784e-24	4.04558e-24	3.91349e-24	3.90752e-24	3.90727e-24	3.90726e-24
P(|C|>=22) ≤	1	4.30574e-05	2.03722e-21	3.80615e-25	1.99359e-25	1.92762e-25	1.92465e-25	1.92452e-25	1.92452e-25
P(|C|>=23) ≤	1	1.19655e-05	1.24984e-22	1.89305e-26	9.80902e-27	9.48029e-27	9.46554e-27	9.46492e-27	9.4649e-27
P(|C|>=24) ≤	1	3.16848e-06	7.51307e-24	9.32821e-28	4.78526e-28	4.62304e-28	4.61578e-28	4.61548e-28	4.61547e-28
P(|C|>=25) ≤	1	7.98889e-07	4.42455e-25	4.55167e-29	2.31332e-29	2.23408e-29	2.23054e-29	2.2304e-29	2.23039e-29
P(|C|>=26) ≤	1	1.86578e-07	2.52182e-26	2.18276e-30	1.10017e-30	1.06215e-30	1.06046e-30	1.06039e-30	1.06038e-30
P(|C|>=27) ≤	1	4.37255e-08	1.44232e-27	1.05037e-31	5.25036e-32	5.06726e-32	5.05913e-32	5.05879e-32	5.05878e-32
P(|C|>=28) ≤	1	1.028e-08	8.27547e-29	5.07064e-33	2.51363e-33	2.42519e-33	2.42127e-33	2.42111e-33	2.42111e-33
P(|C|>=29) ≤	1	2.29658e-09	4.64916e-30	2.42034e-34	1.19062e-34	1.14839e-34	1.14653e-34	1.14645e-34	1.14645e-34

Estimation de la probabilité d'oberver une composante connexe de taille au moins k

Index

Bibliographie

[Ajana et al., 2002] Ajana, Y., Lefebvre, J., Tillier, E., and El-Mabrouk, N. (2002). Exploring the set of all minimal sequences of reversals - an application to test the replication-directed reversal hypothesis. In *Lecture Notes in Computer Science*. Proceedings of WABI 2004.

[Bader et al., 2000] Bader, D., Moret, B., and Yan, M. (2000). A linear time algorithm for computing inversion distance between signed permutations with an expermimental study. *Journal of Computational Biology*.

[Bader et al., 2001] Bader, D. A., Moret, M., Warnow, T., Wyman, S., and M.Yan (2001). High-performance algorithm engineering for gene-order phylogenies. *DIMACS Workshop on Whole Genome Comparison*.

[Bafna and Pevzner, 1995] Bafna, V. and Pevzner, P. (1995). Sorting permutations by transpositions. In *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 614–621.

[Beigel, 1999] Beigel, R. (1999). Finding maximum independent sets in sparse and general graphs. *Proc. ACM-SIAM SODA*.

[Bergeron, 2001] Bergeron, A. (2001). A very elementary presentation of the hannenhalli-pevzner theory. *CPM*.

[Bergeron, 2003] Bergeron, A. (2003). Similarity vs. distance in whole genome comparison. *Mathematics of Evolution and Phylogeny Workshop*.

[Bergeron et al., 2004] Bergeron, A., Blanchette, M., Chateau, A., and Chauve, C. (2004). Reconstructing ancestral gene orders using conserved intervals. In *Lecture Notes in Computer Science*. Proceedings of WABI 2004.

[Bergeron et al., 2002a] Bergeron, A., Chauve, C., Hartman, T., and St-Onge, K. (2002a). On the properties of sequences of reversals that sort a signed permutation. In *Proceedings of JOBIM*, JOBIM.

[Bergeron et al., 2002b] Bergeron, A., Corteel, S., and Raffinot, M. (2002b). The algorithmic of gene teams. *Bioinformatics*.

[Berman and Hannenhalli, 1996] Berman, P. and Hannenhalli, S. (1996). Fast sorting by reversals. In *LNCS-Proceeding of the 7th annual Symposium on Combinatorial Pattern Matching*, pages 168–185. Springer-Verlag.

[Berman et al., 2002] Berman, P., Hannenhalli, S., and Karpinski, M. (2002). 1.375-approximation algorithm for sorting by reversals. *ESA*.

