Aspects algorithmiques des réarrangements génomiques : duplications et ordres partiels

Annelyse Thévenin

Thèse

de

L'Université Paris–Sud

présentée en vue de l'obtention du

Doctorat de l'Université Paris–Sud

Spécialité : Informatique

Par

Annelyse THÉVENIN

Aspects algorithmiques des réarrangements génomiques : duplications et ordres partiels

Soutenue le 6 novembre 2009 devant la commission d'examen :

Mme	Marie-France Sagot	Directrice de Recherche	Rapportrice
M.	Bernard Moret	Professeur	Rapporteur
M.	Philippe Dague	Professeur	Examinateur
Mme	Sylvie Hamel	Professeure agrégée	Examinatrice
M.	Alain Denise	Professeur	Directeur de thèse
M.	Stéphane Vialette	Chargé de Recherche	Directeur de thèse

Laboratoire de Recherche en Informatique, U.M.R. CNRS 8623,
Université Paris-Sud, 91405 Orsay Cedex, France

Numéro d'ordre : 9611

Remerciements

Les mots servent à exprimer les idées ; quand l'idée est saisie, oubliez les mots.
(Tchouang-Tseu).

Je tiens tout abord à remercier Stéphane Vialette, mon encadrant scientifique depuis mon stage de fin d'étude.

On m'a demandé tout à l'heure depuis quand je voulais travailler en génomique comparative... J'ai répondu que mon choix ne s'était pas porté sur le sujet mais que je voulais simplement travailler avec Stéphane ! Bon je vous rassure ce sujet me plait aussi ! Mais quand vous travaillez avec quelqu'un de passionné, avec une capacité à résoudre des problèmes de malade l'air de rien, pédagogue, calme face à n'importe quelle situation, sincère et qui de plus vous fait voyager à travers le monde, bref un mec comme Stéphane, alors le sujet de thèse devient secondaire. Au cours de trois années, diverses péripéthies ont rendu nos réunions difficiles à mettre en place, mais lorsque nous réussissions à nous voir j'ai pu apprécier chacune de ses qualités.

Mes remerciements se tournent également vers Alain Denise, directeur de ma thèse et qui m'a encadré à diverses occasions depuis mes débuts en Bioinformatique. Comme tout scientifique il est débordé ! Mais il a un super pouvoir : vous avez besoin de le voir, pas de problème il arrive à trouver le temps qu'il faut pour écouter, faire le point, donner un conseil ou simplement encourager. Et puis il a aussi un autre super pouvoir, les mauvaises langues diront que c'est une faiblesse mais n'en croyez rien : lorsqu'il vient de vous faire une remarque que vous avez du mal à digèrer (genre : là c'est nul, il va falloir réécrire toute cette section...je résume...) son super pouvoir lui permet de trouver une dizaine de qualités sur votre travail et de vous les dire ! Et là vous repartez réécrire votre section le sourire aux lèvres !

Stéphane et Alain ont cette passion pour la recherche qui fait qu'avec eux ce métier devient une grande fête foraine remplie de jolies problèmes à résoudre. Et pour cela je vous remercie tous les deux : cette thèse est une aventure que je ne regretterai pas.

Par ailleurs, je tiens à remercier l'ensemble de mon jury pour avoir accepter de juger ma thèse, mais aussi pour ses conseils et sa bonne humeur. Plus particulièrement, je souhaite remercier Marie-France Sagot et Bernard Moret pour avoir rapporter ma thèse. Je remercie également Sylvie Hamel d'être venue d'aussi loin pour examiner ma thèse mais aussi pour son accueil enthousiaste lors de mon séjour au DIRO. Merci à Philippe Dague d'avoir présider ce jury.

J'ai passé les trois années de ma thèse au sein de l'équipe Bioinfo du LRI. Je voudrais remercier l'ensemble des membres de l'équipe, ancien ou nouveau, pour la bonne ambiance

qui y régne.

Plus particulièrement et dans le désordre, je remercie : Bastien Rance pour avoir rédigé en même temps que moi et ce dans la bonne humeur ! ; Claire Herrbach pour nos nombreuses discutions en diverses langues... ; Claire Toffano pour son calme, ses conseils, ses relectures de manuscrit ; Christine Froidevaux pour nos pauses thé après des journées de fous, qui étaient une parenthèse agréable et qui m'aidaient à faire le point ; Fred-Éric Lemoine qui est toujours prêt à donner un coup de main ; Jérôme Azé pour m'avoir faire réver à la construction d'une maison de A à Z, on jouera à Resident Evil dedans si un jour tu la construit ? ! ; Les nouveaux thèsards Zahira Azlaoui et Philippe Rinaudo pour leur enthousiasme de jeunes doctorants ; Lucie Gentils pour nos discussions autour du lac, pour l'énergie et l'aide que tu apportais à l'équipe ; Mes co-bureaux divers et variés : même si tous n'ont pas survécu, merci d'avoir accepter le mode *grotte* ; notons qu'au final ce mode commence à avoir des afficianados ! ! ; Patrick Amar pour les nombreuses discussions que nous avons eu, en particulier lorsque que j'avais des choix importants à faire ; Thomas Moncion pour le soutien qu'il m'a apporté l'année dernière ; Tout ceux qui ont su jouer à la balle bleu dans notre chère salle de café...Peti'Tom, Damien, etc ; Sarah Cohen pour nos discutions très très sportives :) qui m'ont aidé à voir les choses d'un autre oeil, merci aussi pour sa patience et tous les conseils qu'elle transmet autour d'elle ; Yann Ponty qui m'a laissé le haïr.

Au final, je remercie tout ceux de l'équipe qui m'ont soutenue d'une façon ou d'une autre dans cette aventure qui a été loin d'être facile !

J'ai aussi eu l'occasion de passer quelques journées au LIGM à l'Université de Marne la Vallée. Je souhaite remercier particulièrement le trio Julien/Florian/Omar, toujours de bonne humeur malgrès les aléas de la thèse ! Merci à Julien de m'avoir permis de faire une de mes meilleures farces, certe douloureuse mais tu l'avouera unique ! Merci à Florian pour nos discutions sur nos malheurs de thésards... :) Merci à Omar pour les soirées à ALEA où tu m'as appris des tours de magie, que nous avons au final tous les deux oubliés... Merci à tous ce que j'ai croisé à Marne avec un sourire aux lèvres !

Je n'aurai pas pu me lancer dans une telle aventure sans mes trois pilliers, indispensables dans tout ce que j'entreprend : mes Amis, ma Famille, le Scoutisme.

À tous qui ont cherché à savoir qui je suis vraiment.

Résumé

En français

La génomique comparative est une discipline importante pour la compréhension de l'évolution du vivant. Différentes méthodes de comparaison de génomes existent, nous nous intéressons ici en particulier au calcul de 3 mesures de (dis)similarités : le nombre d'adjacences, le nombre de points de cassure et le nombre d'intervalles communs. En présence de gènes dupliqués ou lorsque l'ordre des gènes n'est que partiellement connu, calculer ces mesures est un problème connu pour être NP-difficile.
Premièrement, nous désirons calculer les nombres d'adjacences et de points de cassures pour trois modèles (exemplaire, maximum et le modèle intermédiaire introduit dans cette étude) entre deux génomes possédant des duplications. Dans le but d'obtenir des résultats exacts, nous utilisons dans cette thèse une approche par programmation pseudo-booléenne. Après expérimentation sur 12 génomes de γ-protéobactéries, nous obtenons assez de résultats pour évaluer les différentes combinaisons mesure/modèle. De plus, nous proposons et évaluons (à l'aide des précédents résultats) une famille d'heuristiques basée sur une recherche de plus longue sous-séquence commune qui donne de très bons résultats sur ces données.
Parallèlement à cela, nous prouvons que pour différents problèmes de calcul de mesures entre deux génomes avec duplication, aucune approximation en temps polynomial n'est possible (sauf si $\mathbf{P} = \mathbf{NP}$).
Deuxièmement, nous mettons en place une approche pseudo-booléenne pour calculer les nombres d'adjacences et d'intervalles communs entre deux génomes partiellement ordonnées. À l'aide de près de 800 génomes simulés, nous étudions l'influence de paramètres inhérents aux ordres partiels et nous comparons les deux mesures étudiées.

In english

Comparative genomics aims to better understand differences between species. Several methods for genome comparison exist ; in this PhD, we have focused on the computation of three measures of (dis)similarities, namely the number of adjacencies, the number of breakpoints, and the number of commons intervals. In presence of duplicated genes or when the order of genes is only partially known, computing these measures is a NP-hard problem. Our contribution is twofold.
First, we want to compute the number of adjacencies and the number of breakpoints for three models (exemplar, maximum and the intermediate model introduced in this work) between two genomes with duplications. In order to obtain exact results, we have used a pseudo-boolean programming approach. After a test on 12 genomes of γ-proteobacteria, we got enough results to compare different combinations of measure/model. Additionally, we have proposed and evaluated (thanks to the above-mentioned results) a family of heuristics based on a search of a longest common subsequence, which gave very good results on these data.
In parallel, we proved that there exists no approximation algorithm (unless $\mathbf{P} = \mathbf{NP}$) to compute the number of adjacencies (on the exemplar model) and the number of breakpoints (on the exemplar and intermediate models).
Second, we set up a pseudo-boolean approach to compute the number of adjacencies and the number of common intervals between two partially ordered genomes. Using nearly 800 simulated genomes, we have studied the influence of parameters associated to partial orders and compared both measures.

Table des matières

Introduction

Depuis leur apparition sur terre, il y a plus de 3,5 milliards d'années, les êtres vivants ne cessent de se multiplier et de se diversifier. À partir d'un ancêtre commun, ils ont évolué au point de devenir des êtres aussi divers que la méduse *Aequorea Victoria* fluorescente, la *Balaenoptera musculus* pouvant peser 170 tonnes, le *Hypsibius dujardini* capable de vivre à des températures extrêmes (de -272°C à 150°C), la *Streptomyce ambofacien* utilisé dans la production d'antibiotique ou encore l'*Homo sapiens*.

Pourtant, malgré toute cette diversité, les êtres vivants utilisent les mêmes molécules pour se développer et fonctionner : l'ADN, l'ARN et les protéines.

L'ADN constitue le support de l'information génétique, autrement dit le génome. Cette information permet à un être de se développer et de fonctionner et sera transmise, en partie, à ses descendants. Le génome est matérialisé par des chromosomes qui sont eux même composés d'une séquence nucléotidique.

Les gènes sont des sous-séquences de chromosomes. Ils sont transcrits en ARN. Puis, certains de ces ARN (les ARN codants) sont traduits en protéine. Au final, ce sont les protéines et les ARN non codant qui font fonctionner les cellules dont sont constituées les êtres vivants. Chaque protéine et chaque ARN non codant a une (ou plusieurs) fonction(s) bien particulière(s) dans la cellule. Une légère modification dans la séquence de nucléotides dont ils sont originaires peut les rendre inactifs ou leur apporter une nouvelle fonction. Si cette nouvelle fonction n'est pas délétère pour l'être vivant, elle va alors permettre à son espèce d'évoluer.

Le génome est donc un élément indispensable chez tous les organismes. C'est pourquoi, il est fondamental de l'étudier afin de mieux comprendre le fonctionnement et l'évolution des êtres vivants. Dans ce manuscrit, nous nous focalisons sur la compréhension de l'évolution.

Pour cela, nous devons d'abord obtenir la séquence de nucléotides dont sont composés les génomes. Différentes méthodes de séquençage permettent d'obtenir de plus en plus efficacement ces séquences. Aujourd'hui, les banques de données regroupent plusieurs milliers de génomes sous forme de séquence de nucléotides. Puis, une étape d'annotation permet de trouver les gènes dont sont composés les génomes. Notons que certains gènes

sont en plusieurs exemplaires dans un génome : ce sont des gènes dupliqués.

Grâce à ces deux étapes, nous pouvons connaître la composition des génomes en nucléotides et en gènes. Nous avons alors la possibilité, en observant un unique génome, d'étudier ses différentes caractéristiques : la composition en nucléotides de ses gènes, son taux d'ADN codant et d'ADN non codant, la relation entre les gènes consécutifs et leurs produits (protéines et ARN non codants), etc.

Dans ce manuscrit, nous travaillons avec des génomes qui sont sous la forme de séquences de gènes (et non de suite de nucléotides). Plus précisément, nous comparons ces génomes, deux à deux, afin de mettre en avant les points communs et les différences dans leurs séquences de gènes. En particulier, nous étudions l'ordre des gènes qui peut être très ou peu conservés au cours de l'évolution.

Historiquement, la première stratégie avancée consiste à calculer un scénario permettant de passer d'un génome à un autre. Plus précisément, le but est de trouver la plus courte suite d'évènements, tels que des insertions ou des déplacements de gènes, permettant d'obtenir un génome à partir d'un autre. Le nombre d'évènements comptabilisés correspond à une distance de réarrangements. Cette distance est entièrement dépendante des opérations autorisées. Il faut donc autoriser suffisamment d'opérations pour être réaliste mais pas en excès pour pouvoir développer des algorithmes.

Une seconde stratégie repose sur le calcul de mesures de similarité ou de dissimilarité entre deux génomes. Par exemple, nous comptabilisons le nombre de paires de gènes consécutifs simultanément sur les deux génomes : cette mesure de similarité se nomme le nombre d'adjacences. Il existe d'autres mesures telles que le nombre de points de cassure, le nombre d'intervalles communs, le nombre d'intervalles conservés, MAD (nombre Maximum de Perturbations d'Adjacences) et SAD (Somme des Perturbations d'Adjacences).

Dans un premier temps, ces deux stratégies étaient appliquées uniquement entre deux génomes pour lesquels un seul gène par famille était pris en compte. Dans ce cas, il est plus facile de comparer les génomes, et ainsi, nous pouvons étudier, d'un point de vue biologique, la pertinence des mesures proposées. De plus, nous avons supposé que nous connaissions entièrement l'ordre des gènes. Malheureusement, les techniques de séquençage et d'annotations des génomes ne nous permettent pas toujours d'obtenir un ordre total.

Dans ce manuscrit, nous nous intéressons à la génomique comparative dans les cas où nous supprimons une de ces deux hypothèses (génomes sans duplication et totalement ordonné).

Tout d'abord, nous présentons, dans le Chapitre 1, les travaux déjà effectués dans le domaine de la génomique comparative.

Afin de comprendre les outils utilisés dans les chapitres suivants, le Chapitre 2 explicite les notions de base dans le domaine des graphes, des classes de complexité et de la programmation linéaire.

Le Chapitre 3 expose nos travaux dans le domaine de la comparaison de génomes en présence de gènes dupliqués. Les comparaisons sont alors biologiquement plus pertinentes mais le calcul de distances de réarrangements ou de mesures de (dis)similiarité devient alors beaucoup plus complexe. En effet, afin de pouvoir calculer ces distances/mesures, il est nécessaire de désambiguïser les gènes dupliqués. Pour ce faire, un couplage entre les deux génomes doit être élaboré. Différents modèles de couplages existent et pour deux d'entre eux (modèle exemplaire et modèle maximum), des heuristiques ont déjà été proposées.

Afin d'évaluer ces heuristiques, il est indispensable d'obtenir des résultats exacts. C'est pourquoi il faut trouver un algorithme, performant en pratique, permettant de calculer de façon exacte des mesures entre deux génomes entièrement ordonnés, possédant des gènes dupliqués. Ainsi, une banque de résultats est constituée permettant de mettre en place des heuristiques efficaces, mais aussi d'étudier avec justesse les modèles et les mesures proposés.

Concrètement, dans ce Chapitre 3, nous étudions la complexité des problèmes consistant à décider si le nombre minimal de points de cassure, entre deux génomes avec duplication, est nul ou non. Cette étude nous permet d'affiner la complexité de problèmes plus généraux. Puis, nous proposons des algorithmes permettant de calculer deux mesures de (dis)similarités entre deux génomes possédant des gènes dupliqués : le nombre d'adjacences et le nombre de points de cassures. Plus précisément, nous proposons des algorithmes calculant ces mesures suivant trois modèles de couplages : exemplaire, intermédiaire et maximum. Ces algorithmes sont de deux types : des algorithmes exacts, basés sur la modélisation du problème en programme linéaire à variables booléennes, et des heuristiques, basées sur la recherche de plus Longues sous-Séquences Communes (LCS). L'ensemble de ces algorithmes a fait l'objet d'une expérimentation basée sur la comparaison de 12 génomes de γ-protéobactéries.

Le Chapitre 4 présente nos travaux dans le domaine de la comparaison de génomes partiellement ordonnés. Il est important de noter que nous ne prenons alors plus en compte les duplications de gènes. Les génomes sont maintenant représentés par des ordres partiels. La comparaison des génomes va permettre d'obtenir une distance de réarrangements ou une mesure de (dis)similarité mais aussi un génome dont l'ordre des gènes sera maintenant total. Plus précisément, une comparaison entre un génome partiellement ordonné et un

génome proche et totalement ordonné est effectuée. Un ordre total respectant l'ordre partiel étudié est alors recherché de telle façon qu'une mesure donnée soit optimale entre cet ordre total et le génome proche et totalement ordonné.

Concrètement, dans ce Chapitre 4, nous utilisons deux mesures de similarités : le nombre d'intervalles communs et le nombre d'adjacences. D'une part, nous calculons un ordre total pour un génome partiellement ordonné tel que le nombre d'intervalles communs entre cet ordre total et un génome totalement ordonné soit maximal. D'autre part et de façon similaire, nous calculons un ordre total tel que, cette fois-ci, le nombre d'adjacences soit maximal. Finalement, à partir de deux génomes partiellement ordonnés, nous calculons deux ordres totaux tels que le nombre d'adjacences entre les deux soit maximal. Pour chacun de ces trois problèmes, un algorithme exact, basé sur une modélisation en programme linéaire à variables booléennes, est présenté. Il est évalué à l'aide de génomes partiellement ordonnés simulés.

L'ensemble de ces travaux s'est effectué en collaboration avec Sébastien Angibaud de l'Université de Nantes et nos encadrants respectifs (Guillaume Fertin, Irena Rusu et Stéphane Vialette). Pour ma part, je me suis penchée plus particulièrement sur les algorithmes exacts.

Chapitre 1

Comparaison génomique

Sommaire

Ce chapitre introduit des notions de biologie, d'évolution et de comparaison génomique, importantes pour comprendre les problématiques étudiées dans ce manuscrit. De plus, il présente les travaux déjà effectués dans le domaine de la génomique comparative.

Chaque organisme vivant possède des informations spécifiques lui permettant de se développer et de fonctionner. Cette information est appelée *génome* et est transmise

aux descendants (voir Section 1.1). Au vue de l'importance des génomes pour chaque organisme, mieux les connaître permet clairement de mieux comprendre les êtres vivants. En effet, nous pouvons, *via* l'étude des génomes, étudier l'évolution des espèces en vue de construire des arbres phylogéniques, étudier les régulations de groupes de gènes, étudier le lien entre des gènes physiquement proches et leurs produits respectifs, observer des régions de gènes conservées ou au contraire des régions qui ont subi des évolutions rapides. Pour cela, il faut tout d'abord séquencer les génomes afin de pouvoir annoter les gènes qui les composent. Ceci nous permet d'étudier les génomes sous forme de séquences de gènes. Une partie des recherches effectuées alors s'appuie sur la génomique comparative (voir Section 1.2).

Plus précisément, deux méthodes sont proposées pour comparer des génomes : calculer une distance de réarrangements ou calculer une mesure de (dis)similarité. Historiquement, les études effectuées sur des génomes ne prenaient pas en compte les gènes dupliqués (c'est-à-dire les gènes présents en plusieurs exemplaires dans un génome) et supposaient toujours que l'ordre des gènes était totalement connu. Depuis une dizaine d'années, des études sont effectuées en supprimant une de ces deux hypothèses (voir Sections 1.3 et 1.4, respectivement).

1.1 La génomique

1.1.1 Le génome

La génomique est la science qui étudie les génomes. Dans un souci de clarté, nous allons tout d'abord exposer une définition générale d'un génome ; nous donnerons par la suite quelques précisions.

Le *génome* d'un être vivant est l'ensemble de son matériel génétique. Il contient les informations permettant son développement et son fonctionnement et il est transmis, en partie, aux générations suivantes permettant ainsi l'hérédité. Il est situé dans un ou plusieurs chromosomes situés dans les cellules dont sont composés les organismes. Plus précisément, le stockage se fait au niveau de doubles hélices d'ADN (Acide DésoxyriboNucléique) - principal composant des chromosomes.

L'ADN est constitué de nucléotides non aléatoirement répartis sur des brins. Deux brins forment une double hélice. Un nucléotide est composé d'un groupement phosphate (ou acide phosphorique), d'un sucre à 5 atomes de carbone désoxyribose et d'une base azotée : une Adénine (`A`), une Cytosine (`C`), une Guanine (`G`) ou une Thymine (`T`).

La taille du génome peut varier de quelques kilo-bases chez les virus à plusieurs centaines de milliers de méga-bases chez certains eucaryotes (c'est-à-dire des êtres constitués de cellules possédant un noyau).

La Figure 1.1 représente schématiquement une double hélice constituée de deux brins d'ADN. Nous pouvons remarquer que les deux brins sont complémentaires au niveau de leurs bases : une guanine est associée à une cytosine ("≡"), une adénine est associée à une thymine ("=") et *vice versa*. Cette association spécifique est appelée *hybridation* moléculaire.

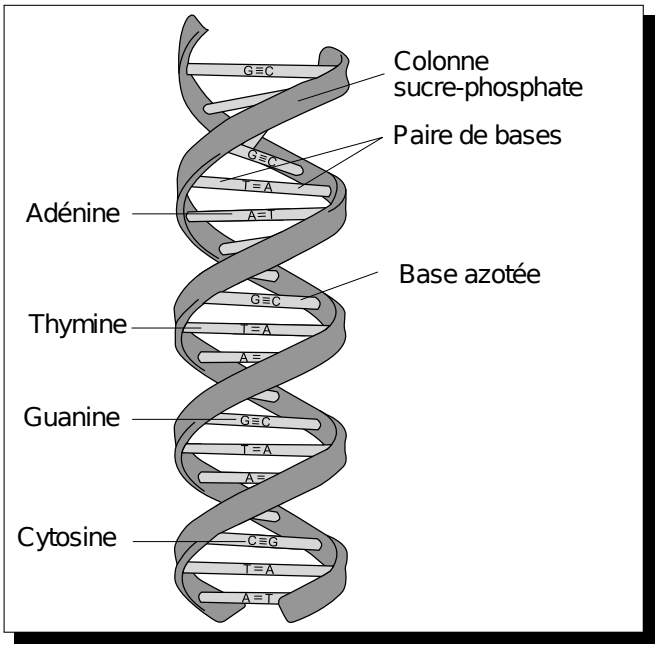


FIGURE 1.1 – **Structure schématique d'un double brin d'ADN**

Un génome, à une échelle plus grande, peut être aussi vu comme un ensemble de *gènes*. Un gène est composé d'un ensemble spécifique de nucléotides consécutifs le long d'un brin. Il est transcrit en *ARN* (Acide RiboNucléique) : les bases `A`, `C`, `G` et `T` d'un gène sont transcrites respectivement en bases `U` (Uracile), `G`, `C` et `A` pour former un ARN. Certains ARN (les ARN codants) sont ensuite traduits en *protéines* constituées d'acides aminés (voir Figure 1.2). Cette traduction respecte un code : 3 bases consécutives (appelées *codon*) sont traduites en un acide aminé spécifique (voir Tableau 1.1). L'ensemble des codons possibles (64) est supérieur au nombre d'acides aminés disponibles (20 chez les eucaryotes), si bien qu'un même acide aminé peut être codé par plusieurs codons : si le tryptophane n'est codé que par le codon `UGG`, la glycine peut être codée par les codons `GGU`, `GGC`, `GGA` ou `GGG`. Nous disons alors que le code est *dégénéré*.

	U		C		A		G		
U	UUU	phénylalanine	UCU	sérine	UAU	tyrosine	UGU	cystéine	U
	UUC		UCC		UAC		UGC		C
	UUA	leucine	UCA		UAA	*stop*	UGA	*stop*/sélénocystéine	A
	UUG		UCG		UAG		UGG	tryptophane	G
C	CUU		CCU	proline	CAU	histidine	CGU	arginine	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	glutamine	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	isoleucine	ACU	thréonine	AAU	asparagine	AGU	sérine	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	lysine	AGA	arginine	A
	AUG	méthionine/*start*	ACG		AAG		AGG		G
G	GUU	valine	GCU	alanine	GAU	acideaspartique	GGU	glycine	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	acideglutamique	GGA		A
	GUG		GCG		GAG		GGG		G

TABLE 1.1 – **Les 64 codons de la table du code génétique.** Le codon `ACC` code pour la thréonine et le codon `UGG` code pour la tryptophane. Le codon `AUG` code le premier acide aminé d'une protéine (la méthionine) ; les codons `UAA`, `UAG` et `UGA` codent pour l'arrêt de la synthèse de la protéine.

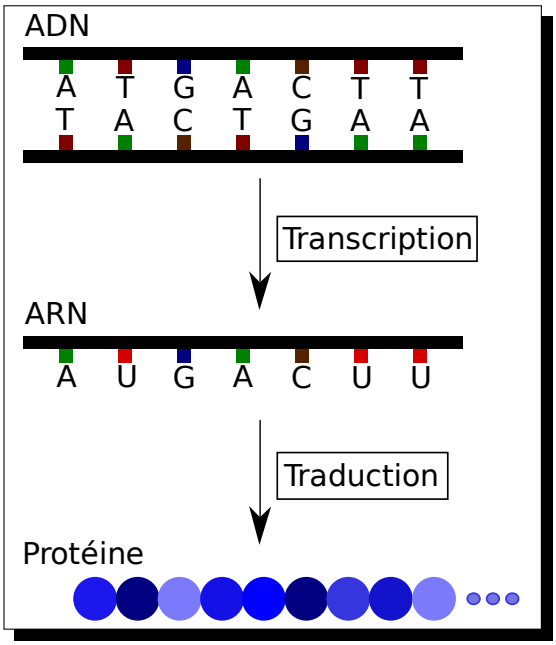


FIGURE 1.2 – **Dogme de la biologie moléculaire.** Un gène situé sur un des deux brins d'une hélice d'ADN est transcrit en ARN. Puis ce dernier peut-être traduit en protéine.

Les protéines sont des molécules indispensables au fonctionnement des cellules ; elles ont des fonctions de catalyse[1], de transport, de communication, de signalisation, de reconnaissance, de structure et de maintien de l'intégrité et de la transmission à la descendance du génome. Notons qu'une modification au niveau d'un gène (changement d'un nucléotide par exemple) peut avoir des conséquences au niveau des ARN et des protéines (à la dégénérescence du code près) et a donc des conséquences sur le fonctionnement des cellules.

Pour résumer, nous avons donc le génome (matériel génétique) stockant des gènes, linéairement répartis sur des couples de brin d'ADN, qui sont transcrits en ARN, eux-même traduits en protéines, "ouvriers" des cellules.

À cette définition générale des précisions s'imposent. Le génome se situe dans différentes partie de la cellule (voir l'encart page 19 pour une illustration). Pour les cellules d'eucaryotes, l'ADN se situe principalement dans le noyau mais pas uniquement. En effet, une petite partie de l'ADN peut être stockée dans les mitochondries ainsi que dans les chloroplastes. Dans les cellules procaryotes (c'est-à-dire sans noyau), l'ADN est contenu dans le cytoplasme sous forme d'ADN chromosomique et d'ADN plasmidique. Nous notons par ailleurs que certains chromosomes sont linéaires (pour la plupart des plantes et des animaux) alors que d'autres sont circulaires (pour la plupart des bactéries).

De plus, beaucoup d'ARN ne sont pas traduits et sont alors eux-aussi des "ouvriers" de la cellule (les ARN de transfert et les ARN ribosomiques par exemple). Finalement, la traduction des codons peut être légèrement différente chez les mitochondries et chez certaines bactéries.

1. Modification de la vitesse d'une réaction.

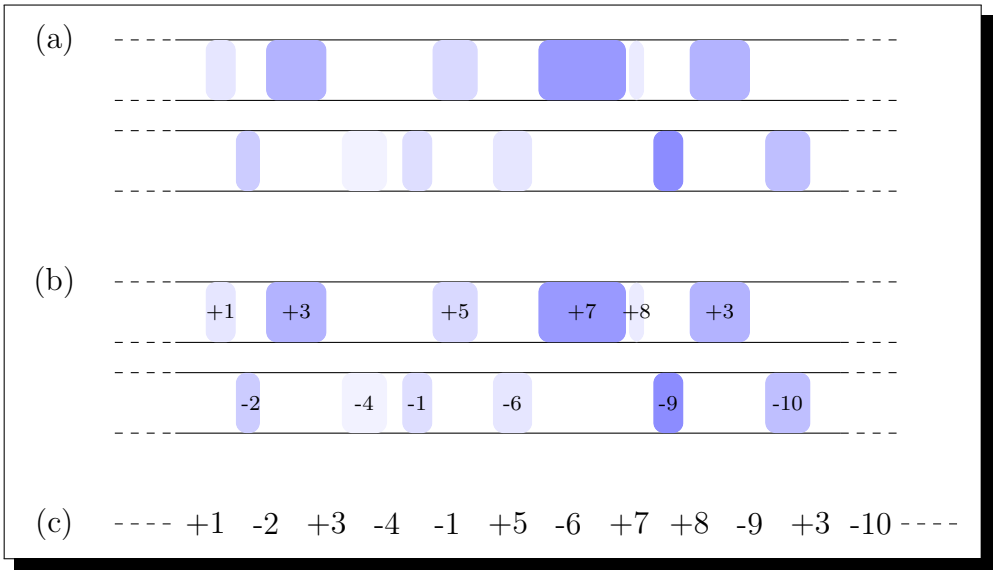


FIGURE 1.3 – **Représentation d'une séquence de gènes. (a)** Un génome est composé de gènes alignés sur deux brins complémentaires. **(b)** Un gène est représenté par un entier signé. Deux gènes ont le même signe s'ils sont situés sur le même brin. Deux gènes homologues (voir Section 1.1.2) sont représentés par le même entier (au signe près). **(c)** Finalement, le génome est représenté par une unique séquence d'entiers, qui ne prend pas en compte le nombre de nucléotides séparant deux gènes.

Avant de poursuivre, nous posons quelques notations. Un génome sera représenté comme la séquence des gènes d'un unique double brin d'ADN linéaire (voir Figure 1.3). Nous étudions donc uniquement les génomes composés d'une seule hélice d'ADN.

Un **gène** sera, dans la suite, représenté uniquement par un entier. Chaque gène présent sur un génome a un **signe** représentant le brin sur lequel il est situé : "+" ou "-".
Si 0 est un gène alors nous pouvons avoir $+0$ et -0 présent sur des génomes (ce sont des gènes homologues - voir Section 1.1.2 - situés sur deux brins différents). Il faut alors voir $+0$ et -0 comme des labels et non comme des entiers.

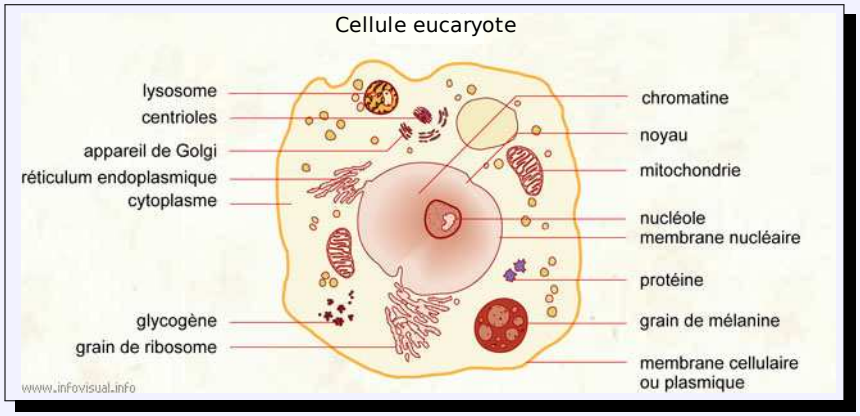
Un **génome** est représenté par une séquence d'entiers.
Dans le reste de ce manuscrit, nous étudions toujours deux génomes G_1 et G_2. Pour tout $x \in \{1; 2\}$, nous notons n_x la taille de G_x, c'est-à-dire le nombre de gènes appartenant à G_x.
La séquence **identité** de taille n, $+1\ +2 \ldots\ +n$, représente le génome nommé Id.

`Exemple` : Le génome illustré par la Figure 1.3 (a) est répresenté par la séquence +1 -2 +3 -4 -1 +5 -6 +7 +8 -9 +3 -10. Nous avons n_x égal à 12.

Une cellule

Une cellule est la brique de base des êtres vivants. Elle est composée de différents éléments dont :

- le **noyau**, présent uniquement chez les êtres eucaryotes, qui est le lieu de la transcription de l'ADN en ARN. Il contient une partie importante du génome.
- les **mitochondries** permettent de créer et de stocker de l'énergie à partir de molécules organiques.
- les **chloroplastes**, présents uniquement dans les cellules végétales, sont le siège de la photosynthèse. Ils absorbent l'énergie lumineuse pour la transformer en énergie chimique sous forme d'ATP.
- les **plasmides** sont des molécules d'ADN présent, en général, dans des cellules de procaryotes capables de réplication autonome.
- le **réticulum endoplasmique** est le lieu de maturation de certaines protéines.
- l'**appareil de Golgi** synthétise la plupart des sécrétions cellulaires.
- les **lysosomes** sont chargées, chez les cellules animales, de la digestion des molécules complexes.



1.1.2 Les homologies

Pour comprendre les problèmatiques qui nous intéressent ici, nous devons nous intéresser à l'*évolution*. Cette dernière désigne l'ensemble des transformations des êtres vivants conduisant à la diversité actuelle des espèces. Elle peut être représentée par un *arbre phylogénétique* où chaque feuille représente une espèce et chaque branche une relation inter-espèces. À chaque embranchement se trouve l'ancêtre commun d'un ensemble d'espèces.

L'évolution est la conséquence de deux phénomènes : les *mutations* et la *sélection naturelle*. Les mutations apportent des modifications dans les génomes induisant des transformations au sein d'une espèce. La sélection naturelle va conduire à la propagation ou la disparition d'une modification. Par exemple, une mutation favorisant la reproduction va encourager la propagation de la population mutée. Au final, l'évolution peut conduire

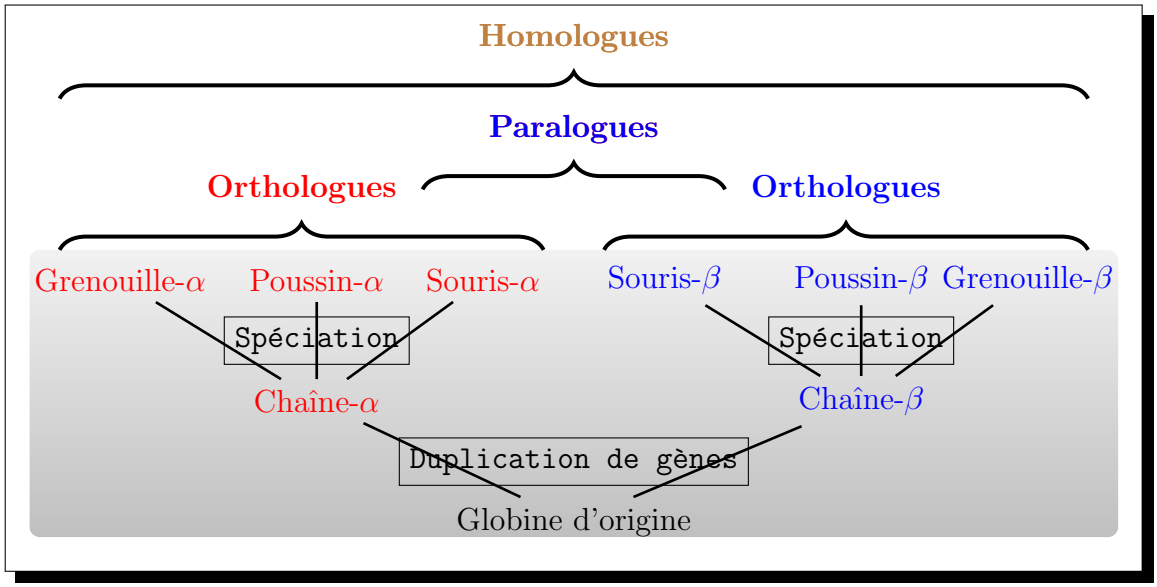


FIGURE 1.4 – **Illustration des notions d'homologie, d'orthologie et de paralogie** *via* un exemple simplifié de l'arbre phylogénétique des gènes codant pour les globines. Le gène globine d'origine a subit une duplication (mutation) ; ce qui fit apparaître un gène α et un gène β dans un même génome. Ces deux gènes ont évolué indépendamment (spéciation) et sont tous les deux présents chez la grenouille, le poussin et la souris.

à une différenciation entre deux populations appartenant à une même espèce au point, parfois, d'obtenir une nouvelle espèce. Ce phénomène de différenciation se nomme la *spéciation*.

L'évolution conduit au phénomène de l'homologie. Deux gènes sont dits *homologues* si nous pouvons supposer qu'ils sont les héritiers d'un même gène ancestral. Les *orthologues* et les *paralogues* distinguent deux types de séquences homologues. L'orthologie décrit la relation entre des gènes de différentes espèces qui dérivent d'un même ancêtre commun après une spéciation. La paralogie décrit la relation entre des gènes d'une même espèce qui dérivent d'une *duplication* de gène (voir ci-dessous). La Figure 1.4 illustre ces trois définitions.