[Caprara, 1997] Caprara, A. (1997). Sorting by reversals is difficult. *ACM RECOMB*.

[Caprara, 1999] Caprara, A. (1999). Formulations and hardness of multiple sorting by reversals. In *RECOMB*.

[Christie, 1998] Christie, D. (1998). *Genome Rearrangement Problems*. PhD thesis, University of Glasgow.

[Chung and Lu, 2002] Chung, F. and Lu, L. (2002). Connected components in random graphs with given expected degree sequences. *Annals of Combinatorics*, 6 :125–145.

[Cosner et al., 2000] Cosner, M., Moret, R. J. B. M. E., Raubeson, L., Wang, L., Warnow, T., and Wyman, S. (2000). An empirical comparison between bpanalysis and mpbe on the campanulaceae chloroplast genome dataset. *Comparative Genomics*.

[Dalevi et al., 2002] Dalevi, D. A., Eriksen, N., Eriksson, K., and Andersson, S. G. (2002). Measuring genome divergence in bacteria : A case study using chlamydian data. *Journal of Molecular Evolution*, 55 :24–36.

[DasGupta et al., 1998] DasGupta, B., Jiang, T., Kannan, S., Li, M., and Sweedyk, Z. (1998). On the complexity and approximation of syntenic distance. *Discrete Applied Mathematics*.

[Dias and Meidanis, 2001] Dias, Z. and Meidanis, J. (2001). Genome rearrangements distance by fusion, fission, and transposition is easy. *SPIRE*.

[Dias and Meidanis, 2002] Dias, Z. and Meidanis, J. (2002). The genome rearrangements distance problem using fusion, fission, and transposition with arbitrary weights.

[Durand and Sankoff, 2003] Durand, D. and Sankoff, D. (2003). Tests for gene clustering. *Journal of Computational Biology*, 10 :453–482.

[Eisen et al., 2000] Eisen, Heidelberg, White, and Salzberg (2000). Evidence for symmetric chromosomal inversions around the replication origin in bacteria. *Genome Biology*.

[El-Mabrouk, 2000] El-Mabrouk, N. (2000). Sorting signed permutations by reversals and insertions/deletions of contiguous segments. *Journal of Discrete Algorithms*.

[Eriksen, 2001] Eriksen, N. (2001). (1+epsilon)-approximation of sorting by reversals and transpositions. *WABI*.

[Ermolaeva et al., 2001] Ermolaeva, M. D., White, O., and Salzberg, S. L. (2001). Prediction of operons in microbial genomes. *Nucleic Acids Research*.

[Figeac et al., 2003] Figeac, M., Varré, J., and Delahaye, J. (2003). Détection de structures conservées dans les génomes de procaryotes. *Action Spécifique : ASIM*.

[Figeac and Varré, 2004] Figeac, M. and Varré, J.-S. (2004). Sorting by reversals and common intervals. In *Lecture Notes in Computer Science*. Proceedings of WABI 2004.

[Garey and Jonhson, 1979] Garey, M. and Jonhson, D. (1979). Computers and intractability : A guide to the theory of np-completeness. *Freeman*.

[Gevers et al., 2004] Gevers, D., Vandepoele, K., Simillion, C., and de Peer, Y. V. (2004). Gene duplication and biased functional retention of paralogs in bacterial genomes. *Trends Microbiol*.

[Gu et al., 1999] Gu, Q.-P., Peng, S., and Sudborough, H. (1999). 1/2 approximation algorithms for genome rearrangement by reversals and transpositins.

[Hannenhalli, 1995] Hannenhalli, S. (1995). Polynomial-time algorithm for computing translocation distance between genomes. *CPM*.

[Hannenhalli and Pevzner, 1996] Hannenhalli, S. and Pevzner, P. (1996). To cut ... or not to cut (applications of comparative physical maps in molecular evolution. *Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 304–313.

[Hannenhalli and Pevzner, 1995] Hannenhalli, S. and Pevzner, P. A. (1995). Transforming cabbage into turnip : polynomial algorithm for sorting signed permutations by reversals. *STOC*.

[Hartman and Shamir, 2003] Hartman, T. and Shamir, R. (2003). A simpler 1.5-approximation algorithm for sorting by transpositions. In *CPM*.