Nous parlerons tout particulièrement des gènes paralogues dans le Chapitre 3.

Les mutations

Une mutation est une modification définitive dans la séquence du génome. Lors de la copie du génome ou de son exposition à des agents mutagènes (tels que des radiations, des virus ou des agents chimiques), le génome peut subir des modifications. Malgré l'existence de mécanismes complexes et efficaces de réparation, certaines mutations ont lieu.

Les mutations ponctuelles ne modifient qu'un seul nucléotide. Ce sont les mutations les plus courantes.
- La *substitution* d'un nucléotide d'un gène modifie un codon et donc potentiellement un acide aminé de la protéine correspondante. La protéine modifiée peut rester fonctionnelle.
- L'*insertion* ou la *délétion* d'un nucléotide d'un gène modifie l'ensemble des codons suivant la mutation et donc l'ensemble des acides aminés de la protéine correspondante. Il est fort probable que la nouvelle protéine ne puisse plus assurer sa fonction initiale du fait du décalage imposé dans la lecture des codons.

Les mutations chromosomiques modifient un segment chromosomique, affectant ainsi plusieurs dizaines de gènes voir plusieurs centaines (voir Figure 1.5).
- Une *transposition* correspond au déplacement d'un segment d'un site de chromosome à un autre site (pas obligatoirement du même chromosome).
- Une *translocation* correspond à une opération échangeant deux segments terminaux entre deux chromosomes linéaires.
- Une *inversion* correspond à un renversement d'un segment de chromosome.
- Une *délétion* correspond à la perte d'un segment de chromosome.
- Une *duplication* correspond au dédoublement d'un segment de chromosome.

Les mutations génomiques ont lieu lorsqu'une configuration chromosomique entière est dupliquée, ou bien lorsque les chromosomes de deux génomes différents se fondent dans la même cellule. Ces phénomènes sont rares dans la nature. Ils sont en général désastreux pour un organisme, car ils bouleversent le fragile équilibre des fonctions de milliers de gènes.

Les mutations dynamiques correspondent à une modification du nombre de répétitions de certains codons qui ont pour effet de diminuer ou de supprimer l'expression de gènes. Le nombre de répétitions évolue d'une génération à l'autre.

Les mutations sont les moteurs de l'évolution. En particulier, elles permettent d'obtenir plusieurs exemplaires d'un même gène (des gènes paralogues) qui vont évoluer indépendamment les uns des autres. D'autre part, les mutations les moins favorables à la survie (mutations délétères) sont éliminées *via* la sélection naturelle. Les plus avantageuses ont tendance à être conservées apportant ainsi potentiellement de nouvelles fonctionnalités. Les fonctions initiales sont maintenues par une des copies du génome pendant que les fonctions de gènes paralogues sont modifiées. Finalement, les mutations permettent l'apparition de nouvelles espèces.

Les familles multigéniques

Un gène peut être présent en plusieurs exemplaires sur un génome suite à des duplications. Nous disons alors que, dans ce génome, ce gène est *dupliqué*. L'ensemble des

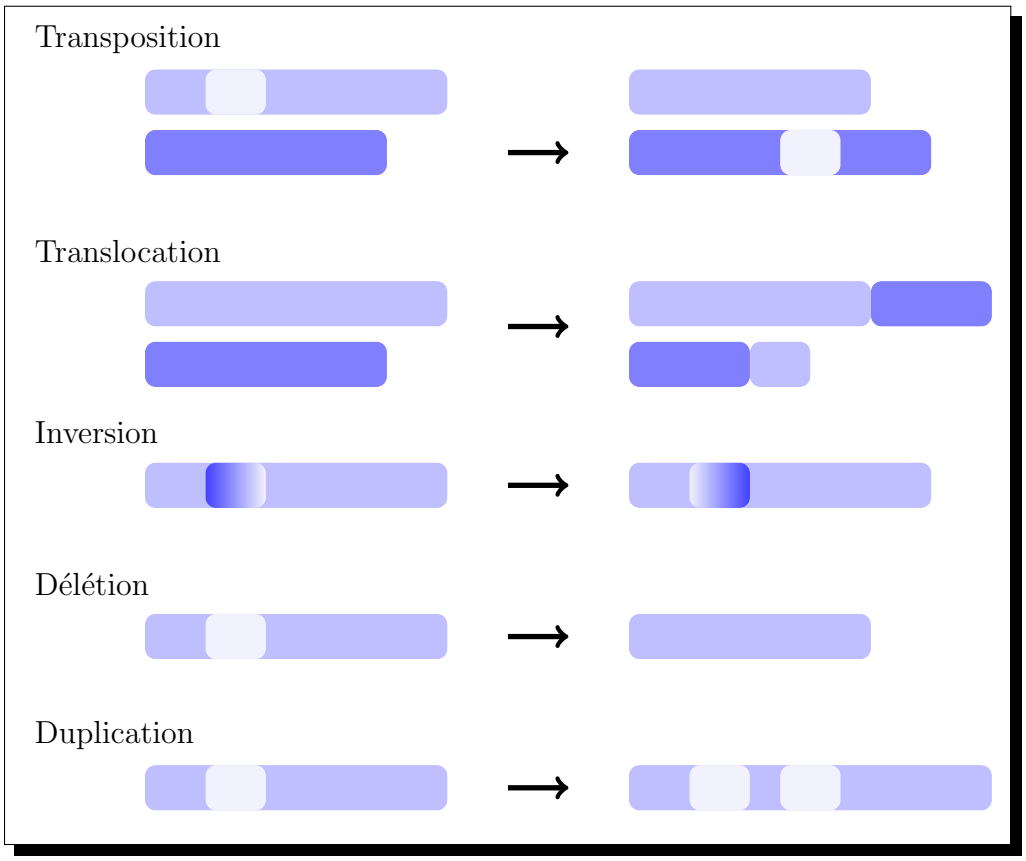


FIGURE 1.5 – **Mutations chromosomiques.** Une transposition, une translocation, une inversion, une délétion et une duplication chromosomique.

exemplaires d'un gène, dans l'ensemble des génomes, constitue une *famille multigénique*. Les gènes d'une même famille sont homologues. La taille de ces familles peut atteindre plusieurs centaines mais les petites familles (taille inférieure à 6) sont majoritaires.

Deux gènes homologues ont souvent quelques différences au niveau de leur séquence de nucléotides. Plusieurs difficultés se posent alors pour définir une famille multigénique.

- Il faut définir de bons critères de similarité permettant de décider si deux gènes sont homologues (séquences de nucléotides, structure et fonctions du produit du gène, etc).
- Il est nécessaire de fixer un seuil pour chacun des critères de similarité choisis : deux gènes sont homologues s'ils sont en-dessous (au-dessus selon le critère) de ce seuil.
- Des critères de similarité entre deux gènes ne suffisent pas. Par exemple, ces derniers ne permettent pas de décider si les gènes g_1, g_2 et g_3 forment une famille multigénique lorsque les paires de gènes (g_1, g_2) et (g_2, g_3) respectent les critères de similarité mais que la paire (g_1, g_3) non.

De nombreuses études permettent de définir ces familles [63, 83, 8]. Pour notre cas, nous supposons toujours que les familles multigéniques sont données, c'est-à-dire que les homologies de gènes sont définies.

Pour tout $x \in \{1; 2\}$, une **famille multigénique** $\mathtt{f}$ (notée par un entier naturel) dans le génome G_x regroupe tous les gènes $+\mathtt{f}$ et $-\mathtt{f}$ présents dans le génome G_x. Par exemple, pour un génome donné, la famille composée des gènes $\{+2, +2, -2, +2, -2\}$ est notée 2. Par abus de langage, nous utilisons la notation de valeur absolue pour noter la **famille d'un gène** : la famille d'un gène g est notée $|\mathsf{g}|$. L'**ensemble des familles multigéniques** du génome G_x est noté $\mathcal{F}_x$.

Le gène à la **position** i sur le génome G_x est noté $G_x[i]$. Le **signe** du gène g dans le génome G_x est noté $s_x(\mathsf{g})$. Deux gènes g_1 et g_2 sont **consécutifs** sur G_x si et seulement s'il existe $i \in [1 : n_x]$, tel que $G_x[i] = \mathsf{g}_1$ et soit $G_x[i+1] = \mathsf{g}_2$ (seulement si $i \neq n_x$) soit $G_x[i-1] = \mathsf{g}_2$ (seulement si $i \neq 1$).

L'ensemble des gènes localisés entre les gènes g_1 et g_2 inclus, sur le genome G_x, est appelé un **intervalle** et est noté $G_x[\mathsf{g}_1, \mathsf{g}_2]$. Les gènes g_1 et g_2 sont les **extrémités** de cet intervalle. L'ordre des gènes d'un intervalle est quelconque.

Dans le génome G_x, la **sous-séquence** $\mathsf{g}_1\ \mathsf{g}_2\ \ldots\ \mathsf{g}_p$ ($p \in [1 : n_x]$) est notée $[\mathsf{g}_1, \mathsf{g}_p]_x$. L'**inversion signée** d'une sous-séquence correspond à cette même sous-séquence dont les gènes ont des signes opposés et dont l'ordre des gènes est inversé.

Nous notons $\mathtt{occ}_x(\mathtt{f}, i, j)$ le nombre d'**occurrences** de gènes appartenant à la famille $\mathtt{f}$ (donc indépendamment du signe des gènes) dans G_x entre les positions i et j incluses, avec $1 \leq i < j \leq n_x$. Pour simplifier les notations, nous abrégeons $\mathtt{occ}_x(\mathtt{f}, 1, n_x)$ par $\mathtt{occ}_x(\mathtt{f})$. Nous posons $\mathtt{occ}_x = max\{\mathtt{occ}_x(\mathtt{f}) \mid \mathtt{f} \in \mathcal{F}_{G_x}\}$.

Un gène g est appelé i-**singleton**$_x$ si $G_x[i] = \mathsf{g}$ et $\mathtt{occ}_x(|\mathsf{g}|) = 1$.

`Exemple` : Soit G_1 = +1 -2 +3 -4 -1 +5 -6 +7 -8 +3 -9 un génome avec des duplications avec $n_1 = 11$; $x = 1$. La famille multigénique $\{+1; -1\}$ est notée 1. Nous avons $\mathcal{F}_1 = \{1; 2; 3; 4; 5; 6; 7; 8; 9\}$. Les gènes $G_1[2]$ et $G_1[6]$ sont respectivement -2 et $+5$. Les gènes $G_1[5]$ et $G_1[6]$ sont consécutifs. Lorsqu'il n'y a pas d'ambiguïté, nous notons plus simplement les gènes par leur label : -1 et $+5$ sont consécutifs. L'intervalle $G_1[-4, -8]$ est l'ensemble de familles de gènes $\{1; 4; 5; 6; 7; 8\}$. L'inversion signée de la sous-séquence $[+1, +5]_1$ (c'est-à-dire +1 -2 +3 -4 -1 +5) est -5 +1 +4 -3 +2 -1.
De plus, $\mathtt{occ}_1(1, 3, 6) = 1$, $\mathtt{occ}_1(1) = 2$ et $\mathtt{occ}_1 = 2$.
Finalement, les gènes -2, -4, +5, -6, +7, -8 et -9 sont des i-singletons$_1$ avec, respectivement, i égal à 2, 4, 6, 7, 8, 9 et 11.

1.1.3 Le séquençage et l'annotation des génomes

Nous avons vu comment nous représentons les génomes informatiquement. Mais afin d'obtenir la composition des génomes en nucléotides et en gènes, il est tout d'abord nécessaire de les séquencer puis de les annoter.

Les méthodes de séquençages

Le séquençage d'un génome permet de connaître sa séquence en nucléotides.

Le séquençage d'ADN a débuté dans les années 70. Depuis, différentes techniques ont été mises en place dans le but d'accélérer le séquençage et de permettre de séquencer de grands génomes [58]. Historiquement, les deux premières techniques de séquençage de l'ADN sont celle de Maxam-Gilbert [45] et celle de Sanger [72]. La méthode de Sanger continue à être employée et de nouvelles techniques sont apparues. Elles ont permis d'accélérer les manipulations, minimiser l'espace occupé et ainsi diminuer le coût du séquençage.

La méthode Sanger se base sur la migration de fragments d'ADN de toutes tailles terminées par un nucléotide spécifique et débutant tous au premier nucléotide de l'ADN séquencé. La migration s'effectue par électrophorèse et permet de déterminer l'ordre des nucléotides en fonction de la taille des brins d'ADN.

La technique du "shotgun" (ou séquençage aléatoire global) est utilisée massivement pour le séquençage d'un grand nombre de génomes. Le principe est de fragmenter plusieurs exemplaires d'un même génome entier en petits fragments. Après amplification par PCR (voir ci-dessous pour des précisions) et séquençage par la méthode de Sanger, les séquences obtenues sont regroupées pour obtenir des "contigs", le but étant de reconstituer le génome en entier. Cette étape est appelée le contigage.

PCR

Une PCR, ou Réaction de Polymérisation en Chaîne ("Polymerase Chain Reaction" en anglais), permet d'amplifier *in vitro* une région spécifique d'acides nucléiques donnée afin d'en obtenir une quantité suffisante pour la détecter et l'étudier.
Pour ce faire, une série de réactions permettant la réplication d'une matrice d'ADN double brin est répétée en boucle. Ainsi, au cours de la réaction PCR, les produits obtenus à la fin de chaque cycle servent de matrice pour le cycle suivant, l'amplification est donc exponentielle.
Pour répliquer un double brin d'ADN, il faut agir en trois étapes :

1. dénaturer l'ADN double brin pour obtenir deux simples brins (pour cela il faut réchauffer l'ADN à environ 95°C) ;
2. amorcer la réplication de la séquence à amplifier à l'aide d'amorces spécifiques ;
3. synthétiser entièrement un nouveau brin complémentaire grâce à une protéine : l'ADN polymérase.

À la fin de chaque cycle, les produits sont sous forme d'ADN double brin.

Le séquençage par hybridation a été introduit en 1988 et se base, non pas sur la migration de fragments en électrophorèse, mais sur l'hybridation. L'idée proposée est de déterminer la fréquence d'hybridation de courtes séquences nucléotides à un ADN génomique, d'assembler l'ensemble de ces courtes séquences parfaitement hybridées en une séquence unique et de comparer celle-ci à un ADN de référence [37, 67, 20].

Le pyroséquençage [1] est une technique qui commence à supplanter la méthode de Sanger. Elle permet d'analyser la synthèse d'ADN cible en temps réel. Les nucléotides ne sont pas ajoutés tous ensemble comme dans les réactions de séquences normales, mais l'un après l'autre. Si le nucléotide ajouté dans le milieu réactionnel correspond à celui attendu, il est incorporé dans le brin en cours de synthèse en libérant un PyroPhosphate. Cette libération déclenche un signal lumineux qui est mesuré par la caméra du séquenceur de manière automatique. Cette méthode est très précise mais permet actuellement de ne séquencer que de courts segments d'ADN (jusqu'à 450 paires de bases). Toutefois l'arrivée d'automates permet le séquençage massif parallèle.

Pour le séquençage de grands génomes des robots, permettant d'automatiser au maximum le séquençage, sont utilisés. De plus, l'utilisation de marqueurs facilite le contigage des grands génomes. Il s'agit d'établir des repères sur le génome à l'aide de marqueurs spécifiques (sites de restriction, microsatellites, STS - "Sequence Tagged Sites") de chaque chromosome. Ainsi, la zone à étudier est limitée ce qui facilite la reconstitution du génome à séquencer.

En 1977, la première séquence d'un être vivant a été obtenue par la méthode de Sanger [71] ; il s'agissait du génome du bactériophage X174 (si nous considérons les virus comme des êtres vivants). En 1996, le premier génome eucaryote est séquencé : *Saccharomyces cerevisiae* [46]. Le premier séquençage du génome d'une plante a eu lieu en 2000 [52] : *Arabidopsis thaliana*. La séquence complète du génome humain contenu dans le noyau de la cellule a été finalisée en 2003 [2]. Les génomes de nombreux agents infectieux, de mammifères et de plantes ont également été séquencés dans leur totalité [3] [4]. Au milieu de l'année 2009, 12 410 espèces de bactéries ont pu être classées grâce au séquençage de l'ARNr 16S [5]. Le nombre de génomes séquencés augmente de façon exponentielle.

Annotations structurales et fonctionnelles

Une fois la séquence en nucléotides du génome obtenue, il est nécessaire de donner du sens à cette information. La première étape est généralement l'*annotation structurale* de la séquence. Il s'agit de positionner, dans le génome, les gènes ainsi que d'autres objets biologiques : motifs de régulation, éléments transposables, etc.

L'identification des positions se fait par différentes méthodes, décrites ci-dessous. Pour une présentation complète et détaillée de ces méthodes voir [66].

La méthode expérimentale consiste à synthétiser expérimentalement un brin d'ADN complémentaire (ADNc) à partir de la séquence connue d'un ARN. Cet ADNc est aligné sur la séquence complète du génome et permet ainsi d'identifier par comparaison des séquences les gènes. Cette méthode est la plus sûre car expérimentale, mais se heurte malgré tout à des problèmes pratiques, en particulier en raison de l'existence d'une phase de maturation des ARN codants qui modifie la séquence de nucléotides. De plus, il est difficile d'obtenir l'ensemble des séquences d'ARN. [6].

2. http ://www.genome.gov/
3. http ://www.ncbi.nlm.nih.gov/Genomes/
4. http ://genomesonline.org
5. www.bacterio.cict.fr
6. Certains gènes se sont pas transcrits en quantité suffisante au moment de la manipulation.

Les méthodes comparatives consistent à comparer la séquence génomique étudiée avec 3 types d'objets biologiques :

- les ESTs (courtes séquences d'ADNc marquant le début et la fin des ARN) déjà connus ;
- les séquences protéiques des sources de données publiques qui sont alors comparées à la traduction de la séquence génomique étudiée. L'identification d'une protéine permet d'identifier la portion de séquence la codant.
- les séquences génomiques d'espèces proches qui peuvent nous permettre d'identifier des gènes proches. Nous pouvons alors les positionner *via* une comparaison de la séquence étudiée.

Les méthodes *ab initio* cherchent à reconnaître les particularités communes à tous les gènes déjà détectés sur un génome (proportion de nucléotides ou de codons particulières, séquences encadrant un gène ou motif spécifique, etc) afin de trouver d'autres gènes sur l'ensemble de la séquence possèdant ces mêmes particularités. La majorité des algorithmes sont à base de modèles de Markov cachés. Ces méthodes, à la différence des méthodes comparatives, permettent de découvrir de nouveaux gènes.

Les méthodes intégratives combinent les méthodes précédemment citées.

Une autre étape fondamentale qui suit le séquençage d'un génome est l'*annotation fonctionnelle*. Il s'agit d'associer à une protéine, ou un ARN non codant, sa ou ses fonctions biologiques.

L'annotation peut être réalisée expérimentalement ou *in silico*. Expérimentalement, les biologistes réalisent des manipulations, par exemple des mutagenèses[7] au sein des gènes pour en perturber l'expression, et peuvent ainsi caractériser la fonction en observant les fonctionnalités perdues ou perturbées par l'organisme. Cette technique a bien sûr des limites, il n'est pas possible par exemple de réaliser ce type d'expérimentation sur des protéines essentielles à la survie de l'organisme. D'autres techniques expérimentales existent. Elles sont d'une grande précision mais elles sont coûteuses en temps et financièrement.

Aujourd'hui, l'annotation fonctionnelle est souvent réalisée *in silico* par comparaison de séquences. Le principe est de rechercher dans les sources de données publiques des protéines proches. Si les fonctions de ces protéines sont déjà connues, alors l'annotateur peut, sous certaines conditions, propager cette information à la protéine de fonction inconnue. D'autre part, d'autres paramètres permettent d'identifier la fonction d'une séquence : charge électrique d'une protéine, hydrophobicité, présence de signatures spécifiques. Lors de ces différentes études, il est nécessaire de prendre en compte le fait que les protéines

7. Introduction volontaire de mutation.

peuvent posséder différents domaines fonctionnels.

L'enchaînement des gènes d'un génome est maintenant obtenu. Il est alors synthétisé sous la forme d'une carte physique.

Ces cartes permettant d'étudier chaque génome (structure, fonction) et les ensembles de génomes (réarrangements de gènes, variation du nombre de copies des gènes, synténie [8], phylogénie, fonctions/structures/évolutions communes).

1.2 Principes de la génomique comparative

Grâce au séquençage et à l'annotation des génomes, nous obtenons des séquences de gènes avec une identification des homologies. Lorsque nous prenons ces séquences individuellement, nous pouvons considérer différentes caractéristiques : la composition en nucléotides des gènes, le taux d'ADN codant et d'ADN non codant, la relation entre les gènes consécutifs et leurs produits, l'étude des motifs de régulation, etc.

Dans ces travaux, nous allons étudier les génomes en les comparant deux à deux. Ceci nous apporte d'autres informations. En effet, nous pouvons alors comparer les génomes (et donc les espèces) pour mettre en avant les groupes de gènes très ou peu conservés, étudier la conservation de l'ordre des gènes, construire un arbre phylogénétique *via* des matrices de distance, etc.

Nous signalons par ailleurs que la comparaison d'espèces peut se faire à d'autres niveaux : étude des caractéristiques physiques visibles (phénotype), étude des gènes (le génotype), étude des protéines (protéome) ou étude des ARN (transcriptome). Dans ce manuscrit, nous nous focalisons uniquement sur la génomique comparative.

L'analyse des réarrangements génomiques dans la biologie moléculaire a commencé à la fin des années 30 avec Dobzhansky et Sturtevant. Ils utilisent l'ordre des gènes de deux génomes différents comme un indicateur d'une distance d'évolution entre les deux organismes correspondants [35, 36]. Notons qu'en 1980 Palmer et ses collègues montrent que le chou et le navet ont des gènes ayant des séquences très similaires ($99,9\%$ pour 99% des gènes) mais l'ordre de leurs gènes est très différent. Nous désirons savoir si deux génomes possèdent le même ensemble de gènes et si les gènes homologues sont placés au même endroit et/ou dans le même ordre. Ces études se basent sur le principe de *parcimonie* : nous supposons que c'est le minimum de changements évolutifs qui explique les relations phylogéniques.

Dans l'approche informatique, Sankoff est le premier à se baser sur la comparaison de l'ordre des gènes plutôt que sur la comparaison traditionnelle des séquences nucléotidiques.

8. La synténie décrit la conservation de l'ordre des gènes entre deux espèces apparentées.

Dans un premier temps et par volonté de simplifier les comparaisons, les génomes étaient étudiés en ne prenant en compte qu'un exemplaire de gène par famille multigénique dans chaque génome. Dans le cas de séquences possédant des duplications, avant toutes comparaisons, un traitement informatique doit donc être effectué afin d'éliminer les duplications.

Pour étudier deux génomes simultanément, nous introduisons de nouvelles notations.

Nous rappelons que nous étudions deux génomes G_1 et G_2. Dans toute notre étude, nous ignorons systématiquement les familles de gènes pour lesquelles aucune occurrence n'est présente sur un des génomes.

Un gène g est un **singleton** si $\mathsf{occ}_1(|\mathsf{g}|) = 1$ et $\mathsf{occ}_2(|\mathsf{g}|) = 1$ (une unique occurrence de gène par famille sur chaque génome G_1 et G_2). Supposons, dans cette sous-section, que tous les gènes de G_1 et G_2 sont des singletons ; nous avons alors $n_1 = n_2$ et sans perdre de généralité, nous posons que G_1 est l'identité, c'est-à-dire $G_1 = +1\ +2 \ldots +n$. G_2 est alors une permutation signée de G_1.

Dans ce contexte d'absence de duplication, nous présentons les deux stratégies utilisées dans la comparaison génomique : calcul d'une distance de réarrangements ou calcul d'une mesure de (dis)similarité [42]. Les notions de complexité (**NP**-difficile, rapport d'approximation, etc) utilisées dans la suite de ce chapitre sont définies dans la Section 2.2 du Chapitre 2.

1.2.1 Distances de réarrangements

Une première technique pour comparer des génomes consiste à trouver un *scénario* permettant de passer d'un génome à l'autre [35, 36, 86, 82]. Le scénario est constitué d'une suite de réarrangements tels que des inversions ou des transpositions. Le but est alors de trouver quelle séquence de mutations peut expliquer le lien entre deux génomes. Pour cela, le principe de *parcimonie* est utilisé : nous minimisons le nombre de réarrangements permettant de passer d'un génome à l'autre.

Sachant qu'il est possible de pondérer les différents réarrangements, nous pouvons aussi chercher à minimiser le *coût* d'un scénario, c'est-à-dire minimiser la somme des coûts des réarrangements.

Le nombre minimal obtenu entre deux génomes correspond à une *distance de réarrangement*.

Les réarrangements acceptés correspondent à des mutations chromosomiques (voir la Sous-section 1.1.2). En pratique, nous pouvons nous restreindre à seulement certains réarrangements dans le but de simplifier le problème d'un point de vue algorithmique. Une distance de réarrangement étant entièrement dépendante des opérations autorisées, il faut autoriser suffisamment d'opérations pour être réaliste mais pas trop pour pouvoir

Id :	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+6$	$+7$
	$+0$	-6	-5	-4	-3	-2	-1	$+7$
	$+0$	$+2$	$+3$	$+4$	$+5$	$+6$	-1	$+7$
G_1 :	$+0$	$+2$	-4	-3	$+5$	$+6$	-1	$+7$

FIGURE 1.6 – **Scénario.** La distance d'inversion entre Id et G_1 est 3.

développer des algorithmes.

Le calcul de distance de réarrangements permet donc de donner une distance entre deux génomes mais aussi de formuler une hypothèse sur le type et l'ordre des mutations chromosomiques qui ont conduit aux espèces actuelles.

Voici les principales distances étudiées.

Distance d'inversion La distance d'inversion correspond au nombre minimum d'inversions permettant de passer d'un génome à l'autre [86] (voir Figure 1.6). Dans le cas où les gènes ne sont pas signés, Caprara a prouvé que calculer cette distance entre deux génomes est un problème **NP**-difficile [22]. De plus, ce problème n'est pas approximable sous 1,0008 [13]. Le meilleur rapport d'approximation connu est $\frac{11}{8}$ [12].

Cette distance, dans le cas signé, a été introduite en 1989 dans [73]. Dans les années 90, Hannenhali et Pevzner présentent un algorithme polynomial pour le cas signé [50, 51]. En 2001, Bader *et al.* présentent un algorithme linéaire pour calculer cette distance entre deux génomes [9].

Distance de transposition La distance de transposition est le nombre minimum de transpositions permettant de passer d'un génome à l'autre [10]. Le brin sur lequel les gènes sont situés, c'est-à-dire leur signe, n'est pas pris en compte. La complexité de ce problème est actuellement inconnue. Le meilleur rapport d'approximation connu est $\frac{11}{8}$ [40].

Distance de translocation La distance de translocation correspond au nombre minimum de translocations permettant de passer d'un génome à l'autre [34]. Le brin sur lequel les gènes sont situés, c'est-à-dire leur signe, n'est pas pris en compte. La complexité du problème associé est encore inconnue. Le meilleur rapport d'approximation connu est 2 [34].

1.2.2 Mesures de (dis)similarité

Cette seconde approche se base sur la structure des génomes. Le but est de mettre en avant les ressemblances et les différences, entre deux génomes, en les identifiant et en les comptabilisant. Nous pouvons, par exemple, situer et compter les paires de gènes consécutifs sur deux génomes. Dans ce cas, nous mettons en avant la théorie de la synténie.

Cette approche ne permet pas d'obtenir un scénario.

Plusieurs mesures de (dis)similarité ont été proposées ; voir ci-dessous.

Nombre d'adjacences et nombre de points de cassure [76] ("adjacencies" et "breakpoints" en anglais). Le nombre d'*adjacences* (mesure de similarité) et le nombre de *points de cassure* (mesure de dissimilarité) sont deux mesures complémentaires.

> Soient $G_1[i]$ et $G_1[i+1]$ deux gènes consécutifs sur G_1, avec $i \in [1 : n[$. La paire $(G_1[i], G_1[i+1])$ forme une **adjacence** si $G_1[i]$ $G_1[i+1]$ ou -$G_1[i+1]$ -$G_1[i]$ est une sous-séquence de G_2, autrement elle forme un **point de cassure** (voir Figure 1.7 (a)).
> S'il n'y a pas d'ambiguïté, et par abus de langage, nous dirons aussi qu'il existe une adjacence (ou un point de cassure) après le gène $G_1[i]$ sur le génome G_1.

Calculer ces deux mesures entre deux génomes sans duplication est un problème trivial.

Nous notons que le nombre de points de cassure a tout d'abord été introduit pour les permutations comme une bonne approximation de la distance d'inversion [56].

Ces mesures se basent sur le fait que si des groupes de gènes apparaissent consécutivement dans plusieurs espèces alors ils devraient être présents dans le même ordre dans le génome ancestral, car ils ne seraient pas séparés durant l'évolution (hypothèse de parcimonie). Ceci se justifie par le fait que la régulation de l'activité d'un gène dépend, en partie, de sa localisation dans le génome.

Nombre d'intervalles communs [85] **et nombre d'intervalles conservés** [11] ("commun intervals" et "conserved intervals" en anglais). Ces deux mesures de similarité comptent le nombre d'intervalles présents sur les deux génomes (*intervalles communs*). Si les extrémités d'un intervalle commun sont identiques sur les deux génomes alors l'intervalle est dit *conservé*.

> Un intervalle $G_1[\mathrm{g}_1, \mathrm{g}_2]$ est un **intervalle commun** entre G_1 et G_2 s'il existe g_3 et g_4 tel que $G_1[\mathrm{g}_1, \mathrm{g}_2] = G_2[\mathrm{g}_3, \mathrm{g}_4]$. Si $\{\mathrm{g}_1; \mathrm{g}_2\} = \{\mathrm{g}_3; \mathrm{g}_4\}$ alors $G_1[\mathrm{g}_1, \mathrm{g}_2]$ est un **intervalle conservé** (voir Figure 1.7 (b)).
> Pour deux génomes possédant le même ensemble de familles, tous les intervalles de taille 1 sont communs et conservés. De plus, l'intervalle de taille n est commun aux deux génomes. Ces $n+1$ intervalles sont appelés **intervalles trivaux**.
> Pour ces deux mesures, le signe des gènes sur les deux génomes n'est pas pertinent.

Calculer le nombre d'intervalles conservés entre deux génomes, sans duplication, est un problème polynomial : il existe un algorithme de temps $O(n)$ [11].

Le nombre d'intervalles communs est une extension du nombre d'adjacences : il se base sur le fait qu'un groupe de gènes reste consécutif dans un génome mais, cette fois, pas nécessairement dans le même ordre ni sur le même brin.

Nombre maximum et somme des perturbations d'adjacences [77] ("Maximum Adjacency Disruption" et "Summed Adjacency Disruption" en anglais). Ces deux mesures de dissimilarité indiquent le nombre de gènes séparant dans un génome deux gènes consécutifs sur l'autre génome.

> Le nombre **maximum de perturbations d'adjacences** (MAD) est le maximum de gènes entre g_1 et g_2 sur un génome pour tous g_1 et g_2 consécutifs sur l'autre génome, c'est-à-dire
>
> $$\max_{i=1,\,2,\,\ldots,\,n} \left\{ |G_2[G_1[i]] - G_2[G_1[i+1]]|, |G_1[G_2[i]] - G_1[G_2[i+1]]| \right\}.$$

> La **somme des perturbations d'adjacences** (SAD) est la somme des nombres de gènes entre deux gènes g_1 et g_2 sur un génome tel que g_1 et g_2 soient consécutifs sur l'autre génome, c'est-à-dire
>
> $$\sum_{i=1,\,2,\,\ldots,\,n} \left\{ |G_2[G_1[i]] - G_2[G_1[i+1]]|, |G_1[G_2[i]] - G_1[G_2[i+1]]| \right\}$$
>
> (voir Figure 1.7 (c)).

Calculer ces deux mesures est un problème trivial.

Finalement, calculer ces 6 mesures entre deux génomes, sans gène dupliqué et dont l'ordre est totalement connu, sont des problèmes triviaux.

Nous avons fait deux fortes hypothèses jusqu'ici : les génomes étudiés étaient sans duplication et l'ordre de leurs gènes étaient totalement connu. Dans les deux sections suivantes, nous allons remettre en cause chacune de ces hypothèses, indépendamment l'une de l'autre.

(a) Adjacences et points de cassure

G_1	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
G_2	+0	▲ +7	▲ +3	▲ −5	−4	▲ +6	▲ +1	+2	▲ −8	▲ +9

(b) Intervalles communs et intervalles conservés

G_1	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
G_2	+0	+7	+3	−5	−4	+6	+1	+2	−8	+9

(c) MAD et SAD

	6	1	5	2	1	2	4	**7**	1	
G_1	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
G_2	+0	+7	+3	−5	−4	+6	+1	+2	−8	+9
	7	4	2	1	2	5	1	6	1	

FIGURE 1.7 – **Mesures de (dis)similarité.** Soient deux génomes G_1 et G_2 de taille 10. **(a)** Entre G_1 et G_2, deux gènes consécutifs sur G_2 forment un point de cassure si et seulement s'ils sont séparés par le symbole ▲. Sinon, ils forment une adjacence (paires de gènes encadrées sur le génome G_2). Nous avons 7 points de cassure et 2 adjacences entre G_1 et G_2. **(b)** Les 13 intervalles communs entre G_1 et G_2, de taille supérieure à 1, sont indiqués par un trait soulignant les gènes de l'intervalle correspondant dans G_2. Les 6 traits gras sont des intervalles conservés. Il y a, en comptant les intervalles triviaux, 23 intervalles communs et 16 intervalles conservés entre G_1 et G_2. **(c)** Sur le génome G_1 (G_2, respectivement), pour deux gènes consécutifs donnés, le nombre de gènes séparant ces deux gènes sur le génome G_2 (G_1, respectivement) est indiqué au-dessus (en-dessous, respectivement) d'une accolade. La mesure MAD est égale à 7 (en gras) et la mesure SAD est égale à 58, entre G_1 et G_2.

1.3 Comparaison génomique avec prise en compte des duplications de gènes

Calculer des mesures de (dis)similarité entre deux génomes sans duplication est simple ; mais nous ne considérons alors qu'un seul homologue par famille et par génome. Ce choix peut se justifier pour des petits virus, des mitochondries ou des segments de chromosomes sans duplication. Cependant il est important, en pratique, de pouvoir prendre en compte les duplications de gène. En effet, le pourcentage de gènes dupliqués dans la plupart des génomes n'est pas négligeable. Nous comptons 15% de gènes dupliqués chez l'*Homo sapiens* [61], 16% chez la *Saccharomyces cerevisiae* [46] et 25% chez *Arabidopsis thaliana* [52]. Nous notons que le taux de duplications varie selon chaque espèce avec une plus forte proportion de duplication de gène chez les plantes.

Nous introduisons de nouvelles notations pour considérer des génomes avec duplications.

Soient deux génomes G_1 et G_2 possédant des gènes dupliqués et deux entiers m_1 et m_2. La paire de génomes (G_1, G_2) est dite de **type** (m_1, m_2) si $\mathsf{occ}_1 = m_1$ et $\mathsf{occ}_2 = m_2$. La paire de génomes (G_1, G_2) est dite **équilibrée** si, pour chaque famille de gène $\mathtt{f} \in \mathcal{F}_1 \cup \mathcal{F}_2$, nous avons $\mathsf{occ}_1(\mathtt{f}) = \mathsf{occ}_2(\mathtt{f})$. Dans le cas contraire, la paire (G_1, G_2) est dite **déséquilibrée**. Notons qu'une paire (G_1, G_2) de type (m, m) n'est pas nécessairement équilibrée.

`Exemple` : Si nous considérons les génomes G_1 = +1 +2 +3 +4 +5 -1 -2 +6 -2 et G_2 = +2 -1 -4 +6 +3 +4 -5 -2 -2, alors la paire (G_1, G_2) est de type (3,3) car pour chaque génome la plus grand famille de gènes (la famille 2 pour G_1 et G_2) possède 3 occurrences de gènes : $\mathsf{occ}_1 = \mathsf{occ}_2 = 3$. Mais la paire (G_1, G_2) est déséquilibrée car $\mathsf{occ}_1(4) = 1$ et $\mathsf{occ}_2(4) = 2$.

1.3.1 Couplage entre gènes homologues

Au début des années 2000, des recherches informatiques ont été réalisées afin de prendre en compte les duplications dans la génomique comparative [75, 21]. La stratégie proposée alors consiste à trouver une correspondance entre les gènes orthologues de deux génomes deux-à-deux, tout en optimisant une distance de réarrangements ou une mesure de (dis)similarité. Ainsi, nous renommons les gènes des deux génomes comparés en fonction de cette correspondance puis nous supprimons les gènes sans correspondance pour obtenir deux génomes sans duplication de gènes. Nous pouvons alors calculer les distances de réarrangements et les mesures de (dis)similarité présentées dans les Sous-sections 1.2.1 et 1.2.2, respectivement. De plus, *via* cette méthode, nous proposons un ensemble de gènes ancestraux défini par les gènes mis en correspondance.

Pour représenter cette correspondance, nous utilisons un *couplage*.

Un **couplage** $\mathcal{C}$ entre deux génomes G_1 et G_2 est un ensemble de paires disjointes $(G_1[i], G_2[j])$, où $G_1[i]$ et $G_2[j]$ appartiennent à la même famille multigénique, c'est-à-dire $|G_1[i]| = |G_2[j]|$. Les gènes de G_1 et G_2 appartenant à une paire du couplage $\mathcal{C}$ sont dit **saturés** par $\mathcal{C}$, ou par raccourci **$\mathcal{C}$-saturés**. La taille du couplage $\mathcal{C}$ est notée $|\mathcal{C}|$.

Un couplage $\mathcal{C}$ entre G_1 et G_2 est dit **maximum** si, pour chaque famille de gènes, il n'y a pas deux gènes de cette famille qui ne sont pas $\mathcal{C}$-saturés et qui appartiennent à G_1 et G_2, respectivement.
Après avoir supprimé les gènes non-saturés et renommé les gènes dans G_1 et G_2, en accord avec un couplage $\mathcal{C}$ de G_1 et G_2, nous avons deux génomes sans duplication que nous nommons $G_1^{\mathcal{C}}$ et $G_2^{\mathcal{C}}$; $G_2^{\mathcal{C}}$ est une permutation signée de $G_1^{\mathcal{C}}$. Nous appelons ces deux nouveaux génomes des génomes **$\mathcal{C}$-élagués**.

Une fois que les génomes $\mathcal{C}$-élagués sont obtenus, nous pouvons calculer des distances de réarrangements et, en un temps polynomial, des mesures de (dis)similarité. Mais il existe de nombreux couplages entre deux génomes donnés (voir Figure 1.8). Ils peuvent conduire à différentes valeurs de distances/mesures. C'est pourquoi nous cherchons un couplage qui optimise une mesure donnée mettant ainsi en avant les réarrangements ou la (dis)similarité choisis. Alors que trouver un couplage quelconque et calculer une mesure peut être simple, trouver un couplage qui optimise une distance/mesure est un problème nettement plus complexe en terme de temps et d'espace de calcul. Notons qu'il existe toujours plusieurs couplages possibles même lors d'une optimisation.

Afin d'obtenir un cadre algorithmique simplifié, deux modèles de couplage sont généralement considérés : le modèle *exemplaire* et le modèle *maximum* (voir Figure 1.8 (a) et (c)).

Pour le modèle **exemplaire**, le couplage $\mathcal{C}$ doit saturer exactement un gène pour chaque famille de gène ; la taille du couplage est donc le nombre de famille de gènes.

Lors de l'introduction du modèle exemplaire [75], Sankoff pose l'hypothèse qu'un seul exemplaire par famille de gène est présent dans le génome ancestral des deux génomes comparés. Ce modèle met alors en avant cet exemplaire de gène, le descendant direct, en supposant que cet exemplaire est celui qui aura été le moins fréquemment déplacé.

Pour le modèle **maximum**, le couplage $\mathcal{C}$ doit être maximum, c'est-à-dire saturer le plus possible de gènes de chaque famille multigénique. Ici encore, la taille de $\mathcal{C}$ est facilement calculable, à partir des deux génomes de départ :

$$|\mathcal{C}| = \sum_{\mathtt{f} \in \mathcal{F}_1 \cup \mathcal{F}_2} \min(\mathtt{occ}_1(\mathtt{f}), \mathtt{occ}_2(\mathtt{f})).$$

Le modèle maximum suppose que le maximum de gènes d'une famille sont présents dans le génome ancestral.

Ces deux modèles sont clairement des modèles extrêmes permettant de simplifier le cadre algorithmique mais ils imposent de fortes contraintes qui sont biologiquement fausses. C'est pourquoi nous introduisons et étudions un troisième modèle, que nous appelons le modèle *intermédiaire* (voir Figure 1.8 (b)). Ce modèle, plus réaliste, fait le lien entre les deux modèles extrêmes que sont les modèles exemplaire et maximum.

Pour le modèle **intermédiaire**, le couplage $\mathcal{C}$ doit saturer *au moins* un gène de chaque famille de gène. La taille du couplage, sous ce modèle, n'est pas une constante. Elle est comprise entre la taille d'un couplage exemplaire et celle d'un couplage maximum et elle est dépendante de la distance ou de la mesure que nous souhaitons optimiser. Nous notons, par ailleurs, que les couplages exemplaire et maximum sont, par définition, des couplages intermédiaires.

1.3.2 État de l'art

Divers travaux ont été effectués sur la comparaison de génomes avec des gènes dupliqués [42].

Distance de réarrangements : distance d'inversion signée

Cette distance a été introduite par Sankoff [74] pour le modèle exemplaire ; et par Chen *et al.* [26] pour le modèle maximum. Calculer la distance d'inversion signée entre deux génomes, en présence de duplications, est un problème **NP**-complet (voir Bryant [21] pour le modèle exemplaire), **APX**-difficile (voir Angibaud *et al.* [4] pour les trois modèles) et non approximable (voir Chen *et al.* [28] pour le modèle exemplaire). Sankoff présente un algorithme exact dans [74] pour le modèle exemplaire.

Mesures de (dis)similarité

Nous notons tout d'abord la définition suivante.

Définition 1.1 *Une mesure de dissimilarité, entre deux génomes signés, est dite* monotonique*, si pour toute paire de génome G_1 et G_2, supprimer toutes les occurrences d'une famille multigénique dans G_1 et G_2 n'augmente pas cette mesure, pour un modèle donné, entre G_1 et G_2.*

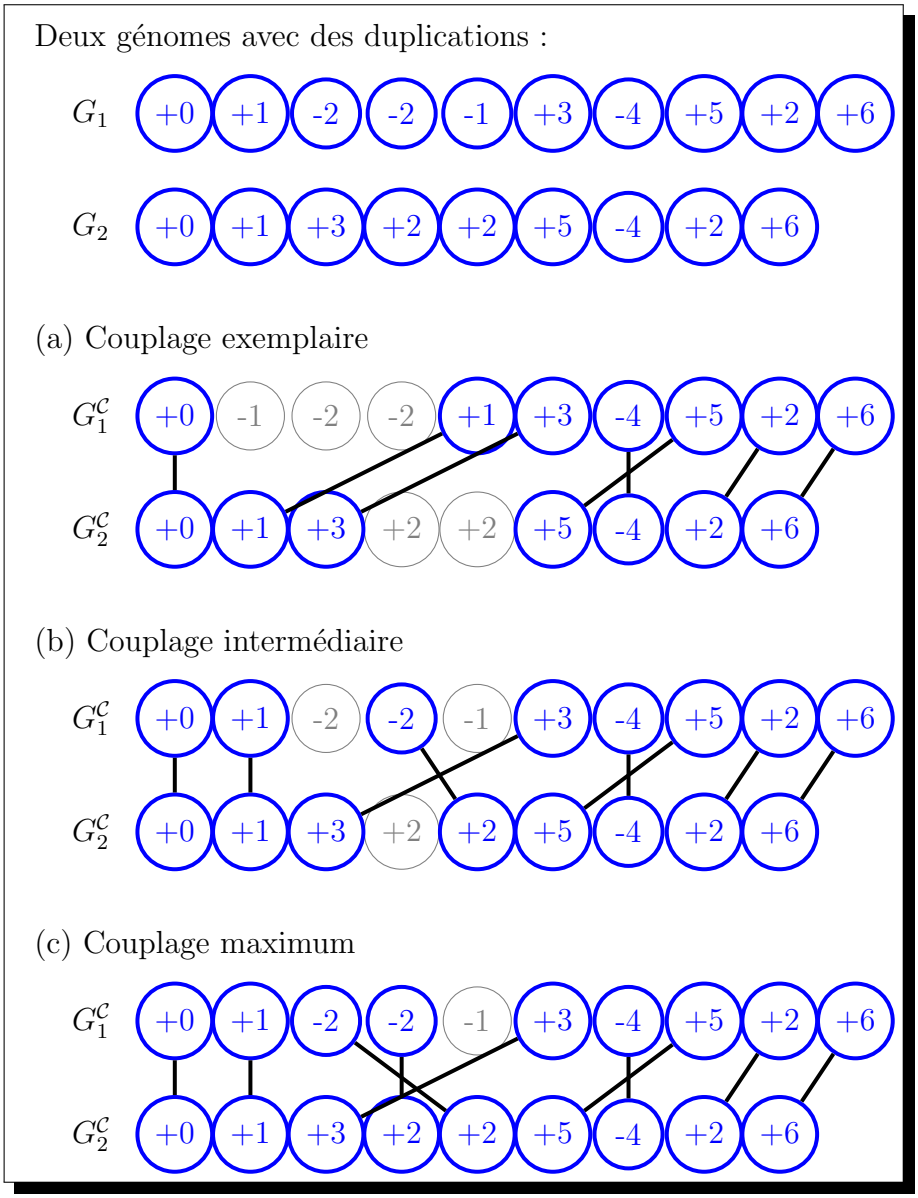


FIGURE 1.8 – **Modèles de couplage entre deux génomes G_1 et G_2 : (a)** exemplaire, **(b)** intermédiaire et **(c)** maximum

Points de cassure Cette mesure de dissimilarité a été introduite par Sankoff [74] (pour le modèle exemplaire), par Angibaud *et al.* [2] (pour le modèle intermédiaire) et par Blin *et al.* [16] (pour le modèle maximum). Calculer cette mesure entre deux génomes avec duplications est un problème **NP**-complet (voir Bryant[21]) et **APX**-difficile (voir Angibaud *et al.* [4]) pour les trois modèles. Pour les modèles exemplaire et intermédiaire, le problème n'est pas approximable (voir Chen *et al.* [28]).

Trois algorithmes exacts, tous basés sur une technique de séparation et évaluation (*branch-and-bound* en anglais, voir l'encadré ci-dessous), ont été proposés, voir ci-dessous.

- Pour le modèle exemplaire, l'algorithme proposé par Sankoff [74] construit tous les couplages exemplaires possibles entre les génomes G_1 et G_2. Pour cela, une paire de gènes d'une même famille est ajoutée à chaque étape. Une étape est annulée dès que le nombre de points de cassure est supérieur au meilleur résultat déjà obtenu. L'algorithme procède ainsi, jusqu'à obtenir un couplage exemplaire minimisant le nombre de points de cassure. Cet algorithme utilise le fait que le nombre de points de cassure est une mesure monotonique (voir Définition 1.1).
- Nguyen *et al.* [68] pour les modèles exemplaire et intermédiaire, proposent de réduire le temps de calcul de l'algorithme de Sankoff en identifiant des sous-ensembles indépendants disjoints de familles multigéniques.
- Blin *et al.* [16], pour le modèle maximum, proposent aussi un algorithme de séparation et évaluation utilisant le principe de monotonie. L'algorithme commence par un couplage vide et étudie toutes les possibilités pour l'étendre, tant que l'actuel couplage ne dépasse pas la meilleure distance, pour finalement obtenir un couplage maximum. Un arbre de suffixe est utilisé pour stocker les sous-mots de G_2 et pour chercher les sous-mots qui sont communs à G_1 et G_2, fournissant ainsi les candidats à l'extension du couplage.

Évaluation et séparation

Un algorithme par séparation et évaluation, *branch and bound* en anglais, est une méthode générique de résolution de problèmes d'optimisation. C'est une méthode d'énumération implicite : toutes les solutions possibles du problème peuvent être énumérées mais, l'analyse des propriétés du problème permet d'éviter l'énumération de mauvaises solutions. Dans un bon algorithme par séparation et évaluation, seules les solutions potentiellement bonnes sont donc énumérées.

En résumé, la séparation permet d'obtenir une méthode pour énumérer toutes les solutions tandis que l'évaluation évite l'énumération systématique de toutes les solutions.

Adjacences Comme nous le verrons dans la Section 3.2.1, minimiser le nombre de points de cassure et maximiser le nombre d'adjacences sont deux problèmes différents, si nous considérons les duplications, pour le modèle intermédiaire. Trouver un couplage qui maximise le nombre d'adjacences est un problème **NP**-complet, **W[1]**-difficile et n'admet pas de $n^{1-\epsilon}$-approximation (voir Chen *et al.* [27]), pour les trois modèles. Pour deux génomes équilibrés, le meilleur rapport d'approximation pour le modèle maximum est 4 (voir Angibaud *et al.* [4]).

Intervalles communs Le problème qui consiste à trouver un couplage, entre deux génomes avec duplications, qui maximise le nombre d'intervalles communs a été introduit par Chauve *et al.* [25] (pour les modèles exemplaire et maximum) et par Angibaud *et al.* [5] (pour le modèle intermédiaire). Ce problème est **NP**-complet (voir Chauve *et al.* [25]) et **APX**-difficile (voir Angibaud *et al.* [4]) pour les trois modèles. Un algorithme exact est présenté par Angibaud *et al.* dans [5].

1.4 Comparaison génomique avec connaissance partiel de l'ordre des gènes

Nous supposons de nouveau que les génomes ne possèdent pas de gènes dupliqués. Nous allons maintenant remettre en cause l'hypothèse de l'ordre total des gènes d'un génome. En effet, nous possédons parfois des génomes dont l'ordre des gènes n'est que partiellement connu. Cela peut être la conséquence de deux choses :

- soit le séquençage et l'annotation effectués ne permettent pas de conclure sur l'ordre de gènes (du à un manque de précision dans la technique employée, à de nombreuses zones dupliquées rendant le contigage difficile, etc),
- soient plusieurs études du génome d'une même espèce ne sont pas complémentaires.

En effet, pour un même génome, plusieurs séquençages et/ou annotations peuvent-être effectués, parfois *via* différentes méthodes, parfois par différentes équipes. Il faut alors réunir l'ensemble des informations pour déduire la véritable séquence du génome. Afin d'illustrer les problèmes qui peuvent alors intervenir, étudions l'exemple donné par Blin *et al.* [15]. Nous ignorons, le temps de cet exemple, le signe des gènes, c'est-à-dire le brin sur lequel ils sont situés.

Soient deux cartes physiques pour un même génome, décrit par C_A et C_B (voir Figure 1.9 (a)). À cause d'incertitude lors du séquençage et de l'annotation, nous ne possédons pas d'ordre total d'un côté comme de l'autre. Si nous combinons l'ensemble des informations de ces deux séquençages, nous obtenons la carte physique C_{AB} (voir Figure 1.9 (b)). Nous remarquons que nous ne connaissons toujours pas la relation entre tous les gènes (entre les gènes 13 et 14 par exemple). De plus, des conflits sont apparus : le gène 4 doit être à

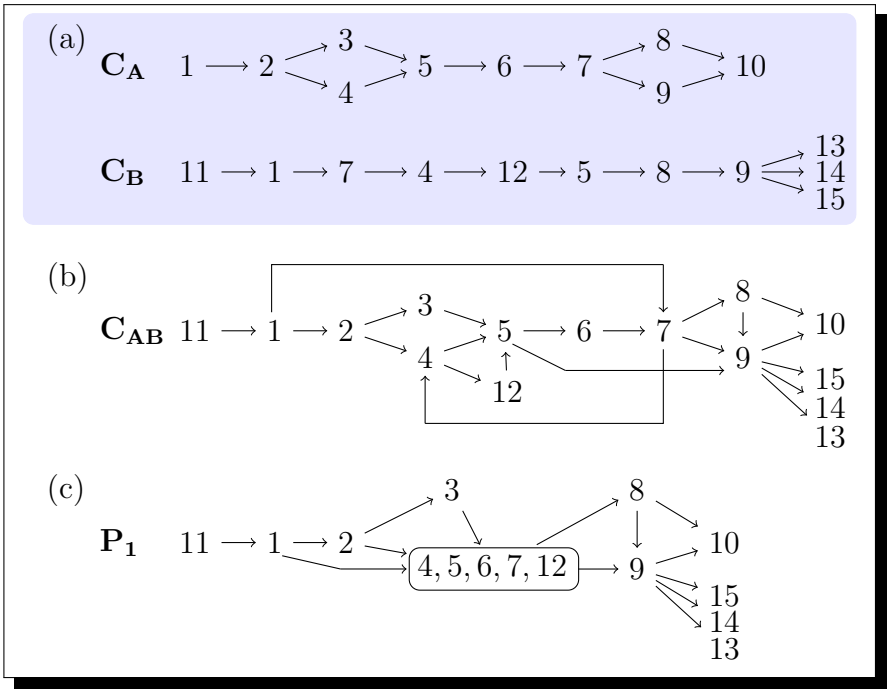


FIGURE 1.9 – **Combinaison d'ordres partiels (a)** Soit deux cartes C_A et C_B. **(b)** En effectuant une union de C_A et C_B, nous obtenons C_{AB}. **(c)** En supprimant les conflits dans C_{AB}, nous obtenons l'ordre partiel P_1.

la fois avant et après le gène 7 (nous avons un *cycle*). Nous n'avons donc toujours pas un ordre total.

Une première étape consiste alors à supprimer les conflits. Pour cela, différentes méthodes existent [87, 53, 54]. Pour cet exemple, la méthode de simplification par composantes étroitement liées ("strongly connected components" , en anglais) est utilisée. Nous obtenons finalement l'ordre partiel P_1 (voir Figure 1.9 (c)). La deuxième étape consiste à trouver l'ordre total du génome à partir de cet ordre partiel.

Notre étude portant sur la deuxième étape, nous parlons maintenant uniquement des génomes obtenus après la suppression des conflits.

1.4.1 Calcul d'extensions linéaires

Pour étudier les génomes, maintenant, nous ne devons plus les représenter par des séquences mais par des ordres partiels [89].

Il est important de rappeler que nous supposons de nouveau que les génomes ne possèdent pas de duplication et ce dans un souci de simplicité algorithmique.

Définition 1.2 *Un* ordre partiel *est un ensemble d'éléments $\mathcal{E}$ et une relation binaire $\preceq$ tels que pour tout x, y et z dans $\mathcal{E}$:*

1. $x \preceq x$
2. $x \preceq y$ *et* $y \preceq x$ *implique* $x = y$, *et*
3. $x \preceq y$ *et* $y \preceq z$ *implique* $x \preceq z$.

Ces trois relations sont des propriétés de réflexivité, d'antisymétrie et de transitivité, respectivement.

Ci-dessous, nous présentons les notations utilisées par la suite, empruntées au livre de Davey et Priestley [32].

> Dans la suite, nous parlons toujours que de deux ordres partiels : P_1 et P_2. Soit $x \in \{1; 2\}$. Les éléments de P_1 et P_2 sont des gènes signés représentés par les entiers $1, 2, \ldots, n_x$, au signe près. Le nombre d'éléments dans P_x, noté n_x, est appelé **taille**.

> Nous notons $\mathrm{g}_1 \preceq_x \mathrm{g}_2$ si le gène g_1 précède le gène g_2 dans P_x. Pour un gène g_2 donné, le nombre de gènes g_1 tel que $\mathrm{g}_1 \preceq_x \mathrm{g}_2$ est noté $prec_x(\mathrm{g}_2)$. De façon symétrique, nous notons $\mathrm{g}_2 \succeq_x \mathrm{g}_1$ si le gène g_2 succède à g_1 dans P_x. Pour un gène g_1 donné, le nombre de gènes g_2 tel que $\mathrm{g}_1 \preceq_x \mathrm{g}_2$ est noté $succ_x(\mathrm{g}_1)$.

Deux gènes g_1 et g_2 sont dits **attenants** dans P_x si

1. $g_1 \preceq_x g_2$ ou $g_2 \preceq_x g_1$ et
2. s'il n'existe pas de gène g_3 tel que $g_1 \preceq_x g_3 \preceq_x g_2$ ou $g_2 \preceq_x g_3 \preceq_x g_1$.

Dans P_x, deux gènes g_1 et g_2 sont **incomparables** (noté $g_1 \|_x g_2$) si ni $g_1 \preceq_x g_2$ ni $g_2 \preceq_x g_1$. Sinon g_1 et g_2 sont **comparables**.

Un **sous-ensemble** $\mathcal{Q}_x$ d'un ordre partiel P_x est un ordre partiel tel que pour tous gènes g_1 et g_2 de $\mathcal{Q}_x$

1. g_1 et g_2 appartiennent à P_x et
2. si g_1 précède g_2 dans P_x alors g_1 précède g_2 dans $\mathcal{Q}_x$.

Un sous-ensemble $\mathcal{Q}_x$ d'un ordre partiel P_x est une **anti-chaîne** si, pour tous gènes g_1 et g_2 de $\mathcal{Q}_x$, $g_1 \preceq_x g_2$ si et seulement si $g_1 = g_2$, c'est-à-dire que toutes les paires de gènes de $\mathcal{Q}_x$ sont incomparables.
La **largeur** d'un ordre partiel P_x est la taille de l'anti-chaîne maximale de P_x.

Soit $\mathcal{Q}_x$ un sous-ensemble de P_x. $\mathcal{Q}_x$ est un **ensemble descendant** si pour tout $g_1 \in \mathcal{Q}_x$, $g_2 \in P_x$ et $g_2 \preceq g_1$, nous avons $g_2 \in \mathcal{Q}_x$. Pour tout sous-ensemble $\mathcal{Q}_x$ de P_x, l'**ensemble inférieur** de $\mathcal{Q}_x$ est l'ensemble $\{g_2 \in P_x : g_2 \preceq g_1 \text{ pour tout } g_1 \in \mathcal{Q}_x\}$, et est noté $\downarrow \mathcal{Q}_x$.

`Exemple` : Dans P_1 (voir Figure 1.10) les gènes 12 et 18 sont attenants. Nous avons $12 \preceq_1 g_2$ pour tout g_2 dans $\{+13; +14; \ldots; +20\}$; $succ_1(+12) = 8$. Et nous avons $+3 \succeq_1 g_1$ pour tout g_1 dans $\{+1; +2\}$; $prec_1 = 2$. Les gènes $+14$, $+15$, $+16$ et $+17$ sont incomparables dans P_1. La largeur de P_1 est 4. Soit le sous-ensemble $\mathcal{Q}_x$ de P_x composé des gènes de $\{+9; +4\}$. L'ensemble inférieur $\downarrow \mathcal{Q}_x$ de $\mathcal{Q}_x$ est $\{+0; +1; +2; +3; +4; -7; -8; +9\}$. Le sous-ensemble de P_x composé des gènes $\downarrow \mathcal{Q}_x$ est un ensemble descendant.

Dans le but de prédire l'ordre total d'un génome, une méthode a été proposée par Zheng *et al.* [88, 78] : confronter l'ordre partiel avec l'ordre total d'un génome proche dont l'ordre des gènes est bien connu. Plus précisément, cette méthode cherche un ordre total (i) respectant l'ordre partiel déjà établi et (ii) optimisant un critère de comparaison entre cet ordre total et celui du génome proche et connu. L'optimisation est soit la minimisation d'une distance de réarrangements ou d'une mesure de dissimilarité, soit la maximisation d'une mesure de similarité. Elle est telle qu'il n'existe pas d'ordre total dont le critère de comparaison avec le génome connu est meilleur.

Une telle optimisation nous permet d'obtenir un ordre total mais aussi d'obtenir un distance ou une mesure entre deux génomes. Nous pouvons, dans la même optique, comparer deux ordres partiels en cherchant deux ordres totaux, en accord avec chacun d'eux,

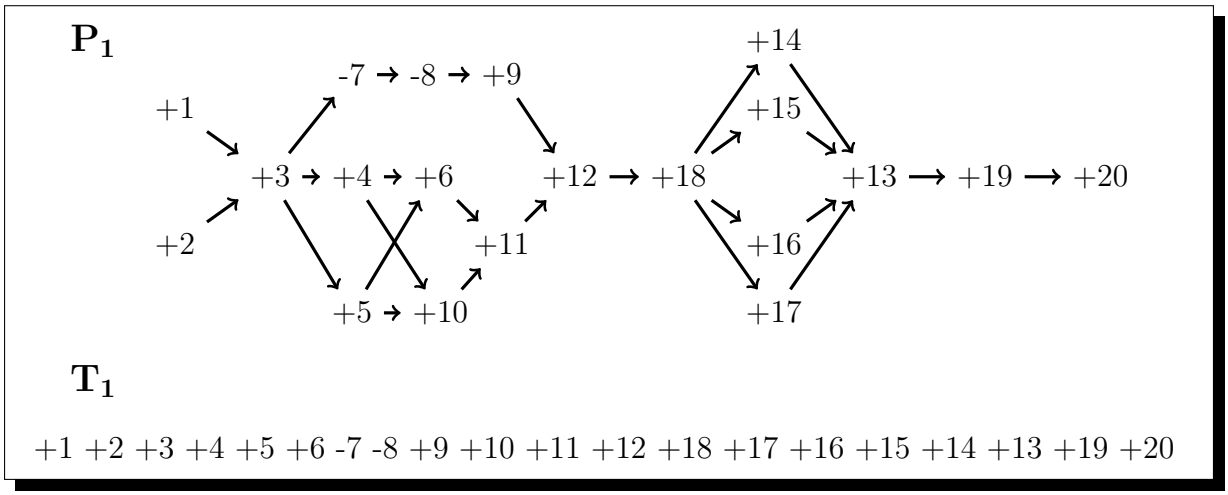


FIGURE 1.10 – **Une extension linéaire** de l'ordre partiel P_1 à 20 gènes.

optimisant une distance/mesure. Nous ne nous basons plus sur un génome totalement ordonné pour étudier un autre génome ; mais nous étudions simultanément deux génomes potentiellement proches afin d'étendre leur ordre de la façon la plus réaliste possible.

Que nous cherchions l'ordre total d'un ou de deux génomes partiellement ordonnés, nous posons les notations suivantes.

Un **ordre total** est un ordre partiel tel que toutes les paires de gènes sont comparables. Un ordre total peut être, de façon équivalente, représenté par une séquence. En particulier, nous appelons l'**idendité** l'ordre total Id tel que $+1 \preceq +2 \preceq \ldots \preceq +n$, ce qui correspond à la séquence $+1\ +2 \ldots + n$.
Une **extension linéaire** de P_x est un ordre total T_x avec le même ensemble de gènes, tel que si g_1 précède g_2 dans P_x alors g_1 précède g_2 dans T_x. Dans la suite, nous aurons toujours T_1 et T_2 des extensions linéaires de P_1 et P_2, respectivement.

L'ensemble des **positions** possibles pour un gène g dans toutes extensions linéaires T_x de P_x est noté $\mathsf{pos}_x(\mathrm{g})$; nous avons $\mathsf{pos}_x(\mathrm{g})$ égale à $[prec_x(\mathrm{g}) + 1, succ_x(\mathrm{g}) - 1]$.
La position du gène g dans un T_x donné est notée $T_x(\mathrm{g})$; $T_x(\mathrm{g})$ appartient à $\mathsf{pos}_x(\mathrm{g})$.

Le gène g_1 est une **i-cheville$_\mathbf{x}$** si quelle que soit l'extension linéaire T_x de P_x nous avons $T_x(\mathrm{g}_1) = i$; c'est-à-dire si
– il y a $i - 1$ gènes g_2 tel que $\mathrm{g}_2 \preceq_x \mathrm{g}_1$ et
– si aucun gène est incomparable avec g_1 dans P_x.
Dans ce cas $\mathsf{pos}_x(\mathrm{g}_1)$ est égale à $\{i\}$.

`Exemple` : Dans P_1 (voir Figure 1.10), les gènes $+3$, $+12$, $+13$, $+18$, $+19$ et $+20$ sont des *i-chevilles*$_1$ avec i égale à 3, 12, 18, 13, 19 et 20 respectivement. Nous avons $\mathsf{pos}_1(+12) = 12$ et $\mathsf{pos}_1(+14) = \{15; 16; 17; 18\}$. T_1 est une extention linéaire possible de P_1. Nous avons $T_1(14) = 18$.

1.4.2 État de l'art

Divers travaux portent sur le calcul d'ordres partiels *via* l'optimisation d'une distance ou d'une mesure [42].

Distance de réarrangements : distance de renversements

Lorsqu'il s'agit de trouver un scénario permettant de passer d'une extension linéaire d'un génome à l'extension linéaire d'un autre génome, la *distance de renversements* est la distance la plus étudiée. Cette distance, entre deux ordres partiels P_1 et P_2, est définie comme le nombre minimum de renversements entre une extension linéaire de P_1 et une extension linéaire de P_2. Cette distance a été introduite par Zheng *et al.* [88]. Le problème correspondant est **NP**-complet (Fu et Jiang [43]). Un algorithme par séparation et évaluation a été proposé par Zheng *et al.* [88].

Mesure de (dis)similarité

Nombres d'adjacences/de points de cassure Si nous désirons calculer une mesure de (dis)similarité, le nombre d'adjacences et le nombre de points de cassure sont les mesures couramment étudiées. Nous rappelons que nous ne prenons pas en compte les duplications de gènes. Nous pouvons alors garantir que maximiser le nombre d'adjacences est équivalent à minimiser le nombre de points de cassure.

La mesure d'adjacences entre deux ordres partiels P_1 et P_2 est définie comme le nombre maximal d'adjacences entre une extension linéaire de P_1 et une extension linéaire de P_2. Calculer cette mesure est un problème introduit par Blin *et al.* [15]. C'est un problème **NP**-complet [15, 43].

Nous distinguons deux cas : comparaison d'un ordre partiel et d'un ordre total (problème MAL-1OP) ou, de manière plus générale, comparaison de deux ordres partiels (problème MAL-2OP).

MAL-1OP

- **Entrée** : Un ordre partiel P_1.
- **Sortie** : Une extension linéaire T_1 de P_1 qui maximise le nombre d'adjacences entre T_1 et Id.

MAL-2OP

• **Entrée** : Deux ordres partiels P_1 et P_2.

• **Sortie** : Deux extensions linéaires T_1 et T_2 de P_1 et P_2 respectivement qui maximisent le nombre d'adjacences entre T_1 et T_2.

Algorithmes exacts Pour la mesure d'adjacences, le seul algorithme exact connu résout le problème MAL-1OP (voir Blin *et al.* [15]). L'idée générale de ce programme dynamique est de calculer pour toutes les anti-chaînes $\mathcal{Q} \subseteq P_1$ et $\mathsf{g}_1 \in \max(\mathcal{Q})$, le nombre $A(\mathcal{Q}, \mathsf{g}_1)$ qui est le nombre maximum d'adjacences obtenues par une extension linéaire de $\mathcal{Q}$ terminant par g_1. Ce nombre peut être calculé grâce à la formule de récursivité suivante : pour tout $\mathcal{Q} \subseteq P_1$, $\mathsf{g}_1 \in max(\mathcal{Q})$,

$$A(\mathcal{Q}, \mathsf{g}_1) = \max_{\mathsf{g}_2 \in \mathcal{Q}'} A(\mathcal{Q}', \mathsf{g}_2) + \begin{cases} 1 & \text{si } |\mathsf{g}_1 - \mathsf{g}_2| = 1 \\ 0 & \text{sinon} \end{cases}$$

où $\mathcal{Q}'$ est l'ensemble des éléments maximaux du plus petit ensemble descendant contenant $\mathcal{Q}$ à l'exception de g_1. L'algorithme tourne en temps $O(n.a_{P_1})$, où a_{P_1} est le nombre d'anti-chaînes dans P_1. Malheureusement a_{P_1} peut être grand (2^n par exemple) et donc ce programme dynamique produit un algorithme impraticable dans la plupart des cas. Cependant nous observons que P_1 est souvent obtenu par la combinaison de peu de cartes physiques. En particulier, si P_1 est obtenu par la combinaison de m cartes physiques, chacune de taille au plus h et de largeur au plus w, alors $a_{P_1} = O((h.2^w)^m)$, et donc le temps total de calcul de l'algorithme est $O((h.2^w)^m n)$.

Un algorithme linéaire en temps existe si P est défini à partir d'une seule carte physique. Ceci découle du fait que les anti-chaînes maximales de P_1 sont totalement ordonnées par P_1, c'est-à-dire qu'il est possible de calculer, pour toutes les anti-chaînes $\mathcal{Q} \in P_1$, le nombre maximum d'adjacences obtenu dans l'extension linéaire de $\downarrow \mathcal{Q}$ par un programme dynamique.

Heuristiques Deux heuristiques ont été proposées pour calculer le nombre maximum d'adjacences. La première heuristique a été proposée par Blin *et al.* [15] pour résoudre MAL-1OP. À partir d'un DAG représentant la fermeture transitive de l'ordre partiel (voir Section 2.1), l'algorithme consiste à trouver, d'une façon itérative, le chemin le plus valable direct ou indirect dans P_1, intégrant le DAG et le calcul de la fermeture transitive.

La deuxième heuristique, permettant cette fois de résoudre MAL-2OP, a été proposée par Fu et Jiang [43]. Cet algorithme utilise les approximations existant pour le problème Minimum Double Feedback Vertex Set.

MINIMUM DOUBLE FEEDBACK VERTEX SET

• **Entrée** : Deux graphes dirigés D_1 et D_2 avec le même ensemble de sommets S.

• **Question** : Trouver le sous-ensemble $S' \subseteq S$ de cardinalité minimale en supprimant les sommets de D_1 et D_2 acyclique.

Intervalles communs La mesure d'intervalles communs entre deux ordres partiels P_1 et P_2 est définie comme le nombre maximal d'intervalles communs entre une extension linéaire de P_1 et une extension linéaire de P_2. Ce problème, noté MICL-2OP, est **NP**-complet (voir Blin *et al.* [15]). De même que pour le nombre d'adjacences, nous définissons le problème où l'ordre d'un des génomes est entièrement connu : MICL-1OP.

MICL-1OP

• **Entrée** : Un ordre partiel P_1.

• **Sortie** : Une extension linéaire T_1 de P_1 qui maxime le nombre d'intervalles communs entre T_1 et Id.

MICL-2OP

• **Entrée** : Deux ordres partiels P_1 et P_2.

• **Sortie** : Deux extensions linéaires T_1 et T_2 de P_1 et P_2 respectivement qui maximisent le nombre d'intervalles communs entre T_1 et T_2.

1.5 Conclusion

Le génome est le support de l'information permettant aux êtres vivants de se développer et de fonctionner et qui est transmise aux descendants. Il est composé d'une suite de gènes. Nous avons vu (Section 1.1) que le génome évolue au cours du temps : des gènes se multiplient, évoluent, se déplacent, disparaissent.

La génomique comparative permet d'étudier les génomes (Section 1.2). Pour cela, des distances de réarrangements et des mesures de (dis)similarité sont proposées. Ainsi, nous pouvons comparer les génomes entre eux pour mettre ainsi en valeur les complexes de gènes, les mutations qui ont eu lieu, un ancêtre commun potentiel, etc.

Dans les chapitres suivants, nous cherchons à déterminer des mesures de (dis)similarité.

Premièrement et à la différence des premiers travaux effectués dans le domaine, nous prenons en compte, dans notre étude, les duplications de gènes. Nous devons donc créer des couplages (voir Section 1.3). Dans le Chapitre 3, nous présentons des résultats de complexité, des algorithmes exacts et des heuristiques permettant de calculer le nombre d'adjacences et le nombre de points de cassure entre deux génomes pour trois modèles différents : exemplaire, intermédiaire et maximum. L'ensemble de nos algorithmes est testé à l'aide de 12 génomes de γ-protéobactéries [59].

Nous avons vu dans la Section 1.4 que le séquençage et l'annotation des génomes ne permettent pas systématiquement d'obtenir des séquences de gènes totalement ordonnées. Dans le Chapitre 4, nous voulons calculer des génomes totalement ordonnés à l'aide de mesures de (dis)similarité. Plus précisément, nous étudions les problèmes MICL-1OP, MAL-1OP et MAL-2OP en proposant un algorithme exact pour chacun de ces problèmes. Ces algorithmes seront évalués à l'aide de génomes partiellement ordonnés simulés par [15].

Mais avant cela, nous présentons dans le Chapitre 2 différentes notions générales, indispensables pour la compréhension des prochains chapitres.

Chapitre 2

Pré-requis

Sommaire

❑ Pour une meilleure compréhension des chapitres suivants, nous allons aborder ici trois concepts importants : les graphes, la complexité des algorithmes et la programmation linéaire. Ce chapitre peut être lu de façon indépendante des autres.

Dans la section 2.1, nous donnons la définition d'un graphe et de quelques notions associées. Dans la section 2.2, les principales classes de complexité de problèmes algorithmiques sont vues. Cette classification nous permet de trouver des algorithmes adaptés pour la résolution de problèmes. Nous présentons aussi des méthodes permettant de connaître la classe de complexité d'un problème. Finalement, dans la section 2.3, nous allons présenter la programmation linéaire et en particulier la programmation linéaire à variables booléennes. Nous présentons certains des solveurs utilisés pour résoudre ces programmes.

2.1 Graphes

Un **graphe** $\mathcal{G}$ consiste en la donnée d'un ensemble S de **sommets** et un ensemble A d'**arêtes** ; nous le notons (S, A). Le nombre de sommets correspond à l'**ordre** du graphe. Une arête a appartenant à A est composée de deux sommets appartenant à S appelés **extrémités** : $a = \{s_1; s_2\}$. L'arête a est dite alors **incidente** au sommet s_1 et au sommet s_2 et les sommets s_1 et s_2 sont **connectés**. Nous appelons une **boucle** une arête dont les deux sommets sont confondus : par exemple $\{s_1; s_1\}$.

Une **chaîne** dans un graphe est une suite de sommets reliés par des arêtes. Une **chaîne simple** ne peut pas visiter le même sommet deux fois. Un graphe **connexe** est un graphe dans lequel chaque paire de sommets est reliée par une chaîne. Un **cycle** est une chaîne simple dont les extrémités coïncident. Un **cycle hamiltonien** est un cycle qui visite tous les sommets du graphe. Un graphe est dit **cyclique** s'il possède un cycle, sinon nous disons que ce graphe est **acyclique**.