[Hartman and Sharan, 2004] Hartman, T. and Sharan, R. (2004). A 1.5-approximation algorithm for sorting by transpositions and transreversals. In *Lecture Notes in Computer Science*. Proceedings of WABI 2004.

[Heber and Stoye, 2001a] Heber, S. and Stoye, J. (2001a). Algorithms for finding gene clusters. *Lecture Notes in Computer Science*, 2149 :252.

[Heber and Stoye, 2001b] Heber, S. and Stoye, J. (2001b). Finding all common intervals of k permutations. *Lecture Notes in Computer Science*, 2089 :207–??

[Hoon et al., 2004] Hoon, M. D., Imoto, S., Kobayashi, K., Ogasawara, N., and Miyano, S. (2004). Predicting the operon structure of bacillus subtilis using operon length, intergene distance, and gene expression information. *Pac Symp Biocomput*.

[Kaplan et al., 2000] Kaplan, H., Shamir, R., and Tarjan, R. E. (2000). Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal on Computing*.

[Kaplan and Verbin, 2003] Kaplan, H. and Verbin, E. (2003). Efficient data structures and a new randomized approach for sorting signed permutations by reversals. In *CPM*.

[Kececioglu and Sankoff, 1993] Kececioglu, J. and Sankoff, D. (1993). Exact and approximation algorithms for the inversion distance between two chromosomes. *Lecture Notes in Computer Science 684*, pages 87–105.

[Kececioglu and Sankoff, 1995] Kececioglu, J. and Sankoff, D. (1995). Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13 :180–210.

[Lathe et al., 2000] Lathe, W. C., Snel, B., and Bork, P. (2000). Gene context conservation of higher order than operons. *Trends in Biochemical Sciences*.

[Lawrence and Roth, 1996] Lawrence, J. and Roth, J. (1996). Selfish operons : Horizontal transfert may drive the evolution of gene clusters. *Genetics*, 143 :1843–1860.

[Lefebvre et al., 2003] Lefebvre, J., El-Mabrouk, N., Tilier, E., and Sankoff, D. (2003). Detection and validation of single gene inversions. *bioinformatics*.

[Lewin, 1987] Lewin, B. (1987). *Genes*. John Wiley and Sons, inc.

[Liben-Novel, 2001] Liben-Novel, D. (2001). On the structure of syntenic distance. *Journal of computational biology*.

[Luc et al., 2003] Luc, N., Risler, J.-L., Bergeron, A., and Raffinot, M. (2003). Gene teams : A new formalization of gene clusters for comparative genomics. *Computational Biology and Chemistry*.

[Mackiewicz et al., 2001] Mackiewicz, P., Mackiewicz, D., Kowalczuk, M., and Cebrat, S. (2001). Flip-flop around the origin and terminus of replication in prokaryotic genomes. *Genome Biology*.

[Maki et al., 2000] Maki, N., Gestwicki, J. E., Lake, E. M., Kiessling, L. L., and Adler, J. (2000). Motility and chemotaxis of filamentous cells of escherichia coli. *Journal of Bacteriology*.

[Marron et al., 2003] Marron, M., Svenson, K. M., and Moret, B. M. E. (2003). Genomic distances under deletions and insertions. *xxx*.

[Meidanis et al., 2000] Meidanis, J., Walter, M., and Dias, Z. (2000). Reversal distance of signed circular chromosomes.

[Moreno-Hagelsieb and Collado-Vides, 2002] Moreno-Hagelsieb, G. and Collado-Vides, J. (2002). A powerful non-homology method for the prediction of operons in prokaryotes. *Bioinformatics*, 18.

[Moret et al., 2001] Moret, B. M. E., Wang, L., Warnow, T., and Wyman, S. (2001). New approaches for reconstructing phylogenies from gene order data. *ISMB*.

[Ogata et al., 1999] Ogata, Goto, Sato, Fujibuchi, Bono, and Kanehisa (1999). Kegg : Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*.

[Overbeek et al., 1999] Overbeek, R., Fonstein, M., D'Souza, M., Pusch, G. D., and Maltsev, N. (1999). The use of gene clusters to infer functional coupling. *Proc. Natl. Acad. Sci. USA*.

[Ozery-Flato and Shamir, 2003] Ozery-Flato, M. and Shamir, R. (2003). Transforming men into mice revisited (a correction to the theory of sorting by reversals and transpositions). *Third Haifa Workshop on Interdisciplinary Applications of Graph Theory, Combinatorics and Computing*.