Le graphe $\mathcal{G}$ est un graphe **orienté** si chaque arête de A est orienté : une arête est composée d'un sommet **initial** et d'un sommet **terminal**. Les arêtes d'un graphe orienté sont appelées **arcs** et sont notées (s_1, s_2) avec s_1 le sommet initial et s_2 le sommet terminal. Le sommet initial et le sommet terminal peuvent correspondre au même sommet s_1 (cas d'une boucle) : (s_1, s_1).

Un **chemin**, dans un graphe orienté, est une suite de sommets reliés les uns aux autres par des arcs. Un **chemin simple** ne peut pas visiter le même sommet plus d'une fois. Un **chemin fermé** a pour dernier sommet le premier. Un **circuit** est un chemin fermé simple. La **fermeture transitive** d'un graphe orienté $\mathcal{G}$ correspond au graphe $\mathcal{G}$ auquel nous ajoutons les arcs (s_1, s_2) si et seulement s'il existe un chemin débutant par s_1 et terminant par s_2.

Un **sous-graphe** induit par un ensemble $S' \subseteq S$ est obtenu en supprimant dans $\mathcal{G}$ tous les sommets dans $S \setminus S'$ ainsi que les arêtes incidentes. Un **stable** d'un graphe $\mathcal{G}$ est un sous-graphe de $\mathcal{G}$ sans arêtes.

Un **arbre** est un graphe connexe sans cycle. Un **DAG** ("Directed Acyclique Graph" en anglais) est un graphe acyclique orienté. Les DAG permettent de représenter des ordres partiels (voir Section 1.4).

La Figure 2.1 illustre ces différentes définitions.

De nombreux problèmes et algorithmes sont associés aux graphes, tels que l'algorithme de parcours en profondeur et les problèmes de COUVERTURE DE SOMMETS et de STABLE MAXIMUM.

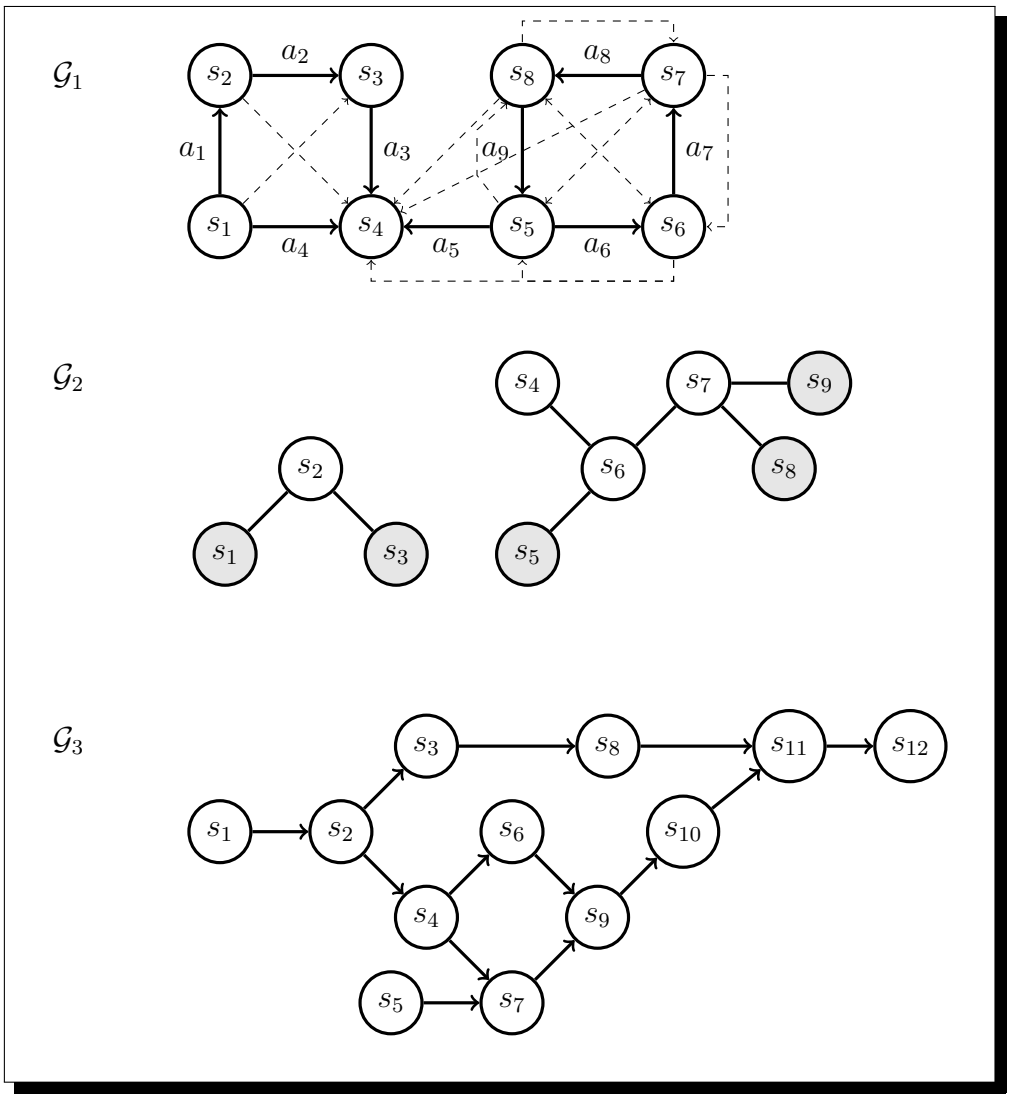


FIGURE 2.1 – **Définitions de graphes.** Le graphe $\mathcal{G}_1$ est un graphe *connexe* et *orienté* avec $S = \{s_1; s_2; \ldots; s_8\}$ et $A = \{a_1; a_2; \ldots; a_9\}$. L'ordre de $\mathcal{G}_1$ est 8. La *fermeture transitive* de $\mathcal{G}_1$ est le graphe $\mathcal{G}_1$ auquel nous ajoutons les 13 arêtes en pointillées. Dans $\mathcal{G}_1$, les sommets s_4 et s_5 sont *connectés* par l'arrête a_5 : le sommet initial est s_5 et le sommet terminal est s_4. La suite d'arêtes (a_7, a_8, a_9, a_5) est un *chemin*. La suite $(a_6, a_7, a_8, a_9, a_6)$ est un *circuit*. La suite (a_4, a_5, a_6) est une *chaîne*. La suite $(a_1, a_2, a_3, a_4, a_1)$ est un *cycle* (non *hamiltonien*) et donc $\mathcal{G}_1$ est un graphe *cyclique*. Le graphe $\mathcal{G}_2$ n'est ni connexe ni orienté. Les sommets grisés $(s_1, s_3, s_5, s_8, s_9)$ représentent les sommets d'un *sous-graphe* de $\mathcal{G}_2$. Ce sous-graphe est un *stable*. Le graphe $\mathcal{G}_3$ est *acyclique* et orienté : c'est donc un *DAG*.

Parcours en profondeur Le parcours en profondeur est un algorithme permettant d'explorer chaque sommet d'un graphe. La stratégie suivie est, comme son nom l'indique, de descendre plus "profondément" dans le graphe tant que cela est possible.

Cet algorithme utilise une coloration des sommets : une couleur pour les sommets "non explorés", une couleur pour les sommets "en cours d'exploration" et une couleur pour les sommets "entièrement explorés". Au départ, tous les sommets sont "non explorés".

Nous commençons à parcourir un graphe à partir d'un sommet s donné. Le sommet s est "en cours d'exploration". Nous parcourons tous les sommets voisins de s. Et nous procédons de même : ils sont colorés "en cours d'exploration" et nous parcourons leurs sommets voisins. Lorsqu'un sommet parcouru n'a pas de voisin "non exploré" il devient lui même "entièrement exploré". Lorsque le sommet de départ s est "entièrement coloré" nous réitérons le processus avec un nouveau sommet de départ "non-coloré".

Cet algorithme a une complexité polynomiale : $O(n + m)$ (n le nombre de sommets et m le nombre d'arêtes). Il permet, entre autre, de prouver si un graphe est connexe ou non.

Couverture de Sommets Le problème Couverture de Sommets ("Vertex Cover" en anglais) est le problème de décision défini ci-dessous.

Couverture de Sommets

- **Entrée** : Un graphe $\mathcal{G} = (S, A)$ et un entier positif k.
- **Question** : Existe-t-il un sous-ensemble de k sommets $S' \subseteq S$ tel que chaque arête de A soit incidente à au moins un sommet de S' ?

Stable Maximum Le problème Stable Maximum ("Maximal independent set" en anglais) dans un graphe est le problème d'optimisation défini ci-dessous.

Stable Maximum

- **Entrée** : Un graphe $\mathcal{G}$
- **Solution** : Un stable $\mathcal{G}'$ de $\mathcal{G}$
- **Mesure** : Maximiser l'ordre de $\mathcal{G}'$

2.2 Complexité

Un problème prend en entrée une **instance** et détermine un ensemble de solutions optimales en fonction de cette instance.

Il existe deux types de problèmes : les problèmes de **décision** (Couverture de Sommets par exemple) et les problèmes d'**optimisation** (Stable Maximum par exemple) [44]. Pour les problèmes de décision, la réponse attendue est *oui* ou *non*. Pour les problèmes d'optimisation, parmi l'ensemble des solutions respectant une propriété, la réponse attendue est celle qui optimise une fonction donnée. Selon le problème considéré, l'optimisation consiste soit à minimiser soit à maximiser une fonction (Définition 2.1).

Définition 2.1 *Un* problème d'optimisation *est défini comme un triplet* $(\text{sol}, \text{val}, \text{but})$ *tel que :*

- *pour chaque instance* x, $\text{sol}(x)$ *est l'ensemble des solutions possibles,*
- *la fonction* val *associe à chaque solution possible une valeur,*
- *il s'agit soit de maximiser* $(\text{but} = \max)$ *soit de minimiser* $(\text{but} = \min)$ *la fonction* val.

Un optimum *du problème d'optimisation est un élément* $y^* \in \text{sol}(x)$ *tel que* $\text{val}(y^*) = \max\{\text{val}(y) : y \in \text{sol}(x)\}$ *si* $\text{but} = \max$, *et* $\text{val}(y^*) = \min\{val(y) : y \in sol(x)\}$ *si* $\text{but} = \min$. *La valeur de la fonction* val *pour l'optimum est notée* $\text{opt}(x)$.

En pratique, beaucoup de problèmes sont des problèmes d'optimisation. Mais il est, en général, très facile de les transformer en problème de décision. Par exemple, le problème d'optimisation Stable Maximum peut être transformé en :

Problème de décision Stable Maximum

- **Entrée** : Un graphe $\mathcal{G}$ et un paramètre k.
- **Question** : Existe-t-il un stable $\mathcal{G}'$ de $\mathcal{G}$ à k sommets ?

Pour résoudre un problème, plusieurs **algorithmes** sont, en général, possibles. Un algorithme est une méthode qui permet de répondre à un problème. Les algorithmes prennent une instance x en entrée et cherchent à renvoyer le plus rapidement possible le résultat exact correspondant au problème associé (c'est-à-dire `oui` ou `non` pour les problèmes de décision et $opt(x)$ pour les problèmes d'optimisation). Ils sont caractérisés par une complexité en temps et une complexité en espace correspondant respectivement au temps et à l'espace utilisés dans le pire des cas. Ces complexités sont généralement

exprimées en ordre de grandeur. Pour simplifier, nous appellerons un algorithme de complexité $O(n^k)$ un **algorithme polynomial** où k est une constante et n est la taille de l'instance (paramètre d'entrée). Si $k = 1$ l'algorithme est spécifiquement appelé **algorithme linéaire**. Un **algorithme exponentiel** désigne les algorithmes de complexité au moins $O(2^n)$.

Il existe une classification des problèmes en fonction de leur complexité. Plus précisément l'appartenance d'un problème à une certaine classe se définit par la complexité des algorithmes permettant de le résoudre. C'est pourquoi connaître la classe de complexité d'un problème, donne clairement un avantage pour sa résolution (par exemple, il n'est pas judicieux de rechercher un algorithme polynomial si nous savons que le problème étudié est un problème **NP**-difficile).

Dans un premier temps, nous présentons les deux principales classes de complexité : **P** et **NP**. Puis nous verrons plus en détail les problèmes de la classe **NP**.

2.2.1 Les classes de complexité P et NP

La classe P

Définition 2.2 *Un problème* Π *est dans la classe* **P** *s'il existe un algorithme résolvant* Π *de façon exact en temps polynomial (c'est-à-dire en* $O(n^k)$*), sur une machine de Turing (voir l'encart Page 55).*

Un problème de la classe **P**, est un problème pour lequel il existe un algorithme polynomial. Ce problème est qualifié de problème polynomial. Pour prouver qu'un problème Π est dans la classe **P**, il suffit donc de trouver un algorithme polynomial le résolvant.
Exemple. Un parcours en profondeur, de complexité $O(n + m)$ (n le nombre de sommets et m le nombre d'arêtes), permet de prouver si un graphe est connexe. Nous en déduisons que *décider si un graphe est connexe* est un problème de la classe **P**.

Les problèmes polynomiaux sont les problèmes les plus simples à résoudre car nous savons qu'il existe toujours un algorithme exact et polynomial à la fois permettant de les résoudre. Pour chacun de ces problèmes, il reste juste à trouver un tel algorithme qui soit tout de même le moins coûteux possible en temps et en espace. En effet, en pratique, les algorithmes polynomiaux ne sont pas toujours utilisables si k est grand, ou pire, si n est grand. Par exemple, un algorithme en $O(n^2)$ est quasiment inutilisable si n est la taille d'un génome.

↻ Machine de Turing [44]

Une **machine de Turing** est un modèle abstrait du fonctionnement des appareils mécaniques de calcul, tel un ordinateur et sa mémoire, créé par Alan Turing [84] en vue de donner une définition précise au concept d'algorithme.
La mise en œuvre concrète d'une machine de Turing est réalisée avec les éléments suivants :

1. Un **ruban** divisé en cases consécutives. Chaque case contient un symbole parmi un alphabet fini. L'alphabet contient un symbole spécial ≪ blanc ≫ et un ou plusieurs autres symboles. Le ruban est supposé être de longueur infinie vers la gauche ou vers la droite, en d'autres termes la machine doit toujours avoir assez de longueur de ruban pour son exécution. On considère que les cases non encore écrites du ruban contiennent le symbole ≪ blanc ≫.
2. Une **tête de lecture/écriture** qui peut lire et écrire les symboles sur le ruban, et se déplacer vers la gauche ou vers la droite du ruban.
3. Un **registre d'état** qui mémorise l'état courant de la machine de Turing. Le nombre d'états possibles est toujours fini, et il existe un état spécial appelé ≪ état de départ ≫ qui est l'état initial de la machine avant son exécution.
4. Une **table d'actions** qui indique à la machine quel symbole écrire, comment déplacer la tête de lecture, et quel est le nouvel état, en fonction du symbole lu sur le ruban et de l'état courant de la machine. Si aucune action n'existe pour une combinaison donnée d'un symbole lu et d'un état courant, la machine s'arête.

Un calcul est constitué d'étapes élémentaires ; à chacune de ces étapes, pour un état donné de la mémoire de la machine, une action élémentaire est choisie dans un ensemble d'actions possibles. Les machines **déterministes** sont telles que chaque action possible est unique, c'est-à-dire que l'action à effectuer est dictée de façon unique par l'état courant de celle-ci. S'il peut y avoir plusieurs choix possibles d'actions à effectuer, la machine est dite **non déterministe**.

La classe NP

Définition 2.3 *L'ensemble des problèmes qui peuvent être résolus par une machine de Turing non déterministe définit la* classe **NP** *(Non-déterministe Polynomial). De façon équivalente, c'est la classe des problèmes qui admettent un algorithme polynomial capable de tester la validité d'une solution du problème, nous disons aussi capable de construire un* certificat.

Cette classe regroupe les problèmes pour lesquels nous pouvons vérifier une solution en temps polynomial mais nous ne connaissons pas d'algorithme polynomial permettant de calculer cette solution.
Exemple. Soient un graphe $\mathcal{G}$ et un cycle de ce graphe. Nous pouvons vérifier en temps polynomial si ce cycle est un cycle hamiltonien pour le graphe $\mathcal{G}$, autrement dit il existe un algorithme polynomial capable de construire un certificat pour le problème de recherche de cycle hamiltonien dans un graphe. Ce problème est donc un problème de la classe **NP**.

De façon évidente $\mathbf{P} \subseteq \mathbf{NP}$ mais nous ne savons pas si $\mathbf{P} \neq \mathbf{NP}$.

Dans la classe **NP**, il existe la classe des problèmes **NP**-difficiles.

Définition 2.4 *Un problème est* **NP**-difficile *si ce problème est au moins aussi difficile que tous les problèmes dans* **NP**. *Autrement dit, tout problème dans* **NP** *peut se réduire à lui.*

Pour décider si un problème est aussi difficile que tous les problèmes d'une classe, nous utilisons une réduction polynomiale.

Définition 2.5 *Soient* Π *et* Π' *deux problèmes. Une* réduction polynomiale *de* Π' *à* Π *est un algorithme de la classe* **P** *transformant toute instance de* Π' *en une instance de* Π. *Cette transformation est telle que pour toute instance de* Π *la réponse est* `oui` *si et seulement si la réponse au problème* Π' *est* `oui` *pour l'instance transformée.*

Ainsi, si nous avons un algorithme pour résoudre Π, *nous savons aussi résoudre* Π', *mais de plus, si la complexité de* Π *est au moins celle de la classe* **NP**, *nous pouvons dire que* Π *est au moins aussi difficile à résoudre que* Π'. Π *est alors* **NP**-difficile *si pour tout problème* Π' *de* **NP**, Π' *se réduit à* Π.

Définition 2.6 *Nous disons qu'un problème est* **NP**-*complet s'il est dans* **NP** *et s'il est* **NP**-difficile.

Exemple. Le problème 3-SAT (voir l'encart Page 57) appartient à la classe **NP** car il existe un algorithme polynomial trivial capable de tester la validité d'une solution. Maintenant, pour prouver que 3-SAT est un problème **NP**-complet, il faut prouver qu'un

problème est polynomialement réductible à 3-SAT. Prenons comme problème polynomialement réductible le problème SAT.

Pour cela, nous transformons toute clause $v_1 \vee v_2 \vee \ldots \vee v_n$ en les clauses :

$$(v_1 \vee v_2 \vee u_1) \wedge (v_3 \vee \neg u_1 \vee u_2) \wedge \ldots \wedge (v_{n-2} \vee \neg u_{n-4} \vee u_{n-3}) \wedge (v_{n-1} \vee v_n \vee \neg u_{n-3})$$

Ces clauses sont satisfaisables si et seulement si la clause $v_1 \vee v_2 \vee \ldots \vee v_n$ est satisfaisable. Donc le problème SAT est polynomialement réductible à 3-SAT. Nous concluons que 3-SAT est **NP**-complet.

SAT

Le problème SAT est un problème de décision qui prend en entrée m clauses dépendant de n variables booléennes v_i ($i \in [1 : n]$). Il renvoie `oui` s'il existe un ensemble de valeurs pour les n variables v_i (une valuation) qui satisfait les m clauses, sinon il renvoie `non`.
Une clause est une proposition de la forme $v_1 \vee v_2 \vee \ldots \vee v_n$ où les v_i sont des littéraux (positifs ou négatifs).
Une clause d'ordre n est donc une disjonction d'au plus n littéraux. Une formule du calcul propositionnel est en forme normale conjonctive (CNF) (ou forme clausale) d'ordre n si elle est une conjonction de clauses d'ordre n.
Le problème SAT est un problème **NP**-complet d'après le théorème de Cook [30]. C'est le premier problème pour lequel la **NP**-Complétude a été prouvée.

Le problème 3-SAT correspond au problème SAT pour les instances d'ordre 3.

2.2.2 Abaisser les exigences

Pour les problèmes **NP**-difficiles, nous possédons uniquement des algorithmes exacts au moins exponentiels. Pour des "petites" instances ces algorithmes peuvent être suffisants. Mais souvent, il est plus réaliste d'abaisser les exigences : soit en perdant l'exactitude des résultats soit en modifiant le problème de départ.

Exactitude non assurée

Pour un problème d'optimisation, il peut être difficile d'obtenir de façon exacte un résultat. Une solution est de chercher des algorithmes rapides en pratique et qui s'approchent le plus près possible du résultat exact. Deux types d'algorithmes existent alors : les algorithmes d'approximation (dont le rapport avec le résultat exact est borné) et les heuristiques (dont le résultat peut-être arbitrairement loin du résultat exact mais qui tentent de fournir de bonnes solutions en pratique).

Approximation

Définition 2.7 *Un* algorithme approche à ϵ près *un problème d'optimisation s'il calcule pour toute entrée x une solution y telle que :*

$$\max\left\{\frac{val(y)}{opt(x)}, \frac{opt(x)}{val(y)}\right\} \leq \epsilon$$

Un problème d'optimisation est ϵ-approximable *s'il existe un algorithme polynomial qui l'approche à ϵ près pour un ϵ tel que $\epsilon < 1$. Le réel ϵ est le* rapport *d'approximation.*

Il existe, entre autres, les deux classes de problèmes approximables suivantes. La classe **APX** est la classe des problèmes pour lesquels il existe un algorithme dont le rapport d'approximation est une constante. Parmis les problèmes de la classe **APX**, les problèmes possédant un schéma d'approximation définissent la classe des problèmes **PTAS**. Dans les deux cas, nous n'obtenons pas obligatoirement le résultat exact mais nous pouvons évaluer la différence entre notre résultat d'approximation et le résultat exact. Plus précisément, la classe **PTAS** est la classe des problèmes qui sont ϵ-approximables pour tout ϵ - voir Définition 2.8. La classe **APX** est la classe des problèmes **NP** qui sont ϵ-approximables pour un ϵ constant - voir Définition 2.9.

Définition 2.8 *Un problème d'optimisation Π appartient à la classe* **PTAS** *("Polynomial-Time Approximation Scheme" en anglais) si, pour tout $\epsilon > 0$, il existe un algorithme A polynomial, tel que pour toute instance x de Π, nous ayons :*
- $A(x) \leq \epsilon.opt(x)$ *si Π est un problème de minimisation,*
- $A(x) \geq opt(x)/\epsilon$ *si Π est un problème de maximisation.*

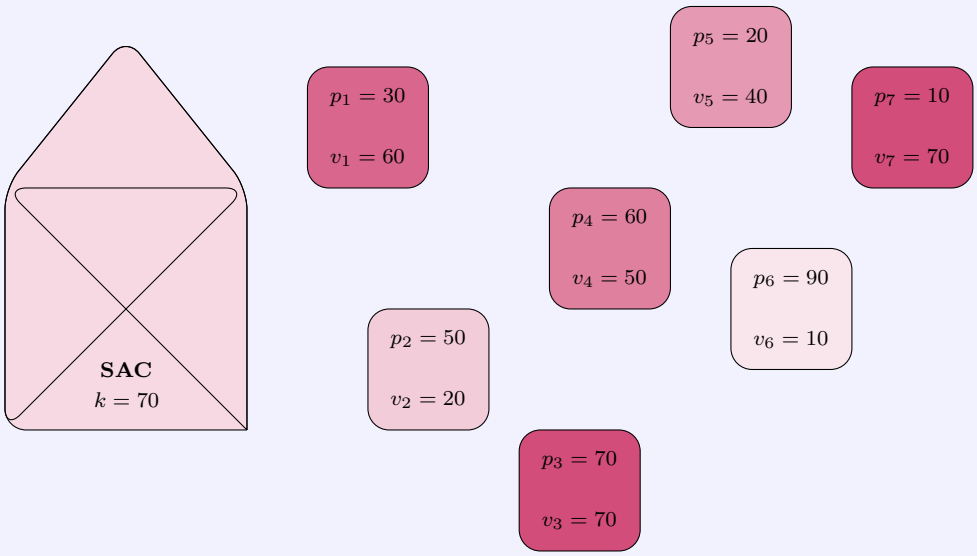
Pour les problèmes **PTAS**, il existe un algorithme permettant de s'approcher aussi près du résultat optimal que nous voulons mais nous devons en payer le prix en temps.
Exemple. Le problème Sac À Dos (voir l'encart Page 59) est ϵ-approximable pour tout ϵ.

Définition 2.9 *Un problème d'optimisation Π appartient à la classe* **APX** *s'il existe une constante ϵ et un algorithme A polynomial tel que pour toute instance x de Π, nous ayons :*
- $A(x) \leq \epsilon.opt(x)$ *si Π est un problème de minimisation,*
- $A(x) \geq opt(x)/\epsilon$ *si Π est un problème de maximisation.*

Les problèmes **APX**, possèdent des algorithmes nous permettant de nous approcher du résultat exact jusqu'à une certaine limite.
Exemple. Le problème de Couverture de Sommets est 2-approximable.

Le problème **Sac à Dos**

Le problème du SAC À DOS, aussi noté KP (KNAPSACK PROBLEM en anglais) est un problème d'optimisation **NP**-complet. De façon imagée, il prend en entrée un ensemble de n objets, possédant chacun un poids p_i et une valeur v_i ($i \in [1 : n]$), et un sac permettant de porter un certain poids k. Le but est de transporter dans le sac le plus de valeurs possibles ; c'est-à-dire de trouver la combinaison d'objets dont la somme des valeurs est maximale et dont la somme des poids est inférieure ou égale à k.



Propriété 2.10 *Soit Π et Π_0 deux problèmes possédant le même ensemble d'instances. Soit Π un problème d'optimisation qui cherche à minimiser une mesure m et Π_0 le problème de décision qui cherche à savoir si la mesure m optimisée est nulle. Si résoudre le problème Π_0 est un problème* **NP***-complet alors le problème Π n'admet aucune approximation en temps polynomial, sauf si* **P**$=$ **NP**.

Preuve. Procédons par l'absurde. Nous supposons qu'il existe un algorithme d'approximation polynomial qui résoud Π tel que $\epsilon < 1$. Nous avons donc :

$$\frac{val(y)}{opt(x)} \leq \epsilon.$$

Pour les instances où $opt(x) = 0$, nous avons :

$$val(y) \leq \epsilon.opt(x).$$

Donc $val(y) = 0$. Nous avons donc un algorithme polynomial qui permet de résoudre le problème Π_0. Ce qui est absurde (à moins que $\mathbf{P} = \mathbf{NP}$) puisque Π_0 est **NP**-complet. □

Heuristique Pour un problème d'optimisation donné, une heuristique est un algorithme, rapide en pratique, qui fournit une solution qui n'est pas nécessairement la solution optimale. À la différence des algorithmes d'approximation, l'écart entre la solution optimale et le résultat d'une heuristique n'est pas borné.

Ces algorithmes sont validés, ou non, s'ils sont réellement rapides et si leur résultat est proche du résultat exact. Par proche nous signifions qu'en moyenne la différence entre le résultat exact et le résultat heuristique est proche de zéro et/ou que le nombre d'instances, pour lequel le résultat heuristique est identique au résultat exact, est grand.

Pour évaluer correctement une heuristique, il faut donc comparer leurs résultats avec ceux que nous pouvons obtenir de façon exacte.

Exemple. Voici une heuristique pour résoudre le problème Couverture de Sommets. Nous sélectionnons un des sommets dont le nombre d'arrêtes incidentes est maximal. Nous le marquons. Puis nous supprimons toutes les arêtes incidentes à ce sommet. Nous réitérons le processus, avec un sommet non-marqué, tant qu'il reste des arêtes. Au final, l'ensemble des sommets marqués est une couverture de sommets. Cette couverture n'est pas obligatoirement minimale. De plus, le résultat peut-être arbitrairement loin du résultat optimal [69].

Modification du problème

Réduction d'instances Un problème **NP**-complet peut être facile à résoudre pour certaines instances. En réduisant un problème à ces instances faciles, nous arrivons à obtenir des résultats exacts. Nous avons alors un nouveau problème qui peut être d'une classe de complexité différente.

Exemple. Calculer le nombre de points de cassures entre deux génomes en prenant en compte les duplications est un problème **NP**-complet (voir Section 1.3.1). Si nous réduisons les instances aux génomes possédant aucun gène dupliqué, le problème devient plus facile à résoudre. Il change même de classe de complexité : le nouveau problème est en effet dans la classe **P** (voir Section 1.2.2).

Tranformation Une technique utilisée pour résoudre un problème est de transformer les instances de notre problème Π en des instances d'un problème Π'. Nous résolvons alors le problème Π' puis nous traduisons le résultat pour répondre au problème Π. Il est important que la transformation et la traduction soient rapides (polynomiaux). L'intérêt de ce procédé est de pouvoir s'appuyer sur les études déjà effectuées sur le problème Π' telles que des algorithmes exacts, des heuristiques ou des algorithmes d'approximation performants. Ainsi nous obtenons de meilleurs et/ou plus de résultats.

Le problème n'a pas changé, il ne change donc pas de classe.

Exemple. Le problème Stable Maximum peut être modélisé en un programme linéaire à variables entières (voir Section 2.3). Nous pouvons donc utiliser les algorithmes déjà existant pour la résolution de programmes linéaires à variables entières afin de résoudre le problème Stable Maximum. La solution du programme linéaire nous donne immédiatement la solution du problème Stable Maximum.

Problème paramétré L'idée fondamentale de la paramétrisation est d'essayer d'obtenir une meilleure vision de la complexité d'un problème en explorant comment certains paramètres spécifiques du problèmes influencent cette complexité. Idéalement, un paramètre k est cherché tel que si k est petit alors le problème peut être résolu efficacement.

Définition 2.11 *Soit Σ^* un alphabet fini. Une* paramétrisation *de Σ^* est une fonction $\mathcal{K} : \Sigma^* \to \mathbb{N}$ calculable en temps polynomial. Un* problème paramétré *(sur Σ) est une paire $(\mathcal{Q}, \mathcal{K})$ où $\mathcal{Q} \subseteq \Sigma^*$ est un ensemble de mots sur Σ et $\mathcal{K}$ une paramétrisation de Σ^*. Si $(\mathcal{Q}, \mathcal{K})$ est un problème paramétré sur l'alphabet Σ, nous appellons* instance *de $(\mathcal{Q}, \mathcal{K})$ un mot x de Σ^* et* paramètre *le nombre $\mathcal{K}(x)$ associé.*

Un algorithme **A** est un algorithme **FPT** relativement à $\mathcal{K}$ s'il existe une fonction $f : \mathbb{N} \to \mathbb{N}$ telle que pour tout $x \in \Sigma^*$, le temps d'exécution de **A** sur l'entrée x est au plus $f(\mathcal{K}(x)).|x|^{O(1)}$.

Un problème paramétré $(\mathcal{Q}, \mathcal{K})$ est **FPT**, c'est-à-dire soluble à paramètre fixé (*fixed-parameter tractable* en anglais), s'il existe un algorithme **FPT** relativement à $\mathcal{K}$ qui reconnaît $\mathcal{Q}$.

Exemple. Le problème Couverture De Sommets se traduit par le problème paramétré suivant :

Couverture De Sommets (paramétré)

- **Entrée** : Un graphe $\mathcal{G}$ et un entier k.
- **Paramètre** : k.
- **Question** : $\mathcal{G}$ a-t-il une couverture de sommets de taille au plus k ?

Le problème de couverture minimal est **FPT** puisqu'il existe un algorithme qui a un temps d'exécution de $2^{v(\mathcal{G})+1}.n^{O(1)}$ (avec $v(\mathcal{G})$ la cardinalité de la couverture recherchée dans $\mathcal{G}$ et n le nombre de sommets de $\mathcal{G}$).

Clairement, si un problème est polynomial alors, pour toute paramétrisation $\mathcal{K}$, le problème paramétré $(\mathcal{Q}, \mathcal{K})$ est **FPT**.

2.3 Programmation linéaire

❑ Pour résoudre, de façon exacte, des problèmes de comparaison de génomes (voir Section 1.2), nous allons, dans les chapitres suivants, les modéliser sous la forme de programmes linéaires.

2.3.1 La programmation linéaire à variables réelles

La programmation linéaire permet de résoudre des problèmes d'optimisation. Elle s'écrit sous la forme d'un **objectif** et de **contraintes**, eux mêmes constitués de **variables**. Le but est de donner des valeurs aux variables de manière à optimiser l'objectif tout en respectant les contraintes. L'objectif est soit de minimiser, soit de maximiser une somme. La programmation linéaire a de nombreuses applications telles que la gestion du flux de transport, la planification de production, la distribution dans des réseaux, les télécommunications, etc.

Nous reportons ci-dessous l'exemple pertinent de [38].

Exemple. Une compagnie pétrolière alimente quotidiennement en carburant les stations services de son réseau. Ces carburants se composent d'un mélange de plusieurs pétroliers, de provenance et de qualités divers. Les cours de ces produits sont extrêmement fluctuants, et sont publiés chaque jour. Chaque matin, la compagnie doit déterminer, compte tenu des cours du jour, la composition du mélange qu'elle livrera, de façon à minimiser son prix de revient ; le carburant obtenu doit cependant satisfaire certains critères de qualités.

Afin de fixer les idées, simplifions le problème à l'extrême. Supposons que la compagnie ne livre qu'un seul type de carburant, la même quantité chaque jour, et que seuls trois

produits - P_1, P_2 et P_3, dont les stocks disponibles peuvent être considérés comme illimités, interviennent dans sa composition, selon des proportions (à déterminer) x_1, x_2 et x_3. Soient c_1, c_2 et c_3 les coûts unitaires respectifs de P_1, P_2 et P_3 : le prix de revient à minimiser est proportionnel à :

$$z = c_1x_1 + c_2x_2 + c_3x_3.$$

Le carburant livré doit cependant présenter un taux d'octane minimum b_1. Notons a_{11}, a_{12} et a_{13} les taux d'octane respectifs de P_1, P_2 et P_3 : les proportions x_1, x_2 et x_3 doivent donc satisfaire à la contrainte :

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \geq b_1.$$

Supposons d'autre part que la teneur en plomb tétraéthyle ne puisse descendre en-dessous d'une certaine limite b_2 ; en notant a_{21}, a_{22} et a_{23} les teneurs en plombs respectives de P_1, P_2 et P_3, nous obtenons une seconde contrainte :

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \geq b_2.$$

Les proportions x_1, x_2 et x_3, bien entendu, doivent être non négatives et former un total de 1 :

$$x_1 + x_2 + x_3 = 1$$

$$x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0$$

Rassemblons les éléments du problème :

$$\min(z) = c_1x1 + c_2x_2 + c_3x_3 \tag{2.1}$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &\geq b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &\geq b_2 \qquad (2.2) \\ x_1 + x_2 + x_3 &= 1 \\ x_i &\geq 0 \quad i = 1, 2, 3. \qquad (2.3) \end{aligned}$$

L'ensemble formé de (2.1), (2.2) et (2.3) s'appelle un *programme linéaire* : la fonction économique z (2.1) et le système de contraintes (2.2) s'expriment en effet linéairement en fonction des variables structurelles x_1, x_2, et x_3. Les contraintes (2.3) sont appelées *contraintes de positivité*.
Cet exemple possède 3 variables et 6 contraintes. Mais il existe des programmes linéaires avec plusieurs miliers de variables et plusieurs milliers de contraintes.

Définition 2.12 *Un programme linéaire (PL) est un problème du type :*

(problème à maximum)	*(problème à minimum)*
$\max(z) = \sum_{j=1}^{n} c_j x_j$	$\min(z) = \sum_{j=1}^{n} c_j x_j$
sous les contraintes	*sous les contraintes*
$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i =, \ldots, m$	$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i =, \ldots, m$
$x_j \geq 0, \quad j = 1, \ldots, n$	$x_j \geq 0, \quad j = 1, \ldots, n$

où les coefficients et variables c_j, a_{ij}, b_i et x_i appartiennent à l'ensemble $\mathbb{R}$ des nombres réels.

Une solution $(x_1, \ldots, x_n)$ est **réalisable** si elle satisfait à toutes les contraintes. Elle est **optimale** si elle est réalisable et si elle optimise l'objectif. Un programme peut avoir une, plusieurs ou aucune solution optimale (si l'objectif n'est pas borné ou si les contraintes se contre-disent et le PL est alors insatisfaisable).

Dans le cas où les variables sont réelles plusieurs algorithmes ont été proposés. En 1951 Dantzig [31] propose l'algorithme `simplex` qui est exponentiel mais qui s'avère en pratique efficace. Depuis d'autres algorithmes ont été proposés ; à noter l'algorithme polynomial de Leonid Khachyan de 1979 [57]. En théorie il est plus efficace que `simplex` mais ce n'est pas le cas en pratique. En 1984, Karmarkar [55] propose une méthode projective efficace en théorie (polynomiale) et comparable à `simplex` en pratique. C'est une méthode de *point intérieur* : la solution candidate courante ne suit pas les bornes de l'espace faisable, comme dans l'algorithme `simplex`, mais approche par l'intérieur de l'espace faisable et atteint la solution optimale de manière asymptotique.

2.3.2 La programmation linéaire à variables entières

Pour modéliser des problèmes d'optimisation de type graphe ou ensemble il est souvent nécessaire d'ajouter des contraintes d'intégralités : les variables du programme doivent être des variables entières. Alors que les programmes linéaires à variables réelles sont dans la classe **P**, les programmes linéaires à variables entières (PLE) sont **NP**-complet [44]. Nous utilisons un PLE pour résoudre des problèmes tel que le problème de stabilité dans un graphe ou le problème Sac à Dos (voir l'encart Page 59).

L'étude de PLE a débuté à la fin des années 50 avec Gomory [48]. Aujourd'hui, nous disposons principalement de deux types de méthodes pour résoudre un PLE : les méthodes d'optimisation par coupe et les méthodes par séparation et évaluation.