[Pellegrini et al., 1999] Pellegrini, M., Marcotte, E. M., Thompson, M. J., Eisenberg, D., and Yeates, T. O. (1999). Assigning protein functions by comparative genome analysis : Protein phylogenetic profiles. *Proc Natl Acad Sci U S A.*, 96 :4285–4288.

[Rogozin et al., 2002] Rogozin, Makarova, Murvai, Czabarka, Wolf, Tatusov, Szekely, and Koonin (2002). Connected gene neighborhoods in prokaryotic genomes. *Nucleic Acids Research*.

[Salgado et al., 2000] Salgado, Moreno-Hagelsieb, Smith, and Collado-Vides (2000). Operons in escherichia coli : genomic analyses and predictions. *Pnas*.

[Salgado et al., 2004] Salgado, H., Gama-Castro, S., Martínez-Antonio, A., Díaz-Peredo, E., Sánchez-Solano, F., Peralta-Gil, M., Garcia-Alonso, D., Jiménez-Jacinto, V., Santos-Zavaleta, A., Bonavides-Martínez, C., and Collado-Vides, J. (2004). Regulondb (version 4.0) : transcriptional regulation, operon organization and growth conditions in escherichia coli k-12. *Nucleic Acids Research*. http ://www.cifn.unam.mx/Computational_Genomics/regulondb/.

[Sankoff, 1999] Sankoff, D. (1999). Genome rearrangement with gene families. *Bioinformatics*, 15(11) :909–917.

[Sankoff, 2002] Sankoff, D. (2002). Short inversions and conserved gene clusters. *Bioinformatics*, 18(10) :1305–1308.

[Sankoff and Blanchette, 1997] Sankoff, D. and Blanchette, M. (1997). The median problem for breakpoints in comparative genomics. *Lecture Notes in Computer Science*.

[Sankoff and Blanchette, 1998] Sankoff, D. and Blanchette, M. (1998). Multiple genome rearrangement and breakpoint phylogeny. *Computational Biology*, 5(3) :555–570.

[Setubal and Meidanis, 1997] Setubal and Meidanis (1997). *Introduction to Computational Molecular Biology*. PWS publishing.

[Shrager, 2003] Shrager, J. (2003). The fiction of function. *bioinformatics*.

[Siepel, 2002] Siepel, A. (2002). An algorithm to find all sorting reversals. In *RECOMB'02*, pages 281–290. ACM Press.

[Siepel and Moret, 2001] Siepel, A. C. and Moret, M. (2001). Finding an optimal inversion median : Experimental results. *Computer Science*.

[Singer and Berg, 1991] Singer, M. and Berg, P. (1991). *Genes And Genomes*. University Science Books.

[Tatusov et al., 1997] Tatusov, R. L., Koonin, E. V., and Lipman, D. J. (1997). A genomic perspective on protein families. *Science*.

[Terai et al., 2001] Terai, G., Tagagi, T., and Nakai, K. (2001). Prediction of co-regulated genes in bacillus subtilis based on the conserved upstream element across three closely related species. *Genome Informatics*.

[Tillier and Collins, 2000] Tillier and Collins (2000). Genome rearrangement by replication-directed translocation. *Nature*.

[Uno and Yagiura, 2000] Uno, T. and Yagiura, M. (2000). Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*.

[Virtanen, 2003] Virtanen, S. (2003). Clustering the chilean web. *Proceedings of the First Latin American Web Congress (LA-WEB 2003)*.

[Walter et al., 1998] Walter, M., Dias, Z., and Meidanis, J. (1998). Reversal and transposition distance of linear chromosomes. *SPIRE*.

[Wanner, 1996] Wanner, B. L. (1996). Phosphorus assimilation and control of the phosphate regulon.

[Yagiura et al., 1994] Yagiura, M., Nagamochi, H., and Ibaraki, T. (1994). Two comments on the subtour exchange crossover operator. *Technical report of IEICE*, 88 :1–10.

[Zheng et al., 2002] Zheng, Y., Szustakowski, J. D., Fortnow, L., Roberts, R. J., and Kasif, S. (2002). Computational identification of operons in microbial genomes. *Genome Research*, 12(8).