Méthodes d'optimisation par coupe À partir du PL correspondant à un PLE (les variables entières deviennent des variables réelles), nous obtenons une solution optimale

pour le PL. Si celle-ci ne correspond pas à une solution du PLE, nous ajoutons une contrainte d'inégalité - nous effectuons une coupe - permettant d'éliminer cette solution dont les variables ne sont pas entières. Avec cette contrainte supplémentaire, nous obtenons un "PL augmenté". Nous itérons ce processus jusqu'à obtenir une solution entière. Il existe plusieurs algorithmes utilisant cette méthode, les coupes de Gomory en particulier [48].

Méthodes d'optimisation par séparation et évaluation Dans un PLE, les solutions sont en nombre fini. Il est donc possible de résoudre un PLE ainsi :

- énumérer les solutions possibles,
- garder les solutions réalisables, et
- calculer pour chacune d'elles l'objectif correspondant et garder celle qui l'optimise.

Bien sûr il y a généralement énormément de solutions possibles ; pour 15 variables et 10 contraintes il y a possiblement plus de 3 millions de solutions possibles. Mais grâce à différentes techniques nous évitons d'énumérer toutes les solutions possibles.

Une des techniques utilisées est l'optimisation par *séparation puis évaluation*. Dans un premier temps, lors de l'étape de séparation, un arbre dont les sommets sont des sous-ensembles de solutions est construit. Dans un deuxième temps, cet arbre est évalué de manière à trouver une solution optimale en évitant de le parcourir entièrement. À chaque sommet une fonction d'évaluation est utilisée permettant ainsi de connaître la meilleure solution en fonction du sous-ensemble en question. Selon le résultat de cette fonction les sommets de l'arbre restants sont parcourus d'une façon particulière.

Il existe un cas particulier de PLE : les programmes linéaires à variables booléennes. Les variables sont toujours entières, plus précisément elles sont booléennes : 0 ou 1. Cette restriction est couramment utilisée et nous voyons apparaître des solveurs spécialisés dans la résolution de ce type de programme. L'obligation de variables booléennes, et non seulement de variables entières, est très utile pour modéliser des problèmes combinatoires. Cette exigence nous permet d'utiliser des techniques spécifiques et efficaces. Nous pouvons, en particulier, utiliser les travaux utilisés pour résoudre les problèmes de type SAT.

2.3.3 Deux solveurs en particulier

Afin de résoudre des programmes linéaires (à variables entières ou non), il existe des méthodes de résolution plus ou moins efficaces, en théorie et en pratique. Pour chacune de ces méthodes, des programmes sont proposés.

Il est important de choisir celui qui sera le plus adapté pour résoudre nos propres programmes linéaires : `CPLEX`[1], `MiniSat+`[39], `pueblo`[80], `sat4jPseudo`[14], `bsolo`[64] etc. En effet, en pratique nous observons qu'un solveur peut-être peu coûteux en temps

1. http ://www.ilog.com/products/cplex/

et en mémoire spécifiquement pour certaines familles de programmes et au contrainte très coûteux pour d'autres programmes ; et inversement pour un autre solveur. Dans notre étude, nous avons utilisé les deux solveurs suivants : `CPLEX` et `MiniSat+`. Ces deux solveurs se sont montrés efficaces pour résoudre spécifiquement une famille de programmes linéaire à variables booléennes (voir Chapitre 3 pour `CPLEX` et Chapitre 4 pour `MiniSat+`).

CPLEX est un logiciel d'optimisation d'ILOG qui a été développé initialement par Robert E. Bixby. Il est nommé ainsi car au départ il se basait sur le même principe que l'algorithme du simplex et utilisait le langage C pour sa programmation. Aujourd'hui, il contient aussi des méthodes de point intérieur et il utilise d'autres langages de programmation.

`CPLEX` résout des problèmes de programmation linéaire, de programmation linéaire à variables entières, de programmation quadratique, et de contraintes quadratiques convexes. *Les problèmes de programmation quadratique sont des problèmes d'optimisation où la fonction objectif est quadratique et les contraintes sont linéaires.* En 2004, `CPLEX` a obtenu le premier prix d'INFORMS Impact [2].

Ci-dessous un exemple de fichier d'entrée de `CPLEX`. Ce fichier doit avoir l'extension `lp`. Il se décompose en trois parties : l'objectif, les contraintes et les variables (précédées de leur type, ici booléen). Les commentaires sont précédés du caractère "–".

```
- l'objectif :
min
2 x_1 + 4 x_2 + 2 x_3 + 2 x_4 + 3 x_5

- les contraintes :
st
x_1 + 2 x_2 - 2 x_3 + 4 x_4 + x_5 >= 3
x_1 - x_2 + x_4 + 3 x_5 >= 1
-2 x_1 + 2 x_2 + x_3 - 4 x_4 + 4 x_5 >= 1

- les variables :
binaries - les variables sont booléennes
x_1
x_2
x_3
x_4
x_5
end
```

2. http ://www.informs.org/article.php ?id=1290&p=140|

CPLEX renvoie alors le fichier ci-après.

```
Tried aggregator 1 time.
MIP Presolve eliminated 1 rows and 1 columns.
MIP Presolve modified 2 coefficients.
Reduced MIP has 2 rows, 4 columns, and 8 nonzeros.
Presolve time =    0.00 sec.
Clique table members: 2
MIP emphasis: balance optimality and feasibility
Root relaxation solution time =    0.00 sec.

        Nodes                                          Cuts/
 Node  Left Objective  IInf  Best Integer  Best Node  ItCnt    Gap
Variable B Parent  Depth

    0     0    4.0000     1                   4.0000      1
*   0+    0               0        7.0000     4.0000      1 42.86%

Solution status = Optimal
Solution value  = 7.0
Variable x_1 has vals 0.0
Variable x_2 has vals 0.0
Variable x_3 has vals 1.0
Variable x_4 has vals 1.0
Variable x_5 has vals 1.0
```

La solution optimale obtenue est 7. Les variables x_3, x_4 et x_5 sont égales à 1 et les variables x_1 et x_2 sont égales à 0.

MiniSat+ est un logiciel spécialisé dans la résolution des programmes linéaires à variables booléennes [39]. Pour cela il utilise une traduction des contraintes vers la forme normale conjonctive (CNF) - voir l'encart Page 57. Les formules CNF sont ensuite données en entrée au solveur `MiniSat+`. La résolution du problème se fait donc à l'aide d'un solveur SAT. Lors de l'évaluation à la compétition SAT de 2005 et de 2007[3], `MiniSat+` a obtenu de très bons résultats. Les sources de `MiniSat+` sont libres d'accès.

Ci-dessous un exemple de fichier d'entrée de `MiniSat+`. Il se compose de deux parties : l'objectif et les contraintes. À la différence de `CPLEX`, il est inutile de préciser le domaine des variables puisqu'elles sont obligatoirement booléennes. Les commentaires sont précédés du caractère "*".

3. http ://www.satcompetition.org/

```
* L'objectif :
min: 2*x_1 + 4*x_2 + 2*x_3 + 2*x_4 + 3*x_5;

* Les contraintes :
1*x_1 + 2*x_2 - 2*x_3 + 4*x_4 + 1*x_5 >= 3;
1*x_1 - 1*x_2 + 1*x_4 + 3*x_5 >= 1;
-2*x_1 + 2*x_2 + 1*x_3 - 4*x_4 + 4*x_5 >= 1;
```

MiniSat+ renvoie alors le fichier suivant.

```
c Parsing PB file...
c Converting 3 PB-constraints to clauses...
c   -- Unit propagations: (none)
c   -- Detecting intervals from adjacent constraints: (none)
c   -- Clauses(.)/Splits(s): (none)
c ---[   2]---> BDD-cost:    6
c ---[   1]---> BDD-cost:    4
c ---[   0]---> BDD-cost:    6
c ============================[MINISAT+]==========================
c |Conflicts |Original         |Learnt                   |Progress|
c |          |Clauses Literals |Max Clauses Literals LPC |        |
c ================================================================
c |        0 |     27       73 |  9       0        0 nan |0.0 %   |
c ================================================================
c Found solution: 9
c ---[   0]---> BDD-cost:    7
c ============================[MINISAT+]==========================
c |Conflicts |Original         |Learnt                   |Progress|
c |          |Clauses Literals |Max Clauses Literals LPC |        |
c ================================================================
c |        0 |     38      104 | 12       0        0 nan |0.0 %   |
c ================================================================
c Found solution: 7
c ---[   0]---> BDD-cost:    4
c ============================[MINISAT+]==========================
c |Conflicts |Original         |Learnt                   |Progress|
c |          |Clauses Literals |Max Clauses Literals LPC |        |
c ================================================================
c |        1 |     50      136 | 16       1        2 2.0 |0.0 %   |
c ================================================================
```

```
c Optimal solution: 7
s OPTIMUM FOUND
v -x_1 x_2 -x_3 -x_4 x_5
c ----------------------------------------------------------------
c
c restarts              : 3
c conflicts             : 3              (3003 /sec)
c decisions             : 23             (23023 /sec)
c propagations          : 0              (0 /sec)
c inspects              : 0              (0 /sec)
c CPU time              : 0.000999 s
c ----------------------------------------------------------------
```

La solution optimale obtenue est 7. Cette fois, les variables x_2, x_5 sont égales à 1 et les variables x_1, x_3 et x_4 sont égales à 0.

Il est difficile de se rendre compte avec un si petit exemple s'il peut être plus avantageux d'utiliser tel ou tel solveur. Pourtant, il est tout de même important de noter que le choix du solveur est une étape fondamentale dans la résolution d'un programme linéaire. Chaque solveur utilise une méthode spécifique. Par conséquence, un même programme linéaire (à variables booléennes ou non) peut être résolu en quelques secondes ou après plusieurs heures selon le solveur utilisé.

Chapitre 3

Comparaison de génomes avec duplications

Sommaire

Comme nous l'avons vu dans le Chapitre 1, la génomique comparative est un outil important pour une meilleure compréhension des génomes. Nous nous penchons dans cette section sur le problème du calcul de mesures entre deux génomes en prenant en compte leurs gènes dupliqués. Plus particulièrement, nous étudions les deux mesures de (dis)similarités suivantes : le nombre *d'adjacences* et le nombre de *points de cassure* [21]. Pour prendre en compte les gènes dupliqués lors du calcul de ces mesures, nous devons

désambiguïser les gènes homologues. Pour cela, nous avons besoin de créer un couplage entre les génomes. L'ensemble des notions et notations utilisées dans ce chapitre ont été introduites dans la Section 1.3.

Il existe deux modèles standard de couplage : le modèle *exemplaire* [75] et le modèle *maximum* [81]. Nous introduisons dans ce travail un troisième modèle qui se veut un compromis entre les deux modèles susmentionnés : le modèle *intermédiaire* [2]. Notons que, pour les modèles exemplaire et maximum, le calcul du nombre de points de cassure et le calcul du nombre d'adjacences sont des problèmes **NP**-complets [21, 16, 17, 18]. D'après ces résultats de complexité, nous pouvons aisément déduire que, pour le modèle intermédiaire, calculer ces mesures est un problème **NP**-complet.

Dans la Section 3.1, nous précisons la complexité des problèmes qui consiste à calculer le nombre minimum de points de cassure pour chacun des trois modèles (exemplaire, intermédiaire et maximum). Pour cela, nous étudions les problèmes de décision suivants : existe-t-il un couplage, entre deux génomes avec des duplications, tel que le nombre de points de cassure soit nul. Si un de ces problèmes est **NP**-complet alors le problème du calcul du nombre de points de cassure minimal, pour le modèle correspondant, n'admet pas d'approximation en temps polynomial (voir Propriété 2.10).

Puis dans la Section 3.2, nous présentons des algorithmes permettant de calculer le nombre d'adjacences maximal et le nombre de points de cassure minimal pour les trois modèles (exemplaire, intermédiaire et maximum). Ces algorithmes sont soit des algorithmes exacts (basés sur de la programmation linéaire à variables booléennes) soit des algorithmes heuristiques (basés sur la recherche des plus Longues Sous-séquences Communes).

Finalement, dans la Section 3.3, nous étudions les résultats obtenus par ces algorithmes sur des données de γ-protéobactéries. À l'aide de ces résultats, nous pouvons alors (i) faire une évaluation de nos algorithmes exacts et heuristiques, (ii) étudier nos trois modèles (exemplaire, intermédiaire et maximum) - en particulier le modèle intermédiaire.

3.1 Nombre de points de cassure nul

Cette section est entièrement consacrée à l'identification d'instances à zéro point de cassure. L'ensemble des travaux présentés dans cette section est le fruit d'une collaboration avec Sébastien Angibaud, Guillaume Fertin, Irena Rusu et Stéphane Vialette [4].

Dans [28], les auteurs ont montré que *déterminer s'il existe un couplage entre deux génomes, pour le modèle exemplaire, tel que le nombre de points de cassure est nul* est **NP**-complet même si tout gène est présent au plus trois fois dans chaque génome, c'est-à-dire les instances de type $(3, 3)$. Cet important résultat implique que le problème de minimisation du nombre de points de cassure pour le modèle exemplaire n'admet aucune approximation en temps polynomial, sauf si $\mathbf{P} = \mathbf{NP}$ (voir Propriété 2.10).

Poursuivant cette ligne de recherche, nous commençons par compléter le résultat de [28] en prouvant que décider s'il existe un couplage exemplaire entre deux génomes tel que le nombre de points de cassure est nul est **NP**-complet, même si tout gène est dupliqué au plus deux fois dans un des génomes (le nombre maximum de duplications est cependant illimité dans l'autre génome). Ce résultat sera par la suite étendu au modèle intermédiaire. Par ailleurs, nous donnerons un algorithme utilisable en pratique - mais exponentiel - pour décider s'il existe un couplage exemplaire entre deux génomes tel que le nombre de points de cassure est nul dans le cas où chaque gène est présent au plus deux fois dans chaque génome. Depuis, il a été montré que ce problème est **NP**-complet [19].

Finalement, nous montrons que décider s'il existe un couplage maximum, entre deux génomes tel que le nombre de points de cassure est nul est résoluble en temps polynomial. Et donc de tels résultats d'approximation négatifs (comme ceux que nous obtenons pour les modèles exemplaire et intermédiaire) ne se propagent pas au cas du modèle maximum.

L'observation suivante nous sera très utile dans la suite de cette section.

Observation 3.1 *Soient G_1 et G_2 deux génomes. Si le nombre de points de cassure, pour le modèle exemplaire, entre G_1 et G_2, est nul, alors il existe un couplage $\mathcal{C}$ de G_1 et G_2 tel que*

1. $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$, *ou*
2. $-(G_1^{\mathcal{C}})^r = G_2^{\mathcal{C}}$, *où* $-(G_1^{\mathcal{C}})^r$ *est une inversion signée du génome* $G_1^{\mathcal{C}}$.

La même observation peut être faite pour les modèles intermédiaire et maximum.

3.1.1 Nombre de points de cassure nul pour le modèle exemplaire

Le problème de déterminer s'il existe un couplage exemplaire tel que le nombre de points de cassure est nul (MCEN) est formellement défini ci-dessous.

MCEN (Mesure d'un Couplage Exemplaire Nulle)

- **Entrée** : Deux génomes G_1 et G_2.
- **Question** : Existe-il un couplage $\mathcal{C}$ de G_1 et G_2 tel que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$ ou $G_1^{\mathcal{C}} = -(G_2^{\mathcal{C}})^r$?

Dans le but de définir plus précisément le contexte d'inapproximabilité du problème de calcul du nombre de points de cassure pour le modèle exemplaire entre deux génomes, nous complétons le résultat de [28], qui montre que MCEN est **NP**-complet même pour des instances de type $(3, 3)$.

Théorème 3.2 MCEN *est* **NP**-*complet même si tout gène apparaît au plus deux fois dans* G_1.

Preuve. L'appartenance de MCEN à **NP** est immédiate. Nous présentons une réduction depuis le problème Couverture de Sommets [44] (voir Illustration 3.1 page 76).

Couverture de Sommets

- **Entrée** : Un graphe $\mathcal{G} = (S, A)$ et un entier positif k.
- **Question** : Existe-t-il un sous-ensemble de k sommets $S' \subseteq S$ tel que chaque arrête dans A soit adjacente à au moins un sommet de S' ?

Soit une instance arbitraire de Couverture de Sommets donnée par un graphe $\mathcal{G} = (S, A)$ et un entier positif k. Nous écrivons $S = \{s_1; s_2; \ldots; s_n\}$ et $A = \{a_1; a_2; \ldots; a_m\}$. Dans la suite de la preuve, les élèments de S (respectivement A) seront vus soit comme des sommets (respectivement arêtes), soit comme des gènes, en fonction du contexte. L'instance correspondante (G_1, G_2) de MCEN est définie comme ceci :

$$G_1 = s_1\ X_1\ s_2\ X_2 \ldots s_n\ X_n$$
$$G_2 = Y[1]\ Y[2]\ \ldots\ Y[k]\ Y_S.$$

Pour chaque $1 \leq i \leq n$, X_i est défini par $X_i = a_{i_1}\ a_{i_2}\ \ldots\ a_{i_j}$, où $a_{i_1}, a_{i_2}, \ldots, a_{i_j}$, $i_1 < i_2 < \ldots < i_j$, sont les arêtes incidentes au sommet s_i. Les mots $Y[i]$, $1 \leq i \leq k$, sont tous égaux et sont définis par $Y[i] = Y_S\ Y_A$ où $Y_S = s_1\ s_2\ \ldots\ s_n$ et $Y_A = a_1\ a_2\ \ldots\ a_m$ (voir Figure 3.1).

Notons que tout gène apparaît au plus deux fois dans G_1 (en effet, les gènes s_i apparaissent une fois et les gènes a_i deux fois). Cependant, le nombre d'occurrences de chaque gène dans G_2 est uniquement borné supérieurement par $k + 1$. En outre, tous les gènes

ont un signe positif, et donc d'après l'Observation 3.1 nous avons besoin uniquement de considérer les couplages $\mathcal{C}$ de G_1 et G_2 tel que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$.

Il est immédiat de vérifier que notre construction peut être exécutée en temps polynomial.

Nous affirmons maintenant qu'il existe une couverture de sommets de taille k dans $\mathcal{G}$ si et seulement s'il existe un couplage pour le modèle exemplaire entre G_1 et G_2 dont le nombre de points de cassure est nul.

Supposons qu'il existe une couverture de sommets $S' \subseteq S$ de taille k dans $\mathcal{G}$. Posons $S' = \{s_{i_1}; s_{i_2}; \ldots; s_{i_k}\}$, $i_1 < i_2 < \ldots < i_k$. Par convenance, nous définissons également i_0 par 0. À partir de S' nous construisons un couplage exemplaire $\mathcal{C}$ comme ci-dessous.

Nous obtenons $G_1^{\mathcal{C}}$ de G_1 *via* une procédure en deux étapes. Dans un premier temps, nous supprimons dans G_1 tous les mots X_i tels que $s_i \notin S'$. Dans un deuxième temps, pour chaque $1 \leq j \leq m$, si le gène a_j est encore présent deux fois, nous supprimons la deuxième occurrence (cette seconde étape concerne les arêtes connectant deux sommets dans S').

Nous passons maintenant à $G_2^{\mathcal{C}}$. Pour $1 \leq j \leq k$, nous considérons le mot $Y[j] = Y_S\ Y_A$ et nous procédons comme suit : (1) nous supprimons dans Y_S tous les gènes sauf s_{i_j} et les gènes $s_\ell \notin S'$ tels que $i_{j-1} < \ell < i_j$, et (2) nous supprimons dans Y_A tous les gènes sauf les gènes a_ℓ qui ne sont pas incidents à s_{i_j} ou qui sont incidents à s_{i_j} et à un sommet plus petit dans S' (c'est-à-dire $a_\ell = \{s_{i_{j'}}, s_{i_j}\}$ pour tout $j' < j$). Finalement, nous supprimons dans le mot restant $Y_S = s_1\ s_2\ \ldots\ s_n$ tous les gènes sauf les s_ℓ ($\notin S'$) tels que $i_k < \ell$.

Puisque S' est une couverture de sommets de $\mathcal{G}$, il s'ensuit que chaque gène est présent une fois dans les génomes obtenus, et donc $\mathcal{C}$ est un couplage exemplaire de G_1 et G_2. Il est maintenant facile de voir que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$, et donc qu'il existe un couplage entre G_1 et G_2, pour le modèle exemplaire, dont le nombre de points de cassure est nul.

Maintenant, supposons que le nombre minimal de points de cassure pour le modèle exemplaire entre G_1 et G_2 soit nulle. Puisque tous les gènes ont un signe positif, il existe un couplage exemplaire $\mathcal{C}$ de G_1 et G_2 tel que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$. Le génome G_1 après $\mathcal{C}$-élaguage peut être écrit

$$G_1^{\mathcal{C}} = Y_S[1]\ Y_A[1]\ Y_S[2]\ Y_A[2]\ \ldots Y_S[k]\ Y_A[k]\ Y_S[k+1]$$

où, $Y_S[i]$, $1 \leq i \leq k+1$, est un mot de S et $Y_A[i]$, $1 \leq i \leq k$, est un mot de A, S et A sont vus comme des alphabets. Maintenant, définissons $S' \subseteq S$ comme ceci : $s_i \in S'$ si et seulement si le gène s_i est présent dans un des $Y_S[j]$, $1 \leq j \leq k$, en dernière position. Par construction, $|S'| \leq k$ (en effet nous pouvons avoir $|S'| < k$ si certains $Y_S[j]$, $1 \leq j \leq k$, sont des mots vides). Nous observons maintenant que, puisqu'aucun gène s_i n'est dupliqué dans G_1, tous les gènes a_ℓ qui sont présents entre un gène $s_i \in S'$ et un gène $s_j \in S$ dans

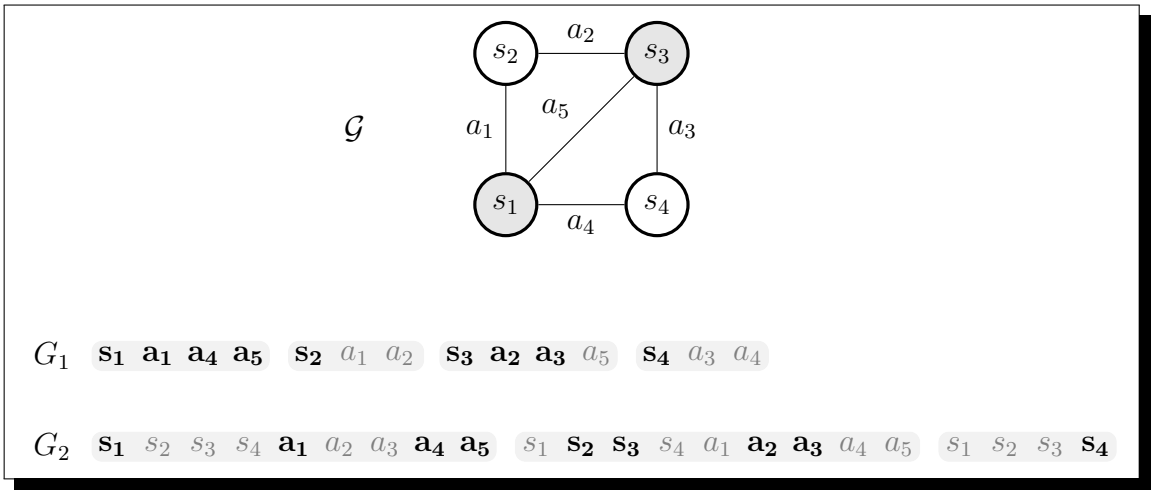


FIGURE 3.1 – **Illustration du Théorème 3.2.** L'ensemble $\{v_1; v_3\}$ est une couverture de sommets minimale pour le graphe $\mathcal{G}$, de taille $k = 2$. Il existe un couplage exemplaire $\mathcal{C}$ tel que le nombre de points de cassure est nul entre les génomes G_1 et G_2, de type $(2, n)$. Les gènes saturés par $\mathcal{C}$ sont mis en gras.

$G_2^{\mathcal{C}}$ devraient être couplés avec des gènes du mot X_i dans G_1. Il s'en suit alors que S' est une couverture de sommets de taille au plus k dans $\mathcal{G}$. □

Rappelons qu'aujourd'hui nous savons que le problème MCEN dans le cas où les instances sont de type $(2, 2)$ est un problème **NP**-complet [19]. Cependant, nous proposons ici un algorithme efficace en pratique - mais exponentiel dans le pire cas - pour MCEN pour des instances de type $(2, 2)$, qui est bien adapté dans le cas où le nombre de gènes qui sont présents deux fois dans G_1 et dans G_2 est relativement petit.

Proposition 3.3 MCEN *pour les instances de type* $(2, 2)$ *(aucun gène n'est présent plus de deux fois dans* G_1 *et dans* G_2*) est résoluble en* $O^*(1.6182^{2k})$ *temps, où* k *est inférieur ou égal au nombre de gènes qui sont présents exactement deux fois dans* G_1 *et dans* G_2*.*

Preuve. D'après l'Observation 3.1, pour chaque instance (G_1, G_2), il est suffisant de se focaliser sur les couplages exemplaires $\mathcal{C}$ tel que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$ ou $G_1^{\mathcal{C}} = -(G_2^{\mathcal{C}})^r$, où $-(G_2^{\mathcal{C}})^r$ est l'inversion signée de $G_2^{\mathcal{C}}$. Dans un premier temps, nous considérons le cas où $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$ (le cas $G_1^{\mathcal{C}} = -(G_2^{\mathcal{C}})^r$ est identique à l'inversion signée près et sera examiné brièvement à la fin de la preuve).

Soit (G_1, G_2) une instance de type $(2, 2)$ de MCEN.

Notre algorithme consiste à transformer une instance (G_1, G_2) en une formule booléenne CNF (Forme Normale Conjonctive) Φ avec peu de grandes clauses, tel que Φ est satis-

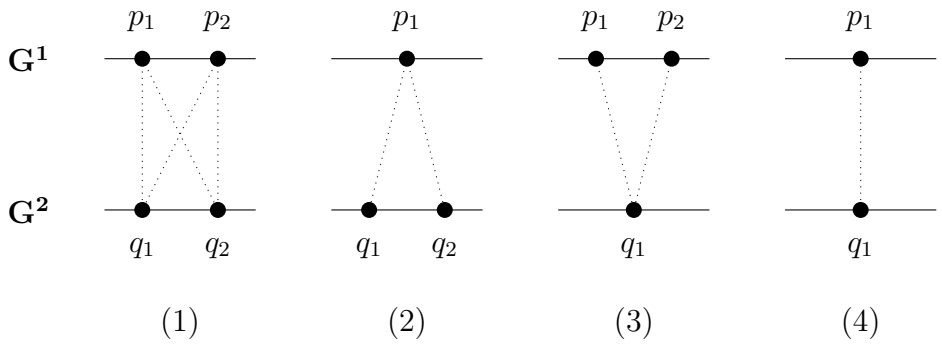


FIGURE 3.2 – Les 4 configurations de famille de gènes pour les instances de type $(2, 2)$: les positions des occurrences des gènes sont p_1 et p_2 dans G_1 et q_1 et q_2 dans G_2.

faisable si et seulement s'il existe un couplage, pour le modèle exemplaire entre G_1 et G_2, tel que le nombre de points de cassure est nul. Par hypothèse, chaque gène signé est présent au plus deux fois dans G_1 et dans G_2. Par conséquent, pour chaque famille de gène `f`, nous avons une des quatre configurations distinctes possibles représentées dans la Figure 3.2, où p_1, p_2, q_1 et q_2 sont les positions des occurrences des gènes $+\mathtt{f}$ et $-\mathtt{f}$ dans G_1 et G_2. En outre, puisque nous regardons pour un couplage exemplaire $\mathcal{C}$ de G_1 et G_2 tel que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$, nous pouvons poser, dans le cas où $occ_1(\mathtt{f}) = 1$ ou $occ_2(\mathtt{f}) = 1$, que toutes les occurrences de `f` ont le même signe (autrement une réduction triviale pourrait être appliquée). En d'autres termes, en se référant à la Figure 3.2, nous posons que $G_1[p_1] = G_2[q_1] = G_2[q_2]$ dans le cas (2), $G_1[p_1] = G_1[p_2] = G_2[q_1]$ dans le cas (3), et $G_1[p_1] = G_2[q_1]$ dans le case (4). Finalement, pour le cas (1), nous pouvons poser que soit toutes les occurrences ont le même signe, soit $G_1[p_1] = -G_1[p_2]$ et $G_2[q_1] = -G_2[q_2]$ (autrement un réduction triviale pourrait être appliquée).

Nous allons maintenant décrire la construction de la formule booléenne CNF Φ. Premièrement, l'ensemble des variables booléennes X est défini comme ceci : pour chaque famille `f` dont une occurrence est à la position p dans G_1 et une autre occurrence à la position q dans G_2 (c'est-à-dire $|G_1[p]| = |G_2[q])|$) nous ajoutons à X la variable booléenne x_q^p. Nous allons maintenant définir les clauses de Φ. Soit `g` un gène quelconque, et soient les positions des occurrences de la famille $|g|$ dans G_1 et dans G_2 celles de la Figure 3.2.

- si $\mathsf{occ}_1(|\mathsf{g}|) = \mathsf{occ}_2(|\mathsf{g}|) = 2$ (cas (1)),
  - si $G_1[p_1] = G_1[p_2] = G_2[q_1] = G_2[q_2]$, nous ajoutons à Φ les clauses $(x_{q_1}^{p_1} \vee x_{q_2}^{p_1} \vee x_{q_1}^{p_2} \vee x_{q_2}^{p_2})$, $(\overline{x_{q_1}^{p_1}} \vee \overline{x_{q_2}^{p_1}})$, $(\overline{x_{q_1}^{p_1}} \vee \overline{x_{q_1}^{p_2}})$, $(\overline{x_{q_1}^{p_1}} \vee \overline{x_{q_2}^{p_2}})$, $(\overline{x_{q_2}^{p_1}} \vee \overline{x_{q_1}^{p_2}})$, $(\overline{x_{q_2}^{p_1}} \vee \overline{x_{q_2}^{p_2}})$ et $(\overline{x_{q_1}^{p_2}} \vee \overline{x_{q_2}^{p_2}})$,
  - sinon, nous avons $G_1[p_1] = -G_1[p_2]$ et $G_2[q_1] = -G_2[q_2]$ (voir la discussion au-dessus),
    - si $G_1[p_1] = G_2[q_1]$ et $G_1[p_2] = G_2[q_2]$)), nous ajoutons à Φ les clauses $(x_{q_1}^{p_1} \vee x_{q_2}^{p_2})$

et $(\overline{x_{q_1}^{p_1}} \vee \overline{x_{q_2}^{p_2}})$,
  - si $G_1[p_1] = G_2[q_2]$ et $G_1[p_2] = G_2[q_1]$)), nous ajoutons à Φ les clauses $(x_{q_2}^{p_1} \vee x_{q_1}^{p_2})$ et $(\overline{x_{q_2}^{p_1}} \vee \overline{x_{q_1}^{p_2}})$,
- si $\mathsf{occ}_1(|\mathsf{g}|) = 1$ et $\mathsf{occ}_2(|\mathsf{g}|) = 2$ (cas (2)), nous ajoutons à Φ les clauses $(x_{q_1}^{p_1} \vee x_{q_2}^{p_1})$ et $(\overline{x_{q_1}^{p_1}} \vee \overline{x_{q_2}^{p_1}})$,
- si $\mathsf{occ}_1(|\mathsf{g}|) = 2$ et $\mathsf{occ}_2(|\mathsf{g}|) = 1$ (cas (3)), nous ajoutons à Φ les clauses $(x_{q_1}^{p_1} \vee x_{q_1}^{p_2})$ et $(\overline{x_{q_1}^{p_1}} \vee \overline{x_{q_1}^{p_2}})$, et
- si $\mathsf{occ}_1(|\mathsf{g}|) = \mathsf{occ}_2(|\mathsf{g}|) = 1$ (cas (4)), nous ajoutons à Φ la clause $(x_{q_1}^{p_1})$.

Grâce à cette construction, si la formule Φ est évaluée à vraie pour une formule propositionnelle f et que $f(x_q^p)$ est vraie pour une famille $|\mathsf{g}|$ présente à la position p dans G_1 et q dans G_2, alors toutes les occurrences de $|\mathsf{g}|$, sauf celle à la position p, devront être supprimées dans G_1 et toutes les occurrences de $|\mathsf{g}|$, sauf celle à la position q, devront être supprimées dans G_2, pour ainsi obtenir une solution pour le modèle exemplaire.

Il reste à faire en sorte que Φ soit évaluée à vraie si et seulement s'il existe un couplage entre G_1 et G_2, pour le modèle exemplaire, dont le nombre de points de cassure est nul. Dans ce but, nous ajoutons à Φ les clauses suivantes. Pour chaque paire de variables $(x_{j_1}^{i_1}, x_{j_2}^{i_2})$ tel que $|G_1[i_1]| \neq |G_1[i_2]|$, $i_1 < i_2$ et $j_1 > j_2$, nous ajoutons à Φ la clause $(\overline{x_{j_1}^{i_1}} \vee \overline{x_{j_2}^{i_2}})$. La construction de Φ est maintenant complète.

Clairement, Φ est évaluée à vrai si et seulement s'il existe un couplage entre G_1 et G_2, pour le modèle exemplaire, dont le nombre de points de cassure est nul. Soit k le nombre de familles qui ont deux occurrences dans G_1 et dans G_2 avec le même signe, c'est-à-dire $G_1[p_1] = G_1[p_2] = G_2[q_1] = G_2[q_2]$. Nous faisons maintenant l'observation importante que toutes les clauses dans Φ ont une taille inférieure ou égale à 2 exceptées les k clauses de taille 4 introduites dans le cas où la famille $|\mathsf{g}|$ a deux occurrences dans G_1 et dans G_2 avec le même signe. En introduisant une nouvelle variable booléenne, nous pouvons facilement remplacer dans Φ chaque clause de taille 4 par deux clauses de taille 3, et donc nous pouvons maintenant supposer que Φ est une formule 3-CNF (c'est-à-dire chaque clause a une taille d'au plus 3) avec exactement $2k$ clauses de taille 3.

De même pour le cas $-(G_1^{\mathcal{C}})^r = G_2^{\mathcal{C}}$, nous remplaçons G_1 par $-(G_1)^r$ et nous construisons une autre formule 3-CNF Φ' comme ci-dessus. Les deux formules 3-CNF ont besoin, cependant, d'être examinées séparément.

Fernau propose dans [41] un algorithme pour résoudre les formules booléennes 3-CNF qui tourne en $O^*(1.6182^\ell)$ temps, où ℓ est le nombre de clauses de taille 3. Par conséquent, MCEN pour les instances de type $(2, 2)$ est résoluble en $O^*(1.6182^{2k})$ temps, où k est le nombre de familles qui ont deux occurrences dans G_1 et dans G_2. $\square$

3.1.2 Nombre de points de cassure nul pour le modèle intermédiaire

Nous passons maintenant au problème qui consiste à déterminer s'il existe un couplage dont le nombre de points de cassure est nul pour le modèle intermédiaire (MCIN). Nous le définissons ainsi.

MCIN (Mesure d'un Couplage Intermédiaire Nulle)

- **Entrée** : Deux génomes G_1 et G_2.
- **Question** : Existe-il un couplage intermédiaire $\mathcal{C}$ de G_1 et G_2 tel que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$ ou $G_1^{\mathcal{C}} = -(G_2^{\mathcal{C}})$?

Nous montrons ici que MCEN et MCIN sont des problèmes équivalents. Nous avons besoin du lemme suivant.

Lemme 3.4 *Soient G_1 et G_2 deux génomes sans duplication et avec le même contenu en gène, et G'_1 et G'_2 deux génomes obtenus à partir de G_1 et G_2 par suppression d'un même gène g. Alors le nombre de points de cassure entre G'_1 et G'_2 est inférieur ou égal à celui entre G_1 et G_2.*

Théorème 3.5 MCEN *et* MCIN *sont des problèmes équivalents.*

Preuve. L'une des deux directions est triviale (chaque couplage exemplaire est en effet un couplage intermédiaire). L'autre direction se déduit du Lemme 3.4. □

Il résulte du Théorème 3.5 que le problème de minimiser le nombre de points de cassure, pour le modèle intermédiaire, n'est pas approximable même pour les instances de type $(3, 3)$ (voir [28]) et si aucun gène est présent plus de deux fois dans G_1 (voir le Théorème 3.2).

3.1.3 Nombre de points de cassure nul pour le modèle maximum

Nous montrons ici que, contrairement aux modèles exemplaire et intermédiaire, décider s'il existe un couplage entre deux génomes, pour le modèle maximum, tel que le nombre de points de cassure est nul, est résoluble en temps polynomial (nous nommons ce problème MCMN). Nous ne pouvons donc pas exclure l'existence d'algorithmes d'approximation pour le calcul du nombre minimum de points de cassure pour le modèle maximum.

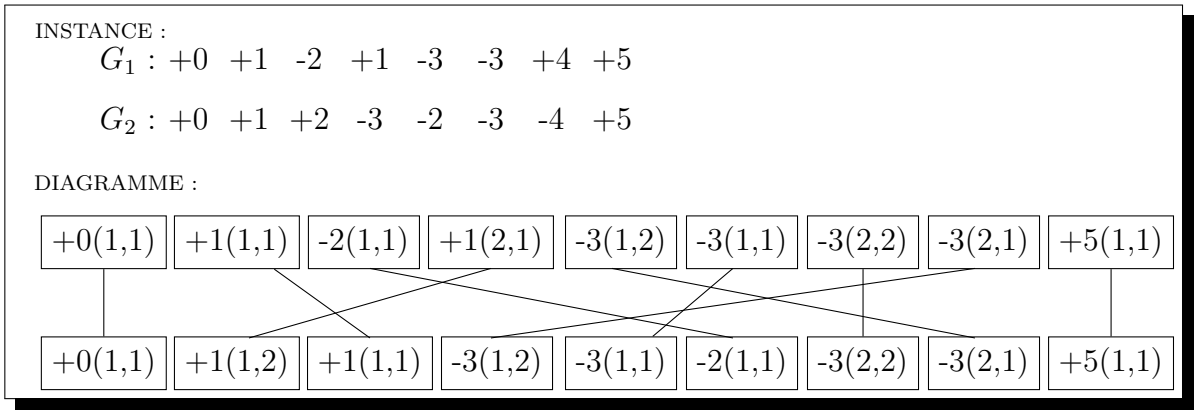


FIGURE 3.3 – **Traduction d'une instance du problème MCMN en un diagramme de couplage.**

MCMN (MESURE D'UN COUPLAGE MAXIMUM NULLE)

- **Entrée** : Deux génomes G_1 et G_2.
- **Question** : Existe-il un couplage maximum $\mathcal{C}$ de G_1 et G_2 tel que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$ ou $G_1^{\mathcal{C}} = -(G_2^{\mathcal{C}})$?

L'idée principale de notre approche est de transformer chaque instance de MCMN en un *diagramme de couplage* et ensuite d'utiliser un algorithme efficace pour trouver le plus grand ensemble de segments non sécants. Notons que ce dernier problème équivaut à trouver la plus grande suite croissante dans des permutations.

Un diagramme de couplage est constitué de deux lignes parallèles de chacune n points et de n segments qui rejoignent un point de chaque ligne de manière distincte. Le graphe d'intersection des segments est appelé *graphe de permutation* (la raison de ce nom est que si les points de la ligne du dessus sont numérotés par $1, 2, \ldots, n$, alors les points de l'autre ligne sont numérotés par une permutation de $1, 2, \ldots, n$) [47].

Nous décrivons maintenant la traduction de la paire de génomes (G_1, G_2) en un diagramme de couplage $D(G_1, G_2)$ (voir Figure 3.3 pour une illustration). Par souci de présentation, nous introduisons les notations suivantes. Pour chaque famille de gène $\mathtt{f}$, nous écrivons $\mathtt{occ}_{\text{pos}}(G, \mathtt{f})$ (respectivement $\mathtt{occ}_{\text{neg}}(G, \mathtt{f})$) pour le nombre d'occurrences du gène $+\mathtt{f}$ (respectivement $-\mathtt{f}$) dans le génome G. D'après l'Observation 3.1, il suffit de considérer deux cas : $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$ ou $G_1^{\mathcal{C}} = -(G_2^{\mathcal{C}})^r$, où $\mathcal{C}$ est un couplage maximum de G_1 et G_2.

Dans un premier temps, nous nous focalisons sur le cas de $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$ (le cas $G_1^{\mathcal{C}} = -(G_2^{\mathcal{C}})^r$ est identique à une inversion signé près). Nous décrivons la construction des points

en haut du diagramme. En lisant le génome G_1 de gauche à droite, nous remplaçons le gène g par la séquence de points d'étiquettes

$$+\mathsf{g}_1(i, \mathsf{occ}_{\text{pos}}(G_2, |\mathsf{g}|)) \quad + \mathsf{g}_1(i, \mathsf{occ}_{\text{pos}}(G_2, |\mathsf{g}|) - 1) \quad \ldots \quad + \mathsf{g}_1(i, 1)$$

si g est la $i^{\text{ème}}$ occurrence positive de la famille $|\mathsf{g}|$ dans le génome G_1 ou par la séquence de points d'étiquettes

$$-\mathsf{g}_1(i, \mathsf{occ}_{\text{neg}}(G_2, |\mathsf{g}|)) \quad - \mathsf{g}_1(i, \mathsf{occ}_{\text{neg}}(G_2, |\mathsf{g}|) - 1) \quad \ldots \quad - \mathsf{g}_1(i, 1)$$

si g est la $i^{\text{ème}}$ occurrence négative de la famille $|\mathsf{g}|$ dans le génome G_1. Une construction symétrique est effectuée pour les points de la ligne du dessous, c'est-à-dire en lisant le génome G_2 de gauche à droite, nous remplaçons un gène g par la séquence de points d'étiquettes

$$+\mathsf{g}_2(i, \mathsf{occ}_{\text{pos}}(G_1, |\mathsf{g}|)) \quad + \mathsf{g}_2(i, \mathsf{occ}_{\text{pos}}(G_1, |\mathsf{g}|) - 1) \quad \ldots \quad + \mathsf{g}_2(i, 1)$$

si g est la $i^{\text{ème}}$ occurrence positive de la famille $|\mathsf{g}|$ dans le génome G_2 ou par la séquence de points d'étiquettes

$$-\mathsf{g}_2(i, \mathsf{occ}_{\text{neg}}(G_1, |\mathsf{g}|)) \quad - \mathsf{g}_2(i, \mathsf{occ}_{\text{neg}}(G_1, |\mathsf{g}|) - 1) \quad \ldots \quad - \mathsf{g}_2(i, 1)$$

si g est la $i^{\text{ème}}$ occurrence négative de la famille $|\mathsf{g}|$ dans le génome G_2. Nous obtenons maintenant le diagramme de couplage $D(G_1, G_2)$ comme suit : chaque point d'étiquette $+\mathsf{g}_1(i, j)$ (respectivement $-\mathsf{g}_1(i, j)$) de la ligne du dessus est connecté au point d'étiquette $+\mathsf{g}_2(j, i)$ (respectivement $-\mathsf{g}_2(j, i)$) de la ligne du dessous par un segment. Clairement, chaque point est incident à exactement un segment, et donc $D(G_1, G_2)$ est bien un diagramme de couplage.

Il est important d'observer que par construction, pour chaque $x \in \{1; 2\}$ et chaque couple de points d'étiquettes $+\mathsf{g}_x(i, j)$ et $+\mathsf{g}_x(i, k)$, $j \neq k$, les deux segments incidents à ces deux points sont sécants ; la même conclusion peut être tirée pour deux points quelconques d'étiquette $-\mathsf{g}_x(i, j)$ et $-\mathsf{g}_x(i, k)$, $j \neq k$. Le lemme suivant précise cette propriété de manière appropriée.

Lemme 3.6 *Si les deux segments* $[+\mathsf{g}_1(i, j), +\mathsf{g}_2(j, i)]$ *et* $[+\mathsf{g}_1(k, \ell), +\mathsf{g}_2(\ell, k)]$ *(respectivement* $[-\mathsf{g}_1(i, j), -\mathsf{g}_2(j, i)]$ *et* $[-\mathsf{g}_1(k, \ell), -\mathsf{g}_2(\ell, k)]$*) sont non sécants dans le diagramme de couplage* $D(G_1, G_2)$*, alors* $i \neq k$ *et* $j \neq \ell$*.*

Théorème 3.7 MCMN *peut être résolu en un temps polynomial.*

Preuve. Soient G_1 et G_2 deux génomes, et m la taille du couplage maximum entre G_1 et G_2. D'après le Lemme 3.6, il existe un couplage maximum $\mathcal{C}$ de G_1 et G_2 tel que $G_1^{\mathcal{C}} = G_2^{\mathcal{C}}$ s'il existe m segments non sécants dans $D(G_1, G_2)$. Le nombre maximum de segments non sécants dans un diagramme de couplage avec n points sur chaque ligne peut être trouvé en temps $O(n \log \log n)$ [23].

De la même manière, pour le cas $G_1^{\mathcal{C}} = -(G_2^{\mathcal{C}})^r$, nous remplaçons G_1 par $-(G_1)^r$ et utilisons le même algorithme sur le diagramme de couplage obtenu. □

3.2 Mesure de points de cassure et d'adjacences

Après avoir affiné les classes de complexité des problèmes de calcul de mesure entre deux génomes en présence de duplications, nous nous penchons sur la recherche d'algorithmes permettant de calculer des mesures : le nombre de points de cassure et le nombre d'adjacences. Les principaux résultats étudiés dans cette section sont le fruit d'une collaboration avec Sébastien Angibaud, Guillaume Fertin, Irena Rusu et Stéphane Vialette [2, 3].

Soient deux génomes G_1 et G_2 et un modèle (exemplaire, intermédiaire ou maximum). Nous désirons trouver un couplage qui (i) est approprié au modèle considéré, et (ii) retourne le nombre optimal de points de cassure/d'adjacences, où l'*optimal* signifie le *minimum* pour les points de cassure et le *maximum* pour les adjacences. Après avoir vu que les problèmes qui en résultent ne sont pas tous différents (en effet, seulement trois d'entre eux le sont), nous proposerons des algorithmes exacts et des heuristiques pour leurs résolutions.

3.2.1 Passer de six problèmes à trois

Passer de six problèmes à quatre

Soit G_1 et G_2 deux génomes et $\mathcal{C}$ un couplage entre G_1 et G_2 pour un modèle quelconque (exemplaire, intermédiaire ou maximum). Nous définissons $\texttt{pdc}(\mathcal{C})$ le nombre de points de cassure entre les deux génomes $\mathcal{C}$-élagués. Le nombre d'adjacences entre deux génomes $\mathcal{C}$-élagués est noté par $\texttt{adj}(\mathcal{C})$.

Il est facile de voir, mais important de noter, que pour un couplage $\mathcal{C}$ entre deux génomes G_1 et G_2, nous avons

$$\texttt{adj}(\mathcal{C}) + \texttt{pdc}(\mathcal{C}) = |\mathcal{C}| + 1. \tag{3.1}$$

Par conséquent, pour n'importe quelle instance donnée, si la taille du couplage $\mathcal{C}$ est fixée, alors trouver le couplage qui minimise le nombre de points de cassure est équivalent à trouver le couplage qui maximise le nombre d'adjacences. Nous avons donc la proposition qui suit.

Proposition 3.8 *Minimiser le nombre de points de cassure pour le modèle exemplaire est équivalent à maximiser le nombre d'adjacences ; le même résultat tient aussi pour le modèle maximum.*

Pour le modèle intermédiaire, le résultat ne se tient plus, car la cardinalité du couplage ne peut pas être inférée des instances.

Passer de quatre problèmes à trois

De plus, et moins simplement, une équivalence existe entre deux des problèmes restant.

Proposition 3.9 *Minimiser le nombre de points de cassure pour le modèle exemplaire est équivalent à minimiser le nombre de points de cassure pour le modèle intermédiaire.*

Preuve. Soient G_1 et G_2 deux génomes. Notons $\mathtt{pdc}_E^{\mathbf{opt}}$ et $\mathtt{pdc}_I^{\mathbf{opt}}$ le nombre minimum de points de cassure obtenu entre G_1 et G_2 pour les modèles exemplaire et intermédiaire, respectivement.

De toute évidence, nous avons $\mathtt{pdc}_E^{\mathbf{opt}} \geq \mathtt{pdc}_I^{\mathbf{opt}}$ puisque une solution pour le modèle exemplaire est aussi une solution pour le problème intermédiaire. Il reste donc à prouver que $\mathtt{pdc}_E^{\mathbf{opt}} \leq \mathtt{pdc}_I^{\mathbf{opt}}$.

À cette fin, considérons une solution optimale du problème pour le modèle intermédiaire, c'est-à-dire un couplage $\mathcal{C}$ entre des gènes de G_1 et G_2 apportant $\mathtt{pdc}_I^{\mathbf{opt}}$ points de cassure. Nous allons alors construire une solution (G'_1, G'_2) pour le modèle exemplaire qui a au plus $\mathtt{pdc}_I^{\mathbf{opt}}$ points de cassure. Ceci est obtenu en utilisant l'algorithme glouton suivant : tant qu'il existe deux gènes saturés dans G_1 qui appartiennent à la même famille (indifféremment du signe), supprimer un des deux gènes arbitrairement, et le gène correspondant dans G_2. L'algorithme ci-dessus garantit une solution pour le modèle exemplaire. Nous soutenons que cette solution a au plus autant de points de cassure que la solution pour le modèle intermédiaire, qui est $\mathtt{pdc}_I^{\mathbf{opt}}$. Afin de prouver cela, nous considérons chaque itération de l'algorithme ci-dessus et montrons que le couplage obtenu a au plus autant de points de cassure que l'instance initiale. Quatre cas distincts doivent-être examinés. Pour tous gènes a, la notation $a_\blacktriangle$ désigne la présence d'un point de cassure après le gène a.

1. $G_1 = \ldots a\ X\ b \ldots$, $G_2 = \ldots a\ X\ b \ldots$ ou $G_2 = \ldots -b\ -X\ -a \ldots$
   La suppression du gène X dans les deux génomes induit les génomes $G'_1 = \ldots a\ b \ldots$, $G'_2 = \ldots a\ b \ldots$ ou $G'_2 = \ldots -b\ -a \ldots$ où aucun point de cassure est introduit.
2. $G_1 = \ldots a_\blacktriangle X\ b \ldots$, $G_2 = \ldots c\ X\ b \ldots$ ou $G_2 = \ldots -b\ -X\ c \ldots$
   La suppression du gène X dans les deux génomes induit les génomes $G'_1 = \ldots a_\blacktriangle b \ldots$, $G'_2 = \ldots c\ b \ldots$ ou $G'_2 = \ldots -b\ c \ldots$ où aucun point de cassure est introduit.
3. $G_1 = \ldots a\ X_\blacktriangle b \ldots$, $G_2 = \ldots c\ X\ b \ldots$ ou $G_2 = \ldots b\ -X\ -c \ldots$
   La suppression du gène X dans les deux génomes les génomes $G'_1 = \ldots a_\blacktriangle b \ldots$, $G'_2 = \ldots c\ b \ldots$ ou $G'_2 = \ldots b\ -c \ldots$ où aucun point de cassure est introduit.
4. $G_1 = \ldots a_\blacktriangle X_\blacktriangle b \ldots$
   Pour un G_2 arbitraire, la suppression du gène X dans les deux génomes induit soit le génome $G'_1 = \ldots a\ b \ldots$, soit le génome $G'_1 = \ldots a_\blacktriangle b \ldots$. Dans les deux cas, le nombre de points de cassure décroit strictement, ce qui contredit l'optimalité de la solution pour le modèle intermédiaire. Par conséquent, ce cas ne peut pas se produire.

Pour chaque cas, la suppression du gène X dans les deux génomes induit deux génomes où aucun point de cassure n'est introduit. Par conséquent, la solution générée ne peut pas admettre plus de points de cassure que $\mathtt{pdc}_I^{\mathbf{opt}}$. Nous avons donc $\mathtt{pdc}_I^{\mathbf{opt}} = \mathtt{pdc}_E^{\mathbf{opt}}$. □

Les trois problèmes principaux

Nous sommes maintenant en position de définir formellement les problèmes d'optimisation qui nous intéressent ici. Comme précédement, G_1 et G_2 sont deux génomes.

Soit $\mathbf{opt}_{\mathrm{E}}$ (respectivement $\mathbf{opt}_{\mathrm{M}}$) le problème qui consiste à trouver un couplage exemplaire (respectivement maximum) $\mathcal{C}$ tel que les génomes $G_1^{\mathcal{C}}$ et $G_2^{\mathcal{C}}$ ont un nombre maximum d'adjacences. La proposition 3.8 garantit que le même couplage $\mathcal{C}$ minimise aussi le nombre de points de cassure. Nous désignons par $\mathtt{adj}_E^{\mathbf{opt}}$ et $\mathtt{pdc}_E^{\mathbf{opt}}$ (respectivement $\mathtt{adj}_M^{\mathbf{opt}}$ et $\mathtt{pdc}_M^{\mathbf{opt}}$) le nombre d'adjacences et le nombre de points de cassure de la solution optimale de $\mathbf{opt}_{\mathrm{E}}$ (respectivement $\mathbf{opt}_{\mathrm{M}}$).

Soit $\mathbf{opt}_{\mathrm{IA}}$ (respectivement $\mathbf{opt}_{\mathrm{IP}}$) le problème qui consiste à trouver un couplage intermédiaire $\mathcal{C}$ tel que les génomes $G_1^{\mathcal{C}}$ et $G_2^{\mathcal{C}}$ ont un nombre d'adjacences maximum (respectivement un nombre de points de cassure minimum). Nous rappelons que d'après la Proposition 3.9, tant que notre but est de réduire le nombre de points de cassure, nous n'avons pas besoin de résoudre en particulier $\mathbf{opt}_{\mathrm{IP}}$, en supposant que nous résolvons $\mathbf{opt}_{\mathrm{E}}$. Nous notons $\mathtt{adj}_I^{\mathbf{opt}}$ le nombre d'adjacences de la solution optimale de $\mathbf{opt}_{\mathrm{IA}}$ et nous notons $\mathtt{pdc}_I^{\mathbf{opt}}$ le nombre de points de cassure de la solution optimale de $\mathbf{opt}_{\mathrm{IP}}$.

3.2.2 Une approche exacte par programmation linéaire à variables booléennes

Minimiser le nombre de points de cassure entre deux génomes avec des gènes dupliqués est un problème **NP**-difficile pour le modèle exemplaire, même si pour toutes les familles $\mathtt{f} \in \mathcal{F}_1 \cup \mathcal{F}_2$ nous avons $\mathtt{occ}_1(\mathtt{f}) = 1$ et $\mathtt{occ}_2(\mathtt{f}) \leq 2$ [21]. Par conséquent, la **NP**-difficulté tient aussi pour les deux modèles intermédiaire et maximum, ce qui signifie que les problèmes $\mathbf{opt}_{\mathrm{E}}$ (et donc $\mathbf{opt}_{\mathrm{IP}}$, voir Proposition 3.9) et $\mathbf{opt}_{\mathrm{M}}$ sont **NP**-difficiles. Du reste, un résultat de [27] implique que $\mathbf{opt}_{\mathrm{IA}}$ est aussi **NP**-difficile.

Dans le but d'améliorer les évaluations d'heuristiques, nous développons dans cette section une approche générique exacte similaire à celle initiée dans [6]. Plus précisément, notre approche repose sur l'expression des différents problèmes en programme linéaire à variables booléennes [79] et l'utilisation de solveur puissant pour obtenir des solutions optimales.

Pour faciliter la présentation, nous commençons par présenter entièrement le programme linéaire à variables booléennes qui permet de maximiser le nombre d'adjacences pour le modèle maximum (problème $\mathbf{opt}_{\mathrm{M}}$). Puis nous donnons certaines règles de réduction de données pour réduire la taille des entrées, et donc accélérer la résolution

du programme. Finalement, nous montrons comment adapter ce programme pour $\mathbf{opt}_{\mathrm{E}}$ et $\mathbf{opt}_{\mathrm{IA}}$.

Maximiser le nombre d'adjacences pour le modèle maximum

Le programme linéaire à variables booléennes que nous proposons ici (mentionné par la suite comme le programme `Couplage-Maximum-Adjacences`) prend en entrée deux génomes avec des gènes dupliqués, et résout le problème $\mathbf{opt}_{\mathrm{M}}$. Nous notons par $\mathcal{F}$ l'ensemble des familles de gènes appartenant aux deux génomes comparés. Le programme est presenté Figure 3.4. La méthode pour adapter le programme aux autres problèmes qui nous intéresse est reportée à la Sous-section 3.2.2.

Le programme `Couplage-Maximum-Adjacences` considère deux génomes G_1 et G_2 de longueur respective n_1 et n_2. La fonction objective, les variables et les contraintes impliquées sont maintenant détaillées.

Les variables :

- ⋆ Les variables $a(i,k)$, $1 \leq i \leq n_1$ et $1 \leq k \leq n_2$, définissent un couplage $\mathcal{C}$: $a_{i,k} = 1$ si et seulement si $G_1[i]$ est couplé avec $G_2[k]$ dans $\mathcal{C}$.
- ⋆ Les variables $b_x(i)$, $x \in \{1;2\}$ et $1 \leq i \leq n_x$, représentent les gènes $\mathcal{C}$-*saturés* : $b_x(i) = 1$ si et seulement si $G_x[i]$ est saturé par le couplage $\mathcal{C}$. Clairement, $\sum_{1\leq i\leq n_1} b_1(i) = \sum_{1\leq k\leq n_2} b_2(k)$, et ceci est précisément la taille de $\mathcal{C}$.
- ⋆ Les variables $c_x(i,j)$, $x \in \{1;2\}$ et $1 \leq i < j \leq n_x$, représentent les *gènes consécutifs* relativement à $\mathcal{C}$: $c_x(i,j) = 1$ si et seulement si $G_x[i]$ et $G_x[j]$ sont tous les deux saturés par $\mathcal{C}$ et aucun gène $G_x[p]$, $i < p < j$, n'est saturé par $\mathcal{C}$.
- ⋆ Les variables $d(i,j,k,\ell)$, $1 \leq i < j \leq n_1$ et $1 \leq k < \ell \leq n_2$, représentent les *adjacences* relativement à $\mathcal{C}$: $d(i,j,k,\ell) = 1$ si et seulement si
  - soit $(G_1[i], G_2[k])$ et $(G_1[j], G_2[\ell])$ appartiennent à $\mathcal{C}$, $G_1[i] = G_2[k]$ et $G_1[j] = G_2[\ell]$, soit $(G_1[i], G_2[\ell])$ et $(G_1[j], G_2[k])$ appartiennent à $\mathcal{C}$, $G_1[i] = -G_2[\ell]$ et $G_1[j] = -G_2[k]$,
  - $G_1[i]$ et $G_1[j]$ sont consécutifs dans G_1 relativement à $\mathcal{C}$, et
  - $G_2[k]$ et $G_2[\ell]$ sont consécutifs dans G_2 relativement à $\mathcal{C}$.

Les contraintes :

Posons $x \in \{1;2\}$, $1 \leq i < j \leq n_1$ et $1 \leq k < \ell \leq n_2$.

- ⋆ La contrainte `C.01` garantit que chaque gène de G_1 et de G_2 est couplé au plus une fois, c'est-à-dire que $b_1(i) = 1$ (respectivement $b_2(k) = 1$) si et seulement si le gène i (respectivement k) est couplé dans G_1 (respectivement G_2) ; voir Figure 3.5 pour une illustration de cette contrainte. Observons que dans chaque couplage, chaque paire de gènes couplés ensembles appartiennent nécessairement à la même famille de gène, et donc nous n'avons pas besoin de préciser explicitement que $a(i,k) = 0$

Le programme `Couplage-Maximum-Adjacences`

Objectif :

`Maximiser` $\sum_{0\leq i<n_1} \sum_{i<j\leq n_1} \sum_{0\leq k<n_2} \sum_{k<\ell\leq n_2} d(i,j,k,\ell)$

Contraintes :

`C.01` $\forall\ 1 \leq i \leq n_1, \quad \sum_{\substack{1\leq k\leq n_2 \\ |G_1[i]|=|G_2[k]|}} a(i,k) = b_1(i)$

$\forall\ 1 \leq k \leq n_2, \quad \sum_{\substack{1\leq i\leq n_1 \\ |G_1[i]|=|G_2[k]|}} a(i,k) = b_2(k)$

`C.02` $\forall\ x \in \{1;2\}, \forall\ \mathtt{f} \in \mathcal{F}, \quad \sum_{\substack{1\leq i\leq n_x \\ |G_x[i]|=|\mathtt{f}|}} b_x(i) = \min(\mathtt{occ}_1(\mathtt{f}), \mathtt{occ}_2(\mathtt{f}))$

`C.03` $\forall\ x \in \{1;2\}, \forall\ 1 \leq i \leq j-1 < n_x,\ c_x(i,j) + \sum_{i<p<j} b_x(p) \geq 1$

`C.04` $\forall\ x \in \{1;2\}, \forall\ 1 \leq i < p < j \leq n_x,\ c_x(i,j) + b_x(p) \leq 1$

`C.05` $\forall\ 1 \leq i < j \leq n_1, \forall\ 1 \leq k < \ell \leq n_2,$ tel que $G_1[i] = G_2[k]$ et $G_1[j] = G_2[\ell]$,
$a(i,k) + a(j,\ell) + c_1(i,j) + c_2(k,\ell) - d(i,j,k,\ell) \leq 3$

`C.06` $\forall\ 1 \leq i < j \leq n_1, \forall\ 1 \leq k < \ell \leq n_2,$ tel que $G_1[i] = G_2[k]$ et $G_1[j] = G_2[\ell]$,
$a(i,k) - d(i,j,k,\ell) \geq 0$
$a(j,\ell) - d(i,j,k,\ell) \geq 0$
$c_1(i,j) - d(i,j,k,\ell) \geq 0$
$c_2(k,\ell) - d(i,j,k,\ell) \geq 0$

`C.07` $\forall\ 1 \leq i < j \leq n_1, \forall\ 1 \leq k < \ell \leq n_2,$ tel que $G_1[i] = -G_2[\ell]$ et $G_1[j] = -G_2[k]$,
$a(i,\ell) + a(j,k) + c_1(i,j) + c_2(k,\ell) - d(i,j,k,\ell) \leq 3$

`C.08` $\forall\ 1 \leq i < j \leq n_1,\ \forall\ 1 \leq k < \ell \leq n_2,$ tel que $G_1[i] = -G_2[\ell]$ et $G_1[j] = -G_2[k]$,
$a(i,\ell) - d(i,j,k,\ell) \geq 0$
$a(j,k) - d(i,j,k,\ell) \geq 0$
$c_1(i,j) - d(i,j,k,\ell) \geq 0$
$c_2(k,\ell) - d(i,j,k,\ell) \geq 0$

`C.09` $\forall\ 1 \leq i < j \leq n_1,\ \forall\ 1 \leq k < \ell \leq n_2,$
tel que $\{|G_1[i]|, |G_1[j]|\} \neq \{|G_2[k]|, |G_2[\ell]|\}$ ou $G_1[i] - G_1[j] \neq G_2[k] - G_2[\ell]$,
$d(i,j,k,\ell) = 0$

`C.10` $\forall\ 1 \leq i < j \leq n_1,$
$\sum_{1\leq k<n_2} \sum_{k<\ell\leq n_2} d(i,j,k,\ell) \leq 1$

Domaines :

$\forall\ x \in \{1;2\}, \forall\ 1 \leq i < j \leq n_1, \forall\ 1 \leq k < \ell \leq n_2, \quad a(i,k),\ b_x(i),\ c_x(i,k),\ d(i,j,k,\ell) \in \{0;1\}$

FIGURE 3.4 – Le programme `Couplage-Maximum-Adjacences` permet de trouver le maximum d'adjacences entre deux génomes pour le modèle maximum.

dans le cas où $G_1[i]$ et $G_2[k]$ sont deux gènes appartenant à des familles de gènes différentes.

- ⋆ La contrainte `C.02` définit le modèle considéré (ici le modèle maximum). Pour chaque famille de gène `f`, $\min(\mathtt{occ}_1(\mathtt{f}), \mathtt{occ}_2(\mathtt{f}))$ occurrences de gènes appartenant à `f` doivent être $\mathcal{C}$-saturés dans G_1 et G_2 (voir une nouvelle fois la Figure 3.5 pour une illustration).
- ⋆ Les contraintes `C.03` et `C.04` permettent de définir la consécutivité des gènes. La variable $c_x(i,j)$ est égale à 1 si et seulement s'il n'existe aucun p tel que $i < p < j$ et $b_x(p) = 1$. Il convient de noter ici qu'il est possible d'avoir $c_x(i,j) = 1$ même si un des gènes $G_x[i]$ ou $G_x[j]$ *n'est pas* $\mathcal{C}$-saturé.
- ⋆ Les contraintes de `C.05` à `C.10` définissent les variables d. Dans le cas où $G_1[i] = G_2[k]$ et $G_1[j] = G_2[\ell]$, les contraintes `C.05` et `C.06` garantissent que nous avons $d(i,j,k,\ell) = 1$ si et seulement si toutes les variables $a(i,k)$, $a(j,\ell)$, $c_1(i,j)$ et $c_2(k,\ell)$ sont égales à 1. Dans le cas où $G_1[i] = -G_2[\ell]$ et $G_1[j] = -G_2[k]$, les contraintes `C.07` et `C.08` garantissent que nous avons $d(i,j,k,\ell) = 1$ si et seulement si toutes les variables $a(i,\ell)$, $a(j,k)$, $c_1(i,j)$ et $c_2(k,\ell)$ sont égales à 1. La contrainte `C.09` fixe les variables $d(i,j,k,\ell)$ à 0 si aucun de ces deux précédents cas n'est retenu. Finalement, grâce à la contrainte `C.10`, il faut avoir au plus une adjacence pour chaque paire (i,j). Voir Figure 3.6 pour une illustration.

L'objectif du programme `Couplage-Maximum-Adjacences` est de maximiser le nombre d'adjacences entre les deux génomes considérés. Cet objectif se réduit donc pour notre modèle à maximiser la somme de toutes les variables $d(i,j,k,\ell)$.

Accélération du programme

Le programme `Couplage-Maximum-Adjacences` a $O((n_1 \cdot n_2)^2)$ variables et $O((n_1 \cdot n_2)^2)$ contraintes. Dans le but d'accélérer l'exécution du programme, nous présentons ici des règles simples pour réduire le nombre de variables et de contraintes impliquées.

Prétraitement des génomes. Les génomes sont des paires prétraitées dans lesquels toutes les familles de gènes n'apparaissant pas au moins une fois dans chaque génome sont supprimées. Cette règle simple donne des résultats significatifs en pratique. Par exemple, pour la base de données de γ-Protéobactéries étudiée dans la Section 3.3, la taille moyenne des génomes est réduite de 3000 à 1300 gènes.

Réduction du nombre de variables et de contraintes. Pour les gènes non-dupliqués, c'est-à-dire les gènes `g` pour lequel $\mathtt{occ}_1(|\mathtt{g}|) = \mathtt{occ}_2(|\mathtt{g}|) = 1$, la variable correspondante $a_{i,k}$ est mise directement à 1, ainsi que les deux variables $b_1(i)$ et $b_2(k)$.

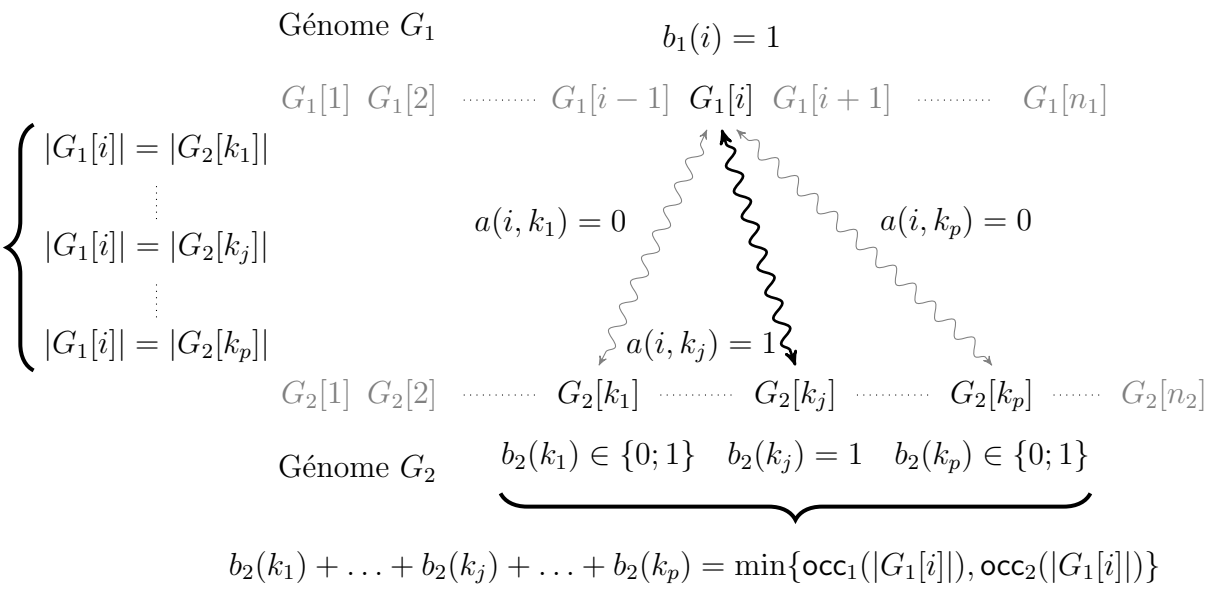


FIGURE 3.5 – **Illustration des contraintes portant sur les variables** $b_1(i)$ $1 \leq i \leq n_1$. Si le gène $G_1[i]$ apparaît aux positions $k_1 < k_2 < \ldots < k_p$ dans G_2 et le gène $G_1[i]$ est couplé au gène $G_2[k_j]$ dans ce couplage solution, alors **(i)** $a(i, k_j) = 1$, *i.e.*, le gène $G_1[i]$ est couplé au gène $G_2[k_j]$, **(ii)** $a(i, k_q) = 0$ pour $1 \leq q \leq p$ et $q \neq j$, *i.e.*, le gène $G_1[i]$ est couplé seulement à un gène dans G_2, **(iii)** $b_1(i) = 1$, *i.e.*, le gène $G_1[i]$ est couplé à un gène de G_2 et **(iv)** $b_2(k_j) = 1$, *i.e.*, le gène $G_2[k_j]$ est couplé à un gène de G_1. Observons qu'il est possible d'ajouter que $b_2(k_q) = 1$ pour $1 \leq q \leq p$ et $q \neq j$ si $\min(\mathsf{occ}_1(|G_1[i]|), \mathsf{occ}_2(|G_1[i]|) \geq 1$ (cette observation n'est cependant plus valide sous le modèle exemplaire).

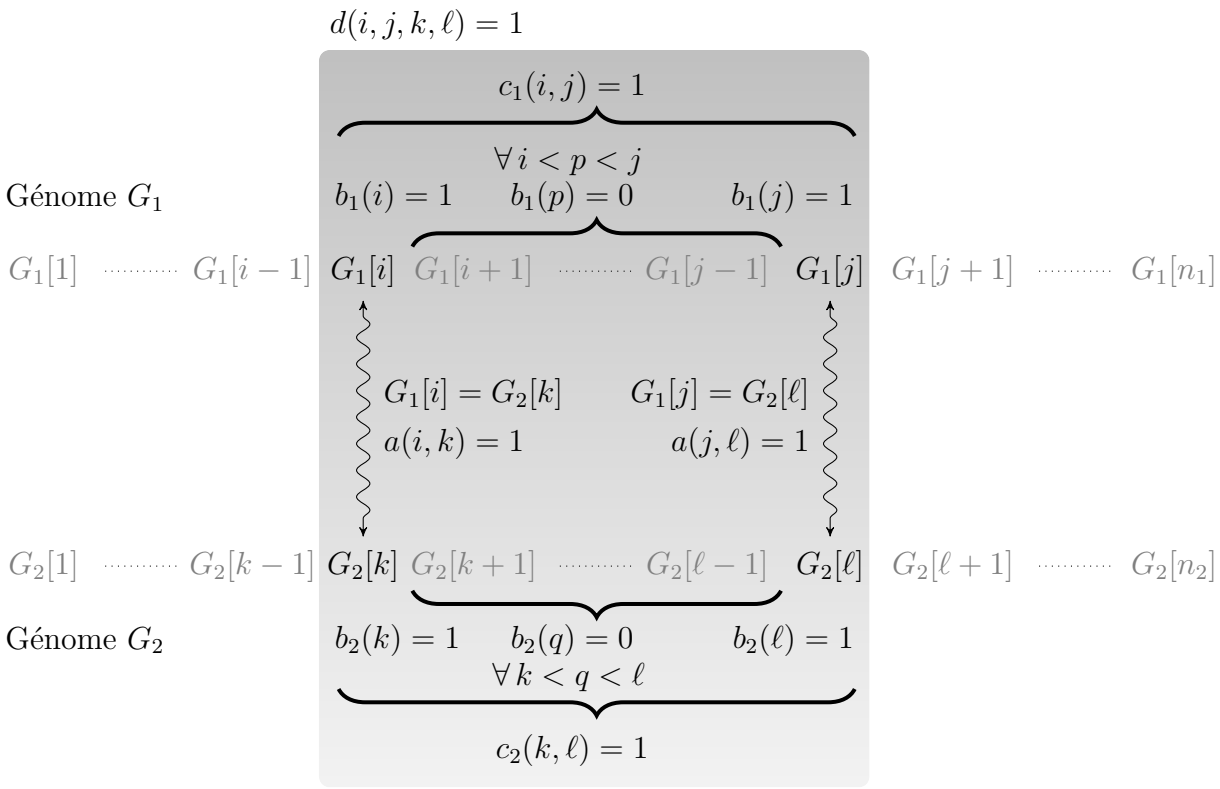


FIGURE 3.6 – **Illustration des contraintes portant sur les variables** $d(i,j,k,\ell)$, $1 \leq i < j \leq n_1$ et $1 \leq k < \ell \leq n_2$, pour $G_1[i] = G_2[k]$ et $G_1[j] = G_2[\ell]$. Les deux gènes $G_1[i]$ et $G_1[j]$ sont adjacents selon le couplage solution si il existe deux gènes $G_2[k]$ et $G_2[\ell]$, $G_1[i] = G_2[k]$ et $G_1[j] = G_2[\ell]$, tel que **(i)** $G_1[i]$ est couplé à $G_2[k]$, *i.e.*, $a(i,k) = 1$, **(ii)** $G_1[j]$ est couplé à $G_2[\ell]$, *i.e.*, $a(j,\ell) = 1$, **(iii)** aucun gène entre $G_1[i]$ et $G_1[j]$ est couplé à un gène de G_2, *i.e.*, $c_1(i,j) = 1$ et **(iv)** aucun gène entre $G_2[k]$ et $G_2[\ell]$ est couplé à un gène de G_2, *i.e.*, $c_2(k,\ell) = 1$. Cette situation se réduit sous notre modèle à $d(i,j,k,\ell) = 1$.

De même, si deux gènes non-dupliqués sont présents consécutivement ou dans un ordre inversé avec des signes opposés sur G_1 et G_2, la variable correspondante d est mise directement à 1 et les contraintes liées sont supprimées.

Si pour deux positions i et j, il existe un gène qui doit être obligatoirement saturé par n'importe quel couplage (par exemple si une famille a toutes ses occurrences entre i et j dans G_1), alors les gènes $G_1[i]$ et $G_1[j]$ ne peuvent pas être consécutifs. Par conséquent, la variable $c_1(i,j)$ et la variable $d(i,j,k,\ell)$ pour tout $1 \leq k < \ell \leq n_2$ sont mises directement à 0 et les contraintes liées sont supprimées. Nous pouvons utiliser le même raisonnement pour deux positions k et ℓ dans G_2 et les variables $c_2(k,\ell)$ et $d(i,j,k,\ell)$ pour tout $1 \leq i < j \leq n_1$.

Pré-couplage Il existe plusieurs couplages permettant d'optimiser une mesure pour un modèle donné. Sachant cela et dans le but de réduire le nombre de variables et de contraintes du programme linéaire à variables booléennes, nous mettons en place un pré-couplage.

Le principe est d'imposer certains couplages de gènes avant l'écriture du programme linéaire à variables booléennes, sans risquer de perdre l'optimalité. Pour cela, nous mettons en place la technique suivante. Soient deux gènes non dupliqués sur G_1 et sur G_2, situés aux positions x_1 et y_1 sur G_1 et x_2 et y_2 sur G_2. Soient E_1 et E_2 l'ensemble des gènes compris entre x_1 et y_1 dans G_1 et entre x_2 et y_2 dans G_2, respectivement. Si $E_1 = E_2$ et s'il existe un couplage entre les gènes de E_1 et de E_2 n'induisant aucun point de cassure alors :

- nous couplons les gènes de E_1 avec ceux de E_2 en respectant ce couplage optimal ;
- les variables $a(i,k)$, $b_1(i)$, $b_2(k)$, $c_1(i,j)$, $c_2(k,l)$ et $d(i,j,k,l)$ ($x_1 \leq i,j \leq y_1$, $x_2 \leq k,l \leq y_2$) relativement à ce couplage sont mises directement à 1 ; les contraintes associées sont supprimées.

Sous le modèle exemplaire, les autres occurrences des gènes appartenant à E_1 (donc non couplées) ne doivent pas être par la suite couplées, par définition du modèle. Les variables correspondantes sont mises à 0 et les contraintes associées sont supprimées. Notons qu'à partir de ce pré-couplage, nous pouvons toujours obtenir un couplage optimisant le nombre d'adjacences ou le nombre de points de cassure, pour les trois modèles.

Adaptation pour les autres mesures et les autres modèles

Le programme `Couplage-Maximum-Adjacences` nous permet de résoudre le problème $\mathbf{opt}_{\mathrm{M}}$. Nous décrivons ici comment modifier ce programme linéaire à variables booléennes pour l'adapter aux deux problèmes restants, à savoir $\mathbf{opt}_{\mathrm{E}}$ et $\mathbf{opt}_{\mathrm{IA}}$. Comme nous allons le voir, seules quelques modifications sont nécessaires.

Modèle exemplaire (problème $\mathbf{opt}_{\mathbf{E}}$). Comme nous l'avons vu précédemment, la contrainte `C.02` explicite le modèle considéré. Pour le modèle exemplaire, exactement une

occurrence de chaque famille de gène doit être saturée. Par conséquent, la contrainte `C.02` devra être écrite comme ci-dessous.

$$\texttt{C.02}\ \forall\ x \in \{1;2\}, \forall\ \texttt{f} \in \mathcal{F}_x, \sum_{\substack{1 \leq i \leq n_x \\ |G_x[i]|=\texttt{f}}} b_x(i) = 1$$

De plus, l'ajout d'un règle simple peut accélérer l'exécution du programme. En effet, nous devons avoir exactement une occurrence de chaque gène dans chaque génome. Par conséquent, pour tout $0 \leq i < j \leq n_x$, $x \in \{1;2\}$, si $|G_x[i]| = |G_x[j]|$ alors $c_x(i,j) = 0$. Les variables d correspondantes sont mises directement à 0 et les contraintes liées sont supprimées. De plus, un prétraitement sur les génomes peut-être appliqué spécifiquement pour le modèle exemplaire. En effet, s'il existe i, $0 \leq i \leq n_x$, tel que $G_x[i] = G_x[i+1]$ alors nous pouvons supprimer soit $G_x[i]$ soit $G_x[i+1]$.

Modèle intermédiaire (problème $\mathbf{opt_{IA}}$). Encore une fois, nous sommes concernés par la contrainte `C.02`. Pour le modèle intermédiaire, au moins un gène de chaque famille de gène doit être saturé. Cette simple réduction se traduit dans la réécriture de la contrainte `C.02` comme ci-dessous.

$$\texttt{C.02}\ \forall\ x \in \{1;2\}, \forall\ \texttt{f} \in \mathcal{F}_x, \sum_{\substack{1 \leq i \leq n_x \\ |G_x[i]|=\texttt{f}}} b_x(i) \geq 1$$

Comme nous l'avons vu dans la section précédente, calculer le nombre minimum de points de cassure ou le nombre maximum d'adjacences ne sont pas des problèmes équivalents pour le modèle intermédiaire. Pour calculer le nombre minimum de points de cassure (problème $\mathbf{opt}_{\mathrm{IP}}$), l'objectif doit être modifié comme ci-dessous (la correction suivante résulte de la Relation (3.1)).

$$\texttt{Minimiser} \sum_{1 \leq i < n_1} b_1(i) - \sum_{1 \leq i < n_1} \sum_{i < j \leq n_1} \sum_{1 \leq k < n_2} \sum_{k < \ell \leq n_2} d(i,j,k,\ell) - 1$$

Clairement, ce dernier programme linéaire à variables booléennes est inutil puisque $\mathbf{opt}_{\mathrm{IP}}$ et $\mathbf{opt}_{\mathrm{E}}$ ont été montré comme des problèmes équivalents (voir Proposition 3.9). Cependant, comme nous allons le voir dans la Section 3.3, le modèle intermédiaire nous donne l'opportunité, pour le même nombre minimum de points de cassure que pour le modèle exemplaire, d'obtenir plus d'adjacences. Comme simple illustration de ce point,

considérons G_1 = +0 +1 -4 +2 -1 +2 -3 +4 et G_2 = +0 +3 +1 +2 -1 -3 +4 deux génomes. Nous avons $\mathtt{pdc}_E^{\mathbf{opt}} = 0$ comme le montre les deux génomes élagués $G'_1 = G'_2 = +0\ +1\ +2\ -3\ +4$; cette solution apporte 4 adjacences. Cependant, la solution pour le modèle intermédiaire $G''_1 = G''_2$ = +0 +1 +2 -1 -3 +4 induit toujours 0 point de cassure, mais apporte 5 adjacences.

3.2.3 Des algorithmes heuristiques

Nous verrons dans la Section 3.3 que notre approche par programmation linéaire à variables booléennes, présentée dans la section précédente, nous permet d'obtenir presque tous les résultats exacts attendus pour les modèles exemplaire, intermédiaire et maximum sur l'ensemble des γ-Protéobactéries étudié, mais que les limites de cette méthode sont atteintes.

En effet, et spécialement pour le modèle intermédiaire, certains résultats ne peuvent être obtenus en utilisant cette méthode, et une des raisons à cela est que les tailles des génomes sont trop grandes pour la méthode par programmation linéaire à variables booléennes pour être en mesure d'être gérées. En d'autres termes, alors que cette méthode semble être prometteuse pour de "petits" génomes (c'est-à-dire des génomes qui, après prétraitement, ne dépassent pas, environ, deux milles gènes), il y a un besoin crucial d'algorithmes rapides (et alors non nécessairement exacts) pour les cas de génomes ayant des tailles plus importantes. Bien sûr, ces algorithmes heuristiques doivent fournir des résultats de haute qualité, c'est-à-dire le plus proche possible des résultats exacts. Dans ce sens et grâce à une comparaison avec les résultats exacts présentés dans la Section 3.3, nous sommes en mesure d'évaluer plusieurs heuristiques et de valider l'efficacité des heuristiques proposées dans cette section.

Dans la suite, neuf heuristiques seront présentées et étudiées (trois pour chaque modèle). Chacune de ces heuristiques entre dans l'une des deux catégories suivantes : (i) les heuristiques basées sur la recherche itérative de *la plus Longue Sous-séquence Commune* ("Longest Common Subsequence" en anglais et *LCS*, par abréviation) de deux génomes, et (ii) les heuristiques qui sont un *hybride* entre la catégorie (i) susmentionnée et la méthode par programmation linéaire à variables booléennes. Notons que deux heuristiques hybrides seront proposées pour chaque modèle.

Avant de décrire en détail nos heuristiques, nous rappelons que pour le modèle intermédiaire, deux problèmes différents existent : nous désirons trouver soit un couplage qui maximise le nombre d'adjacences, soit un couplage qui minimise le nombre de points de cassure. Cependant, nous avons vu que minimiser le nombre de points de cassure pour le modèle intermédiaire est équivalent à minimiser le nombre de points de cassure pour le modèle exemplaire (Proposition 3.9). Donc toute heuristique qui vise à réduire au minimum le nombre de points de cassure pour le modèle exemplaire va de même minimiser le

nombre de points de cassure pour le modèle intermédiaire. Par conséquent, quand nous nous ramenons au modèle intermédiaire, nous allons nous focaliser uniquement sur les heuristiques qui cherchent à maximiser le nombre d'adjacences.

Les heuristiques `IILCS`

Nous commençons par décrire ici l'idée principale qui est derrière les heuristiques basées sur la recherche itérative de la plus Longue Sous-séquence Commune (ou LCS), que nous avons appelé heuristiques `IILCS`. Soient G_1 et G_2 deux génomes : un *LCS* de (G_1, G_2) est une sous séquence S commune à (G_1, G_2) et de longueur maximale, à l'inversion signée près (signes opposés et ordre inversé). L'idée ici est de coupler, à chaque itération, *tous les gènes* qui sont présents dans un *LCS*, jusqu'à ce que le couplage désiré soit obtenu.

Nous notons que cette idée n'est pas nouvelle : elle a déjà été utilisée, par exemple, dans [65]. Dans [6], cette heuristique a été améliorée de la manière suivante : à chaque itération, non seulement nous couplons tous les gènes qui sont contenus dans un *LCS* (**Règle 1**), mais nous enlevons aussi chaque gène non-couplé du génome pour lequel il n'y a aucun gène non-couplé de la même famille dans l'autre génome (**Règle 2**).

Les trois heuristiques que nous présentons utilisent les deux règles mentionnées ci-dessus. Puisque la différence entre les problèmes se situe dans le couplage que nous désirons obtenir, nos trois heuristiques vont différer sur ce point spécifique.

Modèle maximum. Nous appliquons itérativement la **Règle 1** et la **Règle 2** jusqu'à ce que l'algorithme s'arrête. Par définition de la **Règle 1** et de la **Règle 2**, ceci implique que le couplage résultant est maximum. Nous appelons cette heuristique `IILCS_M`.

Modèle intermédiaire. Nous appliquons aussi itérativement la **Règle 1** et la **Règle 2**. La différence ici est la condition d'arrêt : l'algorithme s'arrête dès que chaque famille de gène a été couplée au moins une fois. Nous appelons cette heuristique `IILCS_IA`.

Modèle exemplaire. Nous appliquons itérativement la **Règle 1** et la **Règle 2** une nouvelle fois, mais nous appliquons des suppressions de gènes en plus à chaque itération. En effet, nous avons besoin de nous assurer que seul *un gène* de chaque famille est couplé sur un génome. Dans ce cas, à chaque itération, pour les gènes dupliqués qui sont contenu dans le *LCS* courant, nous conservons et couplons arbitrairement la première occurrence ; alors que pour les gènes `g` qui ne sont pas dans le *LCS*, nous appliquons la règle suivante : si `g` est présent dans le *LCS* (et donc couplé), alors nous supprimons toutes les autres occurrences de `g` dans le reste des deux génomes. Lorsque cette heuristique s'arrête, nous avons la garantie d'obtenir un couplage exemplaire. Nous appelons cette heuristique `IILCS_E`.

Remarquons que, pour chacune de ces trois heuristiques, il peut exister, à chaque itération, plusieurs LCS ; dans ce cas, nous avons décidé de choisir de façon aléatoire l'un d'entre eux. Ainsi, si nous exécutons `IILCS_M` (respectivement `IILCS_IA`, `IILCS_E`) plusieurs fois sur la même instance, nous pouvons obtenir différents couplages, et donc le nombre de points de cassure et d'adjacences peut varier. C'est pourquoi, pour chaque paire de génomes, nous avons décidé de lancer dix fois les heuristiques `IILCS_M`, `IILCS_IA` et `IILCS_E` et de garder le meilleur résultat.

Les heuristiques hybrides

Nous allons maintenant décrire la deuxième catégorie d'heuristiques que nous proposons pour résoudre nos problèmes pour les trois modèles. Ces heuristiques sont basées sur une méthode hybride utilisant l'heuristique `IILCS` appropriée, puis l'algorithme de programmation linéaire à variables booléennes. Le principe est de calculer une partie du couplage par itération de l'heuristique `IILCS` appropriée jusqu'à ce que la taille du LCS soit strictement plus petite qu'un paramètre donné k. Une fois ce critère obtenu, nous complétons le couplage par l'algorithme de programmation linéaire à variables booléennes approprié. Cette heuristique est appelée `HYB_M(k)` (respectivement `HYB_IA(k)`, `HYB_E(k)`) pour le modèle maximum (respectivement intermédiaire, exemplaire).

Pour chacun de ces trois modèles, nous avons testé l'heuristique hybride décrite ci-dessus pour deux valeurs différentes de k, à savoir $k = 2$ et $k = 3$. Nous avons délibérément choisi de petites valeurs pour k, car lorsque k est plus grand, utiliser l'algorithme exact pour compléter le couplage partiel peut augmenter de façon non négligeable le temps de calcul. Nous perdons l'avantage qu'a l'heuristique au niveau de la vitesse. De plus, nous allons voir dans la section suivante que les résultats obtenus avec $k = 2$ et $k = 3$ sont déjà extrêmement bons.

Les neuf heuristiques que nous proposons sont basées sur la recherche de LCS. Ceci se justifie par le fait que pour chaque LCS de taille t trouvée et couplée nous obtenons au moins $t - 1$ adjacences et au plus 2 points de cassure. Nous pouvons donc espérer avoir des résultats heuristiques proches des résultats exacts. Pourtant il est à noter qu'il existe des cas particuliers où les heuristiques sont particulièrement mauvaises. La Figure 3.7 donne un exemple simple de deux génomes G_1 et G_2 pour lesquels les heuristiques `IILCS` et hybride donnent de mauvais résultats ($3m + 3$ points de cassure et $3m + 2$ adjacences contre 4 points de cassure et $6m$ adjacences de façon optimale) pour chacun des trois modèles.

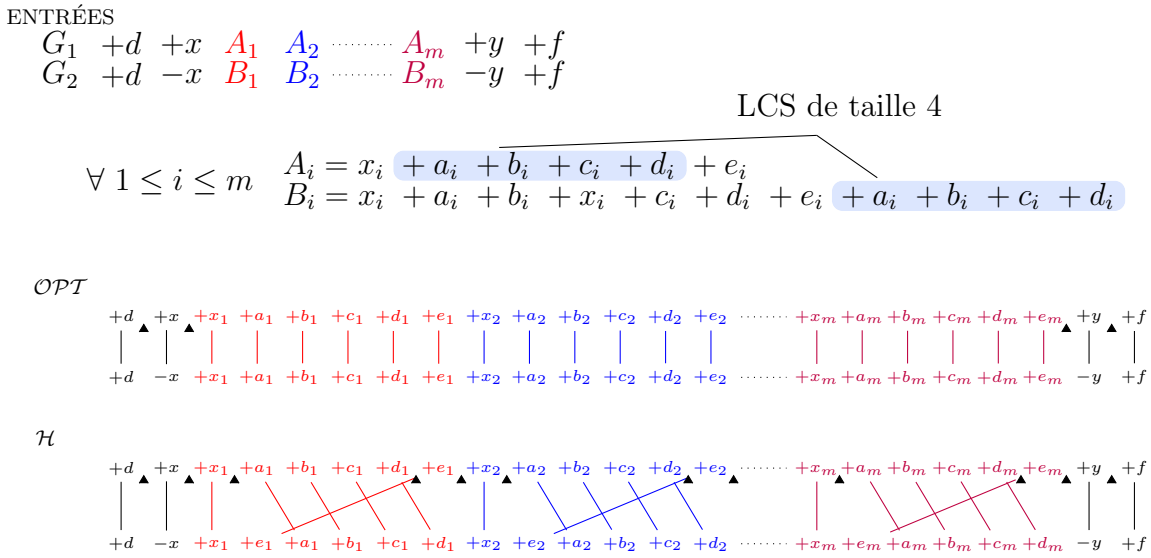


FIGURE 3.7 – **Exemple d'une paire de génomes** (G_1, G_2) **pour laquelle l'heuristique** $\mathcal{H}$ **(IILCS, HYB-2 ou HYB-3) est mauvaise** : nous obtenons $3m + 3$ points de cassures et $3m + 2$ adjacences via $\mathcal{H}$ contre 4 et $6m$ avec un programme exacte. Ce résultat est vérifié quel que soit le modèle choisi (exemplaire, intermédiaire ou maximum). Pour tous les gènes g, la notation $g_{\blacktriangle}$ désigne la présence d'un point de cassure après le gène g.

3.3 Résultats expérimentaux

En nous basant sur notre approche générique exacte par programmation linéaire à variables booléennes et sur nos heuristiques (voir Section 3.2.2 et Section 3.2.3), nous présentons dans cette section une campagne de calculs pour obtenir au mieux tous les résultats pour les problèmes $\mathbf{opt}_{\mathrm{M}}$, $\mathbf{opt}_{\mathrm{IA}}$ et $\mathbf{opt}_{\mathrm{E}}$ (notons que le problème $\mathbf{opt}_{\mathrm{IP}}$ sera aussi discuté pour les raisons développées à la fin de la Section 3.2.2).

La raison de cette campagne est triple. Premièrement, nous avons pour objectif de tester la pertinence de notre approche générique par programmation linéaire à variables booléennes sur des données biologiques réelles. En particulier, nous sommes intéressés par l'identification d'instance non-artificielle difficile à résoudre, et plus généralement dans l'estimation des limites intrinsèques de l'approche proposée. Deuxièmement, nous cherchons à comparer pour un ensemble de données biologiquement pertinentes les trois problèmes $\mathbf{opt}_{\mathrm{M}}$, $\mathbf{opt}_{\mathrm{IA}}$ et $\mathbf{opt}_{\mathrm{E}}$, et comparer les trois modèles. Finalement, dans le but d'évaluer des heuristiques (voir Section 3.2.3), nous apportons ici un ensemble de résultats exacts quasiment complet permettant de se référer, et nous croyons que ces résultats peuvent être intéressants pour la communauté dans l'élaboration de nouvelles approches heuristiques.

Le solveur de programmes linéaires utilisé dans cette campagne est `CPLEX`[1]. Le choix d'utiliser ce solveur se justifie par le fait que ces temps de calcul sont significativement plus faibles par rapport aux temps de calculs des autres solveurs testés. Tous les calculs ont été effectués sur un Quadri Intel(R) Xeon(TM) CPU 3.00 Ghz avec 16GB de mémoire tournant sous Linux. L'ensemble des résultats obtenus (mesures, temps, génomes, etc) sont disponibles à l'adresse suivante : `www.lri.fr/~thevenin`.

3.3.1 Données

Nous conduisons notre campagne de calculs sur un ensemble de génomes γ-Protéobactériens, originalement étudié dans [59]. Notons d'une part que cet ensemble de données est devenu une référence dans la génomique comparative [60, 17, 6]. D'autre part, il est composé de génomes de taille raisonnable, et est donc un bon candidat pour des calculs intensifs. Plus précisément, cet ensemble de données est composé de douze génomes complets de γ-Protéobactéries sur les treize originalement étudiés dans [59]. En effet, le treizième génome (*V.cholerae*) n'a pas été considéré, puisqu'il est composé de deux chromosomes, et il ne correspond donc pas au modèle général que nous considérons ici pour la représentation des génomes. L'ensemble des données est composé des génomes suivants :

1. `http ://www.ilog.com/products/cplex/`

- *Buchnera aphidicola APS* (`Baphi`, Numéro d'accès de Genbank NC_002528),
- *Escherichia coli K12* (`Ecoli`, NC_000913),
- *Haemophilus influenzae Rd* (`Haein`, NC_000907),
- *Pseudomonas aeruginosa PA01* (`Paeru`, NC_002516),
- *Pasteurella multocida Pm70* (`Pmult`, NC_002663),
- *Salmonella typhimurium LT2* (`Salty`, NC_003197),
- *Wigglesworthia glossinidia brevipalpis* (`Wglos`, NC_004344),
- *Xanthomonas axonopodis pv. citri 306* (`Xaxon`, NC_003919),
- *Xanthomonas campestris* (`Xcamp`, NC_0 03902),
- *Xylella fastidiosa 9a5c* (`Xfast`, NC_002488),
- *Yersinia pestis CO_92* (`Ypest-CO92`, NC_003143) et
- *Yersinia pestis KIM5 P12* (`Ypest-KIM`, NC_004088).

La détermination des familles de gène, où chaque famille est supposée représenter un groupe de gènes homologues, est tirée de [17] et donc, bien que ce soit une étape préliminaire cruciale, elle ne sera pas discutée plus avant ici. Initialement, les tailles des génomes sont comprises entre 564 et 5540 gènes (voir Tableau 3.1). Qui plus est, 8% des familles de gènes sont, en moyenne, dupliquées (ces duplications couvrent, en moyenne, 20% des gènes d'un génome). La distance séparant deux occurrences d'une famille de gène est de 29% de la taille totale du génome. Le nombre de gènes `g` séparés de la prochaine occurrence de la famille $|g|$ par un gène non dupliqué est en moyenne de $71,5\%$.

Génome	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92	Y-KIM
Taille	564	4185	1709	5540	2015	4203	653	4192	4029	2680	3599	3879

TABLE 3.1 – **Taille des génomes avant le prétraitement**

3.3.2 Résultats exacts

Après l'étape de prétraitement (voir Section 3.2.2), 10% des familles de gènes sont, en moyenne, dupliquées et ces duplications couvrent, en moyenne, 27% des gènes d'un génome. En outre, le prétraitement de l'ensemble de nos données par paires de génomes réduit drastiquement la taille des génomes. Par exemple, lorsque nous nous focalisons sur la comparaison de `Baphi` et `Ecoli`, leur taille est réduite de 564 et 4185 gènes à 531 et 721 gènes, respectivement. Le Tableau 3.2 indique toutes les tailles réduites pour les comparaisons de paires pour le modèle exemplaire et le Tableau 3.3 indique toutes les tailles réduites pour les comparaisons de paires pour le modèle maximum. Notons que cette dernière table donne aussi toutes les tailles réduites des comparaisons de paires pour le modèle intermédiaire (en effet, concernant l'étape de prétraitement, un couplage maximum peut être vu comme le "pire" cas parmi tous les couplages intermédiaires possibles).

De plus, toujours après le prétraitement, la distance séparant deux occurrences d'une même famille de gène est de 32% de la taille totale du génome pour le modèle exemplaire et de 30% pour les modèles intermédiaire et maximum. Le nombre de gènes g séparés de la prochaine occurrence de la famille |g| par un gène non dupliqué, est en moyenne 96% de la taille des génomes pour le modèle exemplaire. Sous les modèles intermédiaire et maximum, cette moyenne est de $71,5\%$.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92	Y-KIM
Baphi		721	498	709	525	723	382	545	540	460	721	718
Ecoli	531		1227	2088	1370	3279	570	1365	1355	908	2427	2452
Haein	434	1490		1307	1403	1514	440	924	900	718	1384	1399
Paeru	470	1889	969		1078	1912	510	1675	1665	984	1776	1785
Pmult	448	1637	1390	1382		1660	465	967	945	748	1517	1537
Salty	533	3253	1233	2066	1373		569	1365	1352	913	2448	2476
Wglos	380	769	497	769	533	779		610	603	496	765	763
Xaxon	416	1435	794	2005	860	1467	465		3445	1488	1328	1343
Xcamp	416	1437	791	2022	854	1470	464	3455		1470	1308	1321
Xfast	400	1096	711	1289	754	1117	437	1645	1623		1041	1050
Y-CO92	530	2536	1189	2014	1327	2566	537	1290	1268	881		3388
Y-KIM	523	2537	1185	2011	1323	2574	558	1286	1262	882	3362	

Table 3.2 – **Tailles réduites des génomes pour la comparaison de paires pour le modèle *exemplaire*.** À titre d'illustration, le prétraitement des données pour la comparaison de `Baphi` avec `Ecoli` réduit la taille des deux génomes à 531 (`Baphi`) et 721 (`Ecoli`) gènes.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92	Y-KIM
BAPHI		745	504	733	533	738	389	564	559	465	745	748
ECOLI	535		1254	2129	1395	3320	582	1411	1398	927	2470	2525
HAEIN	437	1529		1338	1431	1540	451	964	939	734	1422	1455
PAERU	473	1931	987		1098	1942	522	1726	1711	1007	1815	1848
PMULT	451	1675	1426	1414		1688	477	1008	985	773	1556	1595
SALTY	537	3310	1263	2107	1399		581	1413	1395	931	2490	2547
WGLOS	381	791	508	800	542	794		639	631	502	792	799
XAXON	419	1476	809	2044	878	1495	477		3539	1522	1364	1395
XCAMP	419	1473	803	2057	869	1494	476	3541		1499	1339	1366
XFAST	403	1122	723	1316	769	1135	449	1697	1679		1073	1091
Y-CO92	534	2584	1220	2052	1353	2607	579	1333	1311	901		3473
Y-KIM	527	2587	1216	2048	1349	2613	570	1329	1305	902	3422	

Table 3.3 – **Tailles réduites des génomes pour la comparaison de paires de génomes pour les modèles *maximum* et *intermédiaire*.** À titre d'illustration, le prétraitement des données pour comparer `Baphi` avec `Ecoli` réduit la taille des deux génomes par 535 (`Baphi`) et 745 (`Ecoli`) gènes.

La campagne de calculs, dans laquelle tous les programmes linéaires à variables

booléennes ont été étendus avec les règles d'accélération décrites dans la Sous-section 3.2.2, a donné les résultats présentés dans le Tableau 3.4 (le symbole `X` représente les instances non résolues).

Les temps indiqués par la suite sont les temps nécessaires à l'écriture du programme linéaire à variables booléennes et à sa résolution. Notons que le temps d'écriture du programme, pour les trois modèles, est de l'ordre de la seconde.

Résultats pour les trois modèles

Modèle maximum. Rappelons tout d'abord que calculer le nombre minimum de points de cassure et le nombre maximum d'adjacences pour le modèle maximum sont des problèmes équivalents (Proposition 3.8). Ce problème s'est avéré être le plus facile pour notre ensemble de données de γ-Protéobactéries. `CPLEX` a en effet calculé les mesures de *toutes* les paires (Tableau 3.4) en un peu moins de 3 minutes (3 secondes, en moyenne, pour chaque paire de génomes).

Modèle exemplaire. Une fois de plus, rappelons ici que calculer le nombre minimum de points de cassure et le nombre maximum d'adjacences sont des problèmes équivalents pour le modèle exemplaire (Proposition 3.8).

Contrairement au modèle maximum, nous n'avons pas réussi à calculer les mesures de toutes les paires de génomes. Plus précisément, 65 résultats sur 66 (Tableau 3.4) ont été obtenus grâce au programme linéaire à variables booléennes en un peu moins de 3 minutes.

Nos tentatives pour le calcul du dernier cas ont échoué à cause d'une limite de mémoire imposée par `CPLEX`. Malheureusement, nous n'avons toujours aucune explication à cette situation surprenante et contre-intuitive. Cependant, nous pensons que ce fait n'est pas lié à `CPLEX` (considéré ici comme une boîte noire). En effet, une explosion combinatoire similaire (et plus importante) a été observée lors de l'utilisation d'un autre solveur (`MiniSat+` [39]) lors du calcul du nombre d'adjacences et de points de cassure mais aussi lors du calcul du nombre d'*intervalles communs* à l'aide de programmes linéaires à variables booléennes [6]. L'observation que nous pouvons faire pour ce cas non résolu est qu'il a en entrée deux génomes de grande taille (3362 et 3388) et relativement "similaires" : `Ypest-CO92` et `Ypest-KIM`. En effet, le pourcentage d'adjacences par rapport à la taille du couplage, pour le modèle maximum, est égal à 98% pour ces deux génomes, contre 49% en moyenne (voir Tableau 3.11). C'est le seul couple de génomes à avoir autant de gènes et une similarité aussi importante. C'est pourquoi nous suspectons fortement que la structure des génomes joue un rôle dans ce problème.

Modèle intermédiaire. Rappelons tout d'abord que minimiser le nombre de points de cassure et maximiser le nombre d'adjacences *ne* sont *pas* des problèmes équivalents

$G_1 - G_2$	OPT_E		OPT_M		OPT_{IA}		OPT_{IB}	
	adj(C)	pdc(C)	adj(C)	pdc(C)	adj(C)	pdc(C)	adj(C)	pdc(C)
BAPHI-ECOLI	368	152	377	156	378	154	372	152
BAPHI-HAEIN	157	265	161	270	162	267	158	265
BAPHI-PAERU	226	232	229	240	230	237	227	232
BAPHI-PMULT	182	254	188	259	189	256	184	254
BAPHI-SALTY	367	154	376	158	377	155	372	154
BAPHI-WGLOS	201	168	203	170	203	168	201	168
BAPHI-XAXON	183	222	188	226	188	223	184	222
BAPHI-XCAMP	183	222	188	226	188	223	183	222
BAPHI-XFAST	158	231	162	236	162	234	158	231
BAPHI-YPEST-CO92	352	166	361	170	362	167	355	166
BAPHI-YPEST-KIM	339	172	348	176	349	172	343	172
ECOLI-HAEIN	489	610	550	665	551	624	507	610
ECOLI-PAERU	570	847	651	1082	668	906	597	847
ECOLI-PMULT	593	622	662	703	670	647	612	622
ECOLI-SALTY	2465	153	2874	277	X	X	2465	153
ECOLI-WGLOS	371	183	380	192	382	187	374	183
ECOLI-XAXON	380	675	425	842	437	718	390	675
ECOLI-XCAMP	378	678	420	845	432	717	386	678
ECOLI-XFAST	301	491	324	564	329	503	308	491
ECOLI-YPEST-CO92	1559	426	1744	596	1789	463	1559	426
ECOLI-YPEST-KIM	1560	422	1747	607	1798	464	1560	422
HAEIN-PAERU	301	550	333	615	337	567	309	550
HAEIN-PMULT	755	497	849	525	849	509	794	497
HAEIN-SALTY	492	612	550	676	553	635	515	612
HAEIN-WGLOS	159	267	165	277	165	271	163	267
HAEIN-XAXON	217	473	241	533	243	482	221	473
HAEIN-XCAMP	216	473	238	530	239	485	218	473
HAEIN-XFAST	191	424	205	468	207	440	194	424
HAEIN-YPEST-CO92	465	597	522	649	523	617	481	597
HAEIN-YPEST-KIM	460	598	517	653	518	622	479	598
PAERU-PMULT	340	592	373	681	380	627	347	592
PAERU-SALTY	559	862	644	1091	658	918	585	862
PAERU-WGLOS	246	248	251	259	253	249	249	248
PAERU-XAXON	536	802	609	1016	620	863	562	802
PAERU-XCAMP	536	801	596	1012	609	847	557	801
PAERU-XFAST	372	499	405	572	410	522	389	499
PAERU-YPEST-CO92	571	790	671	990	685	853	604	790
PAERU-YPEST-KIM	566	786	662	1004	681	855	598	786
PMULT-SALTY	597	622	670	704	677	650	620	622
PMULT-WGLOS	188	262	197	270	197	265	190	262
PMULT-XAXON	245	495	274	557	277	506	248	495
PMULT-XCAMP	241	495	267	555	270	507	245	495
PMULT-XFAST	213	436	234	481	238	451	221	436
PMULT-YPEST-CO92	582	597	648	671	654	621	602	597
PMULT-YPEST-KIM	574	601	639	676	644	631	588	601
SALTY-WGLOS	372	181	381	190	383	185	376	181
SALTY-XAXON	376	684	434	854	445	718	391	684
SALTY-XCAMP	375	684	427	854	440	716	389	684
SALTY-XFAST	300	497	325	569	329	509	310	497
SALTY-YPEST-CO92	1560	439	1758	591	1793	483	1560	439
SALTY-YPEST-KIM	1562	439	1761	606	1800	477	1562	439
WGLOS-XAXON	189	261	194	269	195	266	189	261
WGLOS-XCAMP	189	260	194	268	195	266	189	260
WGLOS-XFAST	158	264	163	272	164	267	160	264
WGLOS-YPEST-CO92	369	182	377	192	380	186	373	182
WGLOS-YPEST-KIM	356	186	364	196	367	187	361	186
XAXON-XCAMP	2880	114	3256	181	X	X	2880	114
XAXON-XFAST	980	375	1075	400	1077	380	1026	375
XAXON-YPEST-CO92	372	624	420	760	432	654	386	624
XAXON-YPEST-KIM	368	624	422	760	432	661	385	624
XCAMP-XFAST	969	373	1060	404	1065	383	1005	373
XCAMP-YPEST-CO92	369	620	412	755	424	644	380	620
XCAMP-YPEST-KIM	365	618	412	749	420	655	379	618
XFAST-YPEST-CO92	298	473	323	542	327	485	306	473
XFAST-YPEST-KIM	292	477	315	545	318	487	298	477
YPEST-CO92-YPEST-KIM	X	X	3327	59	X	X	X	X

TABLE 3.4 – **Nombre d'adjacences** $\mathtt{adj}(\mathcal{C})$ **et nombre de points de cassure** $\mathtt{pdc}(\mathcal{C})$ **pour les problèmes** $\mathbf{opt_E}$**,** $\mathbf{opt_M}$**,** $\mathbf{opt_{IA}}$ **et** $\mathbf{opt_{IP}}$**.** $\mathcal{C}$ représente toujours la solution retournée et X représente les cas non résolus.

pour le modèle intermédiaire. Concernant le problème de maximisation du nombre d'adjacences, 63 sur 66 résultats ont été obtenus en environ 16 minutes. Pour la minimisation du nombre de points de cassure, 59 sur 66 résultats ont été obtenus en moins d'une heure. Nous notons que, de nouveau, les paires de génomes pour lequel nous n'avons pu obtenir de résultats ont une grande similarité entre leurs deux génomes. C'est-à-dire que le pourcentage d'adjacences par rapport à la taille du couplage, pour le modèle maximum, est supérieur à 74% uniquement pour ces paires (voir Tableau 3.11).

Une justification doit être donnée ici pour le calcul du nombre minimum de points de cassure pour le modèle intermédiaire (problème $\mathbf{opt}_{\mathrm{IP}}$). En effet, minimiser le nombre de points de cassure pour les modèles exemplaire et intermédiaire ont été montrés comme des problèmes équivalents dans Section 3.2.1 (voir la Proposition 3.9), et donc la colonne de droite du Tableau 3.4 peut apparaître superflue ; en effet il peut être vérifié que dans le Tableau 3.4 les colonnes $\mathtt{pdc}(\mathcal{C})$ pour le modèle exemplaire ($\mathbf{opt}_{\mathrm{E}}$) et pour le modèle intermédiaire ($\mathbf{opt}_{\mathrm{IP}}$) sont toujours identiques.

La raison principale pour la représentation de ces deux résultats est de mettre en avant le fait que, bien que nous obtenons le même nombre minimum de points de cassure que pour le modèle exemplaire, un couplage minimisant le nombre de points de cassure pour le modèle intermédiaire peut comporter plus d'adjacences (voir l'exemple à la fin de la Section 3.2.2 pour une illustration simple). Cela pourrait être utile pour l'obtention d'une solution qui réalise un double objectif : minimiser le nombre de points de cassure (objectif primaire) et, parmi les solutions optimales, maximiser le nombre d'adjacences (objectif secondaire). Ce dernier objectif devrait être, cependant, modulé ici puisque l'objectif secondaire n'est pas explicitement indiqué dans le programme linéaire à variables booléennes, c'est-à-dire qu'il n'est pas garanti que le nombre maximum d'adjacences soit atteint.

Pré-couplage Pour ces trois modèles, un pré-couplage a été réalisé (voir Section 3.2.2). Avant de comparer les résultats pour ces trois modèles, étudions l'apport de ce pré-couplage.

Pour le modèle exemplaire, nous avons en moyenne 2488 gènes, dont 1668 dont la famille n'a qu'une occurrence sur G_1 et sur G_2 et que nous pouvons donc immédiatement coupler. Le pré-couplage permet de coupler en moyenne 192 autres gènes, il reste alors, en moyenne, 628 gènes à traiter. Sous le modèle maximum, nous avons en moyenne 2547 gènes, dont 1649 dont la famille n'a qu'une occurrence sur G_1 et sur G_2 et que nous pouvons donc immédiatement coupler. La règle permet de coupler en moyenne 74 autres gènes, il reste alors en moyenne 824 gènes à traiter.

Le détail de ces moyennes est indiqué dans le Tableau 3.10. Concrètement, ces pré-couplages nous ont permis, pour les trois modèles, de diminuer nos temps de calculs. De

plus, nous avons pu résoudre, pour le modèle exemplaire, 4 programmes linéaires à variables booléennes irrésolus jusqu'à alors pour cause de problème de mémoire.

Comparaison des trois modèles

Nous nous penchons maintenant sur la comparaison des résultats pour les différents modèles. Premièrement, les deux rapports (moyens) suivants peuvent être calculés à partir du Tableau 3.4 :

$$\frac{\mathtt{pdc}_E^{\mathbf{opt}}}{\mathtt{pdc}_M^{\mathbf{opt}}} = 0,89 \qquad \text{et} \qquad \frac{\mathtt{adj}_E^{\mathbf{opt}}}{\mathtt{adj}_M^{\mathbf{opt}}} = 0,92.$$

Ces deux rapports illustrent le fait que (i) le nombre minimum de points de cassure pour les modèles exemplaire et maximum diffèrent d'environ 10%, et (ii) le nombre maximum d'adjacences pour les modèles exemplaire et maximum différent aussi d'environ 10%.

À la lumière de ces rapports, il convient donc de souligner ici que, pour la plupart des applications pratiques, le choix du modèle (exemplaire ou maximum) sur lequel nous nous focalisons ne doit pas être sous-estimée, car différents résultats peuvent découler de cette décision. En effet, si nous désirons mettre en avant des groupes de synténie, il serait plus judicieux d'optimiser une de ces mesures pour le modèle maximum. Par contre, si nous voulons minimiser les différences entre deux génomes, il convient de choisir de travailler sous le modèle exemplaire.

Sous le modèle intermédiaire, nous avons déjà vu que $\mathtt{pdc}_E^{\mathbf{opt}} = \mathtt{pdc}_I^{\mathbf{opt}}$. Nous comparons dans le Tableau 3.5 les résultats pour le modèle intermédiaire avec les résultats pour les deux autres modèles. Pour compléter le Tableau 3.5, nous indiquons dans le Tableau 3.6 combien de fois la même mesure optimale (nombre de points de cassure ou nombre d'adjacences) est trouvée pour le modèle intermédiaire et pour un autre modèle (exemplaire ou maximum). Nous pouvons noter ici que, pour la majorité des cas, nous obtenons plus d'adjacences en utilisant le modèle intermédiaire. La situation est plus contrastée lorsque nous considérons le nombre de points de cassure : pour l'un, nous avons $\mathtt{pdc}_E^{\mathbf{opt}} = \mathtt{pdc}_I^{\mathbf{opt}}$ (dénoté par 100% dans le Tableau 3.6), et pour l'autre, pour 3% des comparaisons, nous obtenons le même nombre de points de cassure quand nous maximisons le nombre d'adjacences pour le modèle intermédiaire que le modèle exemplaire.

Pour comparer $\mathtt{adj}_I^{\mathbf{opt}}$ et $\mathtt{pdc}_I^{\mathbf{opt}}$, les deux rapports (moyens) suivants peuvent être comparer (voir Tableau 3.5) :

$$\frac{\mathtt{adj}(\mathcal{C}_{I,\mathtt{pdc}}^{\mathbf{opt}})}{\mathtt{adj}_I^{\mathbf{opt}}} = 0,94 \qquad \text{et} \qquad \frac{\mathtt{pdc}_I^{\mathbf{opt}}}{\mathtt{pdc}(\mathcal{C}_{I,\mathtt{adj}}^{\mathbf{opt}})} = 0,97.$$

Le modèle intermédiaire nous donne l'opportunalité de mener le calcul avec un double objectif (minimiser le nombre de points de cassure et maximiser le nombre d'adjacences).

	$\mathtt{adj}_I^{\mathbf{opt}}$	$\mathtt{adj}(\mathcal{C}_{I,\mathtt{pdc}}^{\mathbf{opt}})$	$\mathtt{pdc}(\mathcal{C}_{I,\mathtt{adj}}^{\mathbf{opt}})$	$\mathtt{pdc}_I^{\mathbf{opt}}$
$\mathtt{adj}_E^{\mathbf{opt}}$	**1,08**	**1,02**	-	-
$\mathtt{adj}_M^{\mathbf{opt}}$	**1,01**	0,95	-	-
$\mathtt{pdc}_E^{\mathbf{opt}}$	-	-	1,03	**1,00**
$\mathtt{pdc}_M^{\mathbf{opt}}$	-	-	**0,92**	**0,90**

TABLE 3.5 – **Comparaison des résultats** pour $\mathtt{adj}_I^{\mathbf{opt}}$: nombre maximum d'adjacences pour le modèle intermédiaire, $\mathtt{adj}(\mathcal{C}_{I,\mathtt{pdc}}^{\mathbf{opt}})$: nombre d'adjacences induit par le couplage $\mathcal{C}_{I,\mathtt{pdc}}^{\mathbf{opt}}$ produisant le nombre minimum de points de cassure pour le modèle intermédiaire, $\mathtt{pdc}(\mathcal{C}_{I,\mathtt{adj}}^{\mathbf{opt}})$: nombre de points de cassure induit par le couplage $\mathcal{C}_{I,\mathtt{adj}}^{\mathbf{opt}}$ produisant le nombre maximum d'adjacences pour le modèle intermédiaire, $\mathtt{pdc}_I^{\mathbf{opt}}$: nombre minimum de points de cassure pour le modèle intermédiaire, $\mathtt{adj}_E^{\mathbf{opt}}$: nombre maximum d'adjacences pour le modèle exemplaire, $\mathtt{adj}_M^{\mathbf{opt}}$: nombre maximum d'adjacences pour le modèle maximum, $\mathtt{pdc}_E^{\mathbf{opt}}$: nombre minimum de points de cassure pour le modèle exemplaire, $\mathtt{pdc}_M^{\mathbf{opt}}$: nombre minimum de points de cassure pour le modèle maximum. Les valeurs en gras indiquent les cas où le modèle intermédiaire est plus performant que le modèle exemplaire ou le modèle maximum. Nous montrons ici les rapports en moyenne, *e.g.*, $\frac{\mathtt{adj}_I^{\mathbf{opt}}}{\mathtt{adj}_E^{\mathbf{opt}}} = 1,10$, et donc, en moyenne, le nombre d'adjacences pour le modèle exemplaire est d'environ $1/1,10 = 90\%$ du nombre d'adjacences obtenus par le modèle intermédiaire.

Conformément à ce qui précède, il est possible d'argumenter que, pour le modèle intermédiaire, maximiser le nombre d'adjacences est meilleur que minimiser le nombre de points de cassure puisque la solution obtenue en maximisant le nombre d'adjacences donne environ $1/0,97 \approx 103\%$ du nombre minimum de points de cassure (minimiser le nombre de points de cassure donne environ 94% du nombre maximum d'adjacences). Cependant nous pensons que les deux rapports sont trop proches pour tirer une conclusion définitive sur la question de "$\mathtt{adj}_I^{\mathbf{opt}}$ *versus* $\mathtt{pdc}_I^{\mathbf{opt}}$".

L'idéal serait d'avoir un programme permettant d'optimiser deux mesures simultanément. Une première approche serait de choisir parmi tous les couplages optimisant une première mesure, celui qui optimise une deuxième mesure. Notons qu'un tel résultat pourrait être obtenu de la manière suivante : calculer le nombre minimum de points de cassure pour le modèle intermédiaire, transformer le programme linéaire à variables booléennes en ajoutant une contrainte qui oblige a avoir le nombre minimum de points de cassure déjà obtenu, et finalement, en modifiant l'objectif, calculer le nombre maximum d'adjacences. Inversement, nous pouvons utiliser une contrainte pour obtenir le nombre maximum d'adjacences, puis calculer le nombre minimum de points de cassure.

Le problème qui consiste à trouver un couplage qui minimise le nombre de points de

	$\mathtt{adj}_I^{\mathbf{opt}}$	$\mathtt{adj}(\mathcal{C}_{I,\mathtt{pdc}}^{\mathbf{opt}})$	$\mathtt{pdc}(\mathcal{C}_{I,\mathtt{adj}}^{\mathbf{opt}})$	$\mathtt{pdc}_I^{\mathbf{opt}}$
$\mathtt{adj}_E^{\mathbf{opt}}$	0%	8.5%	-	-
$\mathtt{adj}_M^{\mathbf{opt}}$	11%	0%	-	-
$\mathtt{pdc}_E^{\mathbf{opt}}$	-	-	3%	100%
$\mathtt{pdc}_M^{\mathbf{opt}}$	-	-	0%	0%

TABLE 3.6 – **Pourcentage de cas où la même mesure optimale (le nombre maximum d'adjacences et le nombre minimum de points de cassure) a été trouvé pour le modèle intermédiaire et pour les deux autres modèles (exemplaire et maximum).** Par exemple, pour 5 des 58 paires de génomes c'est-à-dire $8,5\%$, nous observons que $\mathtt{adj}_E^{\mathbf{opt}} = \mathtt{adj}(\mathcal{C}_{I,\mathtt{pdc}}^{\mathbf{opt}})$ (voir Tableau 3.5 pour les notations).

cassure avec le maximum d'adjacences sera appelé $\mathbf{opt}_{\text{IPA}}$; le problème qui consiste à trouver un couplage qui maximise le nombre d'adjacences avec le minimum de points de cassure sera appelé $\mathbf{opt}_{\text{IAP}}$. Un travail préliminaire nous a permis d'obtenir les résultats pour le problème $\mathbf{opt}_{\text{IPA}}$(voir Tableau 3.12). Nous avons donc un nombre minimal de points de cassure et un nombre d'adjacences maximisé. Nous avons 8 (sur 66) exécutions qui n'ont pas été achevées à cause d'un problème de mémoire. Le nombre d'adjacences obtenu est à 26 reprises identique à $\mathtt{adj}_I^{\mathbf{opt}}$ (nombre optimal d'adjacences pour le modèle intermédiaire). En moyenne, nous avons $99,06\%$ du nombre d'adjacences optimales (le minimum étant 97%).

Du point de vue des mesures, cette approche est donc évidemment plus performante que celle qui consiste uniquement à minimiser le nombre de points de cassure qui était à 94% du nombre d'adjacences optimale. Mais, comme il était à prévoir, le temps de calcul et le besoin de mémoire sont plus importants pour résoudre $\mathbf{opt}_{\text{IPA}}$ que pour résoudre $\mathbf{opt}_{\text{IP}}$. Finalement, pour les plus petits génomes, nous obtenons d'excellents résultats, pour le modèle intermédiaire, pour l'optimisation de deux mesures simultanément : le nombre de points de cassure et le nombre d'adjacences.

Nombres de couplages optimaux et inversion d'objectif

Afin de mieux comprendre la structure des génomes et leur influence sur nos algorithmes, nous calculons (i) le nombre de couplages optimaux et (ii) le nombre d'adjacences et de points de cassure possibles entre deux génomes.

Nombres de couplages optimaux Nous cherchons à calculer le nombre de couplages optimisant une mesure donnée, entre deux génomes, pour un modèle donné. En effet, il peut exister plusieurs couplages entre deux génomes induisant une même mesure optimale

tout en respectant un même modèle.

Pour cela, nous utilisons une première fois le programme linéaire à variables booléennes correspondant ; nous obtenons un premier couplage optimal. Puis nous ajoutons à notre programme une contrainte qui impose que le couplage solution soit différent de celui obtenu précédemment. La nouvelle contrainte impose que la somme des variables $a(i, k)$, $0 \leq i \leq n_1$, $0 \leq k \leq n_2$, égales à 1 dans la solution précédemment trouvée, doit être inférieure au nombre de ces variables. Nous itérons ainsi jusqu'à obtenir un couplage dont la mesure n'est plus la mesure optimale de départ. Nous avons alors l'ensemble des couplages optimisant une mesure donnée, pour un modèle donné. Pour bien obtenir tous les couplages, nous ne devons pas effectuer de pré-couplage (voir Section 3.2.2).

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92
ECOLI	> 1000										
HAEIN	16	X									
PAERU	24	X	> 514								
PMULT	48	X	X	507							
SALTY	576	X	> 430	X	X						
WGLOS	6	448	8	X	28	48					
XAXON	16	> 167	> 1000	X	> 1000	X	16				
XCAMP	16	X	512	X	> 1000	X	16	X			
XFAST	8	> 5470	576	X	> 1000	X	12	X	X		
Y-CO92	> 1000	X	> 1000	X	> 83	X	448	X	X	X	
Y-KIM	> 1000	X	> 594	X	X	X	448	X	X	X	X

Table 3.7 – **Nombres de couplages optimaux pour le modèle exemplaire** entre deux génomes qui maximisent le nombre d'adjacences. Par exemple, il y a plus de 1000 couplages de `BAPHI` et `ECOLI` qui maximisent le nombre d'adjacences. Le symbole X indique une absence de résultats.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92
ECOLI	14										
HAEIN	2	X									
PAERU	4	X	>3200								
PMULT	2	X	>2600	>1000							
SALTY	4	X	>2700	X	>700						
WGLOS	8	512	288	X	384	1024					
XAXON	16	>1000	>1000	X	>3580	>1100	32				
XCAMP	32	>1000	768	X	1024	>1000	64	>1830			
XFAST	16	>1000	432	>1000	128	512	32	>1000	>1000		
Y-CO92	16	X	>1000	X	>1000	X	2048	>1000	>1000	960	
Y-KIM	8	X	>1000	X	>1000	>1000	2048	>1000	>1000	896	>1000

Table 3.8 – **Nombres de couplages optimaux pour le modèle maximum** entre deux génomes qui maximisent le nombre d'adjacences. Par exemple, il y a 14 couplages de `BAPHI` et `ECOLI` qui maximisent le nombre d'adjacences. Le symbole X indique une absence de résultats.

En pratique, un travail préliminaire, pour les modèles exemplaire et maximum, montre qu'il peut exister beaucoup de solutions optimales pour un même problème (voir Tableaux 3.7 et 3.8). Nous notons aussi que le nombre de solutions est sensiblement plus important pour le modèle maximum que pour le modèle exemplaire.

Si nous regardons l'ensemble des couplages possibles entre deux génomes, nous observons que les variations n'affectent en fait que peu de gènes. En général, le choix de couplage de la plupart des familles de gènes n'influent pas la mesure finale. Bien sûr, nous n'avons pas de choix à faire pour les familles ne possédant qu'une occurrence ; mais d'autres familles n'affectent pas la mesure. Prenons comme illustration les génomes $G_1 = +a+b\ +x\ +c\ +d\ +e\ +y\ +f\ +g$ et $G_2 = +a+b\ +x\ +e\ +y\ +d\ +x\ +c\ +y\ +f\ +g$. Quel que soit le modèle choisi, l'occurrence du gène x sur le génome G_1 sera couplé soit à la première, soit à la deuxième occurrence sur le génome G_2 sans que cela n'affecte le nombre d'adjacences (ou le nombre de points de cassure). De même, pour le gène y. Au final, nous avons 4 couplages possibles pour lesquels la mesure optimale (le nombre minimum d'adjacences -4- ou le nombre maximum de points de cassure -4-) est obtenue.

Inversion d'objectif Nous étudions le nombre d'adjacences et le nombre de points de cassure possibles, pour un modèle et une paire de génomes donnés. Autrement dit, nous voulons connaître le nombre minimal et le nombre maximal de points de cassure et d'adjacences que nous pouvons obtenir à l'aide de tous les couplages respectant un modèle fixé.

Pour cela il suffit d'inverser l'objectif des programmes linéaires à variables booléennes présentés dans la Section 3.2.2 : soit nous *minimisons* le nombre d'adjacences, soit nous *maximisons* le nombre de points de cassure. Une fois de plus, nous ne devons pas effectuer de pré-couplage (voir Section 3.2.2) afin d'obtenir un résultat optimal. L'ensemble des résultats obtenus est présenté dans le Tableau 3.13. Nous obtenons tous les résultats en 20 secondes, c'est-à-dire plus rapidement que lorsque nous maximisons le nombre d'adjacences ou minimisons le nombre de points de cassure.

La différence entre le maximum et le minimum d'une mesure (adjacences ou points de cassure) est comprise entre 14 et 1353 et vaut en moyenne environ 200. Ceci correspond à un écart moyen valant $14,5\%$ de la taille des couplages finaux. Nous rappelons que 27% des gènes sont dupliqués (après le prétraitement).

3.3.3 Résultats d'heuristiques

Chacune des neuf heuristiques a été testée sur l'ensemble des génomes de γ-Protéobacterie décrit dans la Sous-section 3.3.1. Une synthèse des résultats obtenus est donnée dans le Tableau 3.9. Nous y indiquons, pour les neuf heuristiques, la différence entre leurs résultats et les résultats exacts en moyenne, au pire et au meilleur des cas.

L'ensemble complet des résultats est donné dans les Tableaux 3.14, 3.15 et 3.16 (Section-Annexe 3.5) pour les problèmes $\mathbf{opt}_{\mathrm{M}}$, $\mathbf{opt}_{\mathrm{IA}}$ et $\mathbf{opt}_{\mathrm{E}}$, respectivement. Une

représentation graphique montrant, pour chaque problème, la comparaison entre chaque heuristique et les résultats exacts est présentée dans les Figures 3.8, 3.9 et 3.10 dans la Section-Annexe 3.5 (notons que, dans chaque figure, les résultats sont présentés dans le même ordre arbitraire que les Tableaux 3.14, 3.15 et 3.16).

Concernant les heuristiques basées sur `IILCS`, leurs temps de calculs est approximativement de 40 minutes pour chacun de ces trois problèmes (nous rappelons que, pour chaque paire de génomes, toutes les heuristiques de ce type sont exécutées 10 fois, donc il faut approximativement 4 minutes pour que chaque heuristique achève les 66 comparaisons).

Comme il a été dit précédemment, à propos des heuristiques basées sur la méthode hybride, nous les avons exécutées, pour chacun des trois modèles (i) avec le paramètre k fixé à 2 et (ii) avec le paramètre k fixé à 3. Le temps de calcul total, pour chacune de ces six heuristiques, est d'environ cinq minutes.

	Exemplaire - $\mathtt{adj}_E^{\mathbf{opt}}$			Intermédiaire - $\mathtt{adj}_I^{\mathbf{opt}}$			Maximum- $\mathtt{adj}_M^{\mathbf{opt}}$		
Heuristique	IILCS_E	HYB_E(2)	HYB_E(3)	IILCS_IA	HYB_IA(2)	HYB_IA(3)	IILCS_M	HYB_M(2)	HYB_M(3)
Moyenne	99.36%	99.97%	99.99%	90,56%	99,48%	99,82%	99,05%	99,89%	99,97%
Pire cas	97,89%	99,53%	99,84%	82,09%	98,09%	98,78%	97,53%	99,50%	99,67%
Meilleur cas	100%	100%	100%	98,52%	100%	100%	100%	100%	100%
Nombre d'instances pour lesquelles le résultat optimal a été obtenu	13	52 (sur 65)	59	0	17 (sur 63)	35	11	35 (sur 66)	55

TABLE 3.9 – **Résumé des résultats pour les neuf heuristiques étudiées et comparaison avec les résultats exacts.** À titre d'illustration, pour le modèle intermédiaire, `HYB_IA(2)` offre des résultats qui sont en moyenne 99,48% du nombre optimal de points de cassure, entre 98,09% et 100%. De plus, dans 17 cas, sur les 63 cas résolus par notre programme linéaire à variables booléennes, `HYB_IA(2)` retourne la valeur optimale.

Globalement, les neuf heuristiques (`IILCS` et hybrides) que nous avons proposées sont très performantes pour notre ensemble de données. Pour chacun des trois modèles, l'heuristique `IILCS` appropriée retourne des résultats qui sont en moyenne au moins à 90% de la valeur optimale.

Heuristiques `IILCS` Nous pouvons considérer que `IILCS_IA` est relativement moins efficace que `IILCS_E` et `IILCS_M` : `IILCS_IA` retourne en moyenne des résultats à 90.56% de valeur optimale, tant dis que `IILCS_E` (respectivement `IILCS_M`) atteint 99.36% (respectivement 99%). Ceci peut-être expliqué par le fait que le critère d'arrêt de `IILCS_IA` (nous stoppons l'itération lorsqu'il y a au moins un gène par famille couplé) n'est pas performant pour mettre en valeur la spécificité du modèle intermédiaire. Ce critère d'arrêt permet de respecter le modèle intermédiaire mais il est tout à fait arbitraire. Néanmoins, les performances de `IILCS_IA`, qui ne sont pas aussi bonnes que celles de `IILCS_E` et de `IILCS_M`, restent satisfaisantes.

Notons que nous pourrions facilement améliorer l'heuristique `IILCS_IA` en modifiant la condition d'arrêt de l'itération. Rappelons qu'actuellement cette condition d'arrêt est le fait d'avoir au moins un gène par famille de couplé, nous permettant ainsi d'obtenir un couplage intermédiaire. Si nous continuions les itérations tant que cette condition n'est pas vérifiée et que la taille des LCS est inférieure à 2 nous obtenons toujours un couplage intermédiaire mais avec obligatoirement autant ou plus d'adjacences. Cette modification d'heuristique n'a pas encore été effectuée.

Heuristiques hybrides Sans surprise, nos méthodes hybrides, avec le paramètre $k = 2$, donne de meilleurs résultats que ceux correspondant aux heuristiques `IILCS` (nous sommes en moyenne à 99,48% des valeurs optimales pour les heuristiques les plus faibles). Sans surprise de nouveau, pour $k = 3$, les méthodes hybrides sont encore meilleures (nous sommes en moyenne à 99.82% des valeurs optimales pour les heuristiques les moins efficaces). Dans chaque cas, nous sommes en moyenne proche de la valeur optimale, et de nombreuses valeurs exactes sont obtenues.

Heuristiques aléatoires Une autre façon d'évaluer nos heuristiques est de comparer leurs résultats avec ceux d'une heuristique calculant une mesure à partir d'un couplage créer entièrement aléatoirement. Cette nouvelle heuristique sera nommé `ALEA_E` (respectivement `ALEA_I` et `ALEA_M`) pour le modèle exemplaire (respectivement pour les modèles intemédiaire et maximum). Nous notons qu'à la différence des heuristiques de type `IILCS`, nous n'itérons pas 10 fois cette heuristique.

Un travail préliminaire nous a permis d'obtenir tous les résultats pour le modèle maximum (voir Tableau 3.17 Section-Annexe 3.5). Nous observons que l'heuristique `ALEA_M` donne de très mauvais résultats : le rapport entre le nombre maximum d'adjacences et le nombre d'adjacences obtenu par `ALEA_M` est de 78% en moyenne (64% au minimum et 95% au maximum) .

Il y a deux conclusions principales qui peuvent déduites de cet ensemble de résultats.

Premièrement, chacune des neuf heuristiques, basées sur la recherche de LCS, présentées ici est performante (temps de calcul court et résultats très proches des résultats optimaux) sur l'ensemble des données étudiées. Bien entendu, de meilleurs résultats sont obtenus en utilisant les méthodes hybrides `HYB(3)`, ou `HYB(2)`. Cependant, même après avoir calculé un couplage partiel *via* `IILCS` avec $k = 2$ ou $k = 3$, nous ne pourrons pas obtenir dans un temps raisonnable tous les résultats avec un programme linéaire à variables booléennes. En particulier, cette situation survient lorsque les génomes sont de très grandes tailles. Dans ce cas, les bonnes performances des heuristiques `IILCS` montrent que nous pouvons toujours les utiliser pour obtenir des résultats rapides et bons.

Deuxièmement, ces résultats montrent que l'idée assez intuitive consistant à trouver et coupler itérativement les gènes des plus Longues Sous-séquences Communes est très efficace pour les deux mesures (nombre d'adjacences et nombre de points de cassure) et pour les trois modèles (exemplaire, intermédiaire, maximum). Il est à noter que la même conclusion a été observée dans [6] concernant la mesure du nombre d'*intervalles communs* pour le modèle maximum.

3.4 Conclusion

Ce travail portait sur les problèmes de calcul de mesures (nombre d'adjacences et de points de cassure) entre deux génomes avec des gènes dupliqués, pour trois modèles : exemplaire, intermédiaire et maximum.

Dans un premier temps, nous avons élargi les résultats présentés dans [28] en prouvant que décider s'il existe un couplage entre deux génomes dont aucun gène n'est présent plus de deux fois, pour les modèles exemplaire et intermédiaire, tel que le nombre de points de cassure est nul est **NP**-complet. Par conséquent, nous pouvons affirmer que, pour ces deux modèles, il n'existe aucun algorithme d'approximation résolvant en temps polynomial le problème de minimiser le nombre de points de cassure. D'après la Proposition 3.8, pour le modèle exemplaire, il n'existe aucun algorithme d'approximation résolvant en temps polynomial le problème de maximiser le nombre d'adjacences.

Sous le modèle maximum, décider si le nombre de points de cassure est nul entre deux génomes est un problème polynomial. Nous ne pouvons donc pas actuellement conclure sur l'approximation du problème d'optimisation du nombre de points de cassure (ou, d'après la Proposition 3.8 du nombre d'adjacences), pour ce modèle.

Ce travail a aussi été développé dans le but de construire des modèles plus précis pour la génomique comparative et de concevoir des heuristiques efficaces et rapides basées sur les mesures d'adjacences et de points de cassure dans le cas où les duplications de gènes sont prises en compte.

En ce sens, les résultats que nous avons obtenus sont très satisfaisants : d'une part, en pratique quasiment tous les résultats ont été obtenus *via* l'approche exacte de programmation linéaire à variables booléennes (mais il semble difficile de pousser plus loin avec des génomes de plus grande taille). D'autre part, toutes les heuristiques que nous avons proposées sont très performantes sur notre ensemble de données. En effet, leurs résultats sont proches des résultats exacts et même souvent identiques, en particulier pour les heuristiques hybrides. Un bémol tout de même pour l'heuristique `IILCS_I` qui est légèrement moins performante. Il serait intéressant de voir si elle est significativement supérieure à l'heuristique aléatoire `ALEA_I`, tout comme l'est l'heuristique `IILCS_M` par rapport à `ALEA_M`.

Un autre aspect de notre travail a été de se focaliser sur le modèle intermédiaire. En effet, notre principale motivation réside sur le fait que les modèles exemplaire et maximum peuvent être en pratique trop restrictifs pour des applications. Plus précisément, si les deux modèles exemplaire et maximum fournissent un cadre algorithmique clair et simple, nous croyons que le modèle intermédiaire est mieux adapté et plus efficace pour la génomique comparative. De ce point de vue, l'un de nos objectifs était d'observer si les résultats de ce nouveau modèle seraient très différents de ceux des modèles exemplaire et

maximum. Il s'avère que les résultats ne sont pas concluants : (i) $\mathbf{opt}_{\mathrm{IP}}$ est équivalent à $\mathbf{opt}_{\mathrm{E}}$ (Proposition 3.9), et (ii) la résolution de $\mathbf{opt}_{\mathrm{IA}}$ retourne des résultats qui, en termes d'adjacences, sont relativement proches de ceux de $\mathbf{opt}_{\mathrm{E}}$ et de $\mathbf{opt}_{\mathrm{M}}$ (Tableaux 3.5 et 3.6). Par conséquent, pour étudier la pertinence du modèle intermédiaire pour notre base de données et nos mesures, il faudrait une étude plus approfondie, et notamment une étude de la structure des génomes d'entrée (combien de gènes de chaque famille sont conservés et quelles sont leur position, par exemple).

Même si nous n'avons pas été en mesure de distinguer clairement, en utilisant le nombre de points de cassure et le nombre de adjacences, si le modèle intermédiaire diffère radicalement des autres modèles, nous croyons encore en l'intérêt de ce modèle. Premièrement, le fait de donner plus de liberté dans la structure de la solution est certainement un avantage pour des applications pratiques. Deuxièmement, comme l'illustre le Tableau 3.4, le modèle intermédiaire donne en fait lieu à deux problèmes combinatoires : minimiser le nombre de points de cassure *et* maximiser le nombre d'adjacences. Un complément de cette recherche a été donc de développer une approche basée sur un programme linéaire à variables booléennes pour atteindre ce double objectif. Le temps et la mémoire sont couteux mais les résultats préliminaires sont encourageant pour des génomes de petites tailles.

Une autre partie de ce travail a été de mettre en place des règles permettant de diminuer la complexité des programmes ; soit en calculant, lors d'un prétraitement, les valeurs de certaines variables des programmes de façon polynomiale, soit en débutant un couplage (pré-couplage) sans perdre l'optimalité recherchée. Ces deux approches sont clairement profitables à la résolution des programmes et nous pensons qu'il serait bon d'ajouter de nouvelles règles en particulier pour le modèle intermédiaire.

De plus, différents travaux ont débuté afin de mieux comprendre les difficultés inhérentes aux problèmes et aux génomes étudiés : inverser les objectifs et calculer le nombre de couplage optimaux. Premièrement, nous avons pu observer qu'il existe un grand écart entre le nombre minimal et le nombre maximal d'adjacences (ou de points de cassures). Il serait intéressant de voir si cet écart et le temps d'exécution sont corrélés.

Deuxièmement, l'étude de l'ensemble des couplages optimaux pour un génome, une mesure et un modèle donnés, permet de mettre en avant les ensembles de gènes couplés toujours de la même manière. Malheureusement, actuellement, il n'est pas évident d'obtenir tous les couplages optimaux à cause de leur nombre important.

Notons que l'utilisation de la programmation linéaire à variables booléennes pour traduire un problème de comparaison de génomes, *via* l'optimisation d'une mesure, peut-être étudiée pour d'autres mesures ; de même que les heuristiques basées sur la recherche de LCS. Dans ce sens nous avons mené déjà un travail préliminaire pour mesurer de façon exacte le MAD (le nombre maximum de perturbation d'adjacences). Malheureusement le

programme linéaire à variables booléennes que nous proposons a un temps de résolution trop important (seul une petite dizaine de couples de génomes ont pu être ainsi comparés). Par ailleurs, cette méthode a aussi été testée pour calculer la mesure SAD mais, de la même manière, peu de résultat ont pu être obtenu [33].

Dans la section suivante, nous comparons cette fois des génomes sans prendre en compte leurs duplications mais pour lequel nous n'avons qu'un ordre partiel de leurs gènes.

3.5 Annexe : détails des résultats

Cette section regroupe les tableaux et les figures cités dans ce chapitre.

- Tableau 3.10 page 114 : données du pré-couplage.
- Tableau 3.11 page 115 : rapport $\mathtt{adj}(\mathcal{C})/|\mathcal{C}|$.
- Tableau 3.12 page 116 : résultats de $\mathbf{opt}_{\mathrm{IA}}$, $\mathbf{opt}_{\mathrm{IP}}$ et $\mathbf{opt}_{\mathrm{IPA}}$.
- Tableau 3.13 page 117 : mesure minimisée et maximisée.
- Figure 3.8 page 118 : rapport $\mathtt{pdc}_M^{\mathbf{opt}}/\mathtt{pdc}_M^H$ pour le modèle maximum.
- Tableau 3.14 page 119 : le nombre d'adjacences $\mathtt{adj}(\mathcal{C})$ et le nombre de points de cassure $\mathtt{pdc}(\mathcal{C})$ pour le modèle maximum.
- Figure 3.9 page 120 : représentation graphique du rapport $\mathtt{adj}_I^H/\mathtt{adj}_I^{\mathbf{opt}}$ pour le modèle intermédiaire.
- Tableau 3.15 page 121 : le nombre d'adjacences $\mathtt{adj}(\mathcal{C})$ et le nombre de points de cassure $\mathtt{pdc}(\mathcal{C})$ pour le modèle intermédiaire.
- Figure 3.10 page 122 : rapport $\mathtt{pdc}_E^{\mathbf{opt}}/\mathtt{pdc}_E^H$ pour le modèle exemplaire.
- Tableau 3.16 page 123 : le nombre d'adjacences $\mathtt{adj}(\mathcal{C})$ et le nombre de points de cassure $\mathtt{pdc}(\mathcal{C})$ pour le modèle exemplaire.
- Tableau 3.17 page 124 : couplage maximum quelconque.

G_1 - G_2	Exemplar					Maximum/Intermédiaire				
	nb_gènes	triviaux	pré-couplé	restant	final_E	nb_gènes	triviaux	pré-couplé	restant	final_M
BAPHI-ECOLI	1252	906	56	290	521	1280	900	30	350	534
BAPHI-HAEIN	932	768	3	161	423	940	764	2	174	432
BAPHI-PAERU	1179	764	25	390	459	1205	760	16	429	470
BAPHI-PMULT	973	794	4	175	437	984	790	2	192	448
BAPHI-SALTY	1256	902	50	304	522	1275	896	28	351	535
BAPHI-WGLOS	762	714	7	41	370	770	702	4	64	374
BAPHI-XAXON	961	724	15	222	406	982	718	2	262	415
BAPHI-XCAMP	956	726	16	214	406	977	718	2	257	415
BAPHI-XFAST	860	714	0	146	390	867	708	0	159	399
BAPHI-YPEST-CO92	1251	892	117	242	519	1278	886	30	362	532
BAPHI-YPEST-KIM	1241	880	35	326	512	1274	874	14	386	525
ECOLI-HAEIN	2717	1804	151	762	1100	2782	1778	38	966	1216
ECOLI-PAERU	3977	2094	197	1686	1418	4059	2072	50	1937	1734
ECOLI-PMULT	3007	2008	187	812	1216	3069	1994	38	1037	1366
ECOLI-SALTY	6532	4380	1869	283	2619	6628	4348	1060	1220	3152
ECOLI-WGLOS	1339	958	87	294	555	1372	940	20	412	573
ECOLI-XAXON	2800	1646	147	1007	1056	2886	1620	22	1244	1268
ECOLI-XCAMP	2792	1662	147	983	1057	2870	1636	24	1210	1266
ECOLI-XFAST	2004	1302	42	660	793	2048	1290	16	742	889
ECOLI-YPEST-CO92	4963	3328	716	919	1986	5053	3306	260	1487	2341
ECOLI-YPEST-KIM	4989	3314	781	894	1983	5111	3272	260	1579	2355
HAEIN-PAERU	2276	1386	131	759	852	2323	1370	24	929	949
HAEIN-PMULT	2793	2290	181	322	1253	2855	2250	62	543	1375
HAEIN-SALTY	2747	1788	146	813	1105	2802	1766	36	1000	1227
HAEIN-WGLOS	937	786	0	151	427	957	768	0	189	443
HAEIN-XAXON	1718	1168	10	540	691	1771	1158	4	609	775
HAEIN-XCAMP	1691	1178	10	503	690	1740	1168	4	568	769
HAEIN-XFAST	1429	1074	6	349	616	1455	1064	4	387	674
HAEIN-YPEST-CO92	2573	1824	65	684	1063	2640	1794	34	812	1172
HAEIN-YPEST-KIM	2584	1810	159	615	1059	2669	1780	36	853	1171
PAERU-PMULT	2460	1524	126	810	933	2511	1512	16	983	1055
PAERU-SALTY	3978	2108	189	1681	1422	4048	2090	48	1910	1736
PAERU-WGLOS	1279	834	20	425	495	1320	820	10	490	511
PAERU-XAXON	3680	2062	189	1429	1339	3768	2036	36	1696	1626
PAERU-XCAMP	3687	2070	189	1428	1338	3766	2044	38	1684	1609
PAERU-XFAST	2273	1418	51	804	872	2321	1396	24	901	978
PAERU-YPEST-CO92	3790	2084	244	1462	1362	3865	2064	60	1741	1662
PAERU-YPEST-KIM	3796	2048	207	1541	1353	3894	2022	42	1830	1667
PMULT-SALTY	3033	1992	180	861	1220	3086	1982	38	1066	1375
PMULT-WGLOS	998	836	7	155	451	1018	816	2	200	468
PMULT-XAXON	1827	1268	37	522	741	1885	1256	12	617	832
PMULT-XCAMP	1799	1270	36	493	737	1853	1260	12	581	823
PMULT-XFAST	1502	1152	21	329	650	1541	1142	10	389	716
PMULT-YPEST-CO92	2844	2040	89	715	1180	2907	2024	44	839	1320
PMULT-YPEST-KIM	2860	2018	183	659	1176	2942	2000	40	902	1316
SALTY-WGLOS	1348	948	102	298	554	1374	930	30	414	572
SALTY-XAXON	2832	1648	138	1046	1061	2907	1622	24	1261	1289
SALTY-XCAMP	2822	1650	139	1033	1060	2888	1628	26	1234	1282
SALTY-XFAST	2030	1284	47	699	798	2065	1274	20	771	895
SALTY-YPEST-CO92	5014	3326	706	982	2000	5096	3300	252	1544	2350
SALTY-YPEST-KIM	5050	3316	739	995	2002	5159	3276	256	1627	2368
WGLOS-XAXON	1075	802	3	270	451	1114	786	2	326	464
WGLOS-XCAMP	1067	802	3	262	450	1105	784	2	319	463
WGLOS-XFAST	933	776	3	154	423	949	758	2	189	436
WGLOS-YPEST-CO92	1332	948	52	332	552	1369	930	28	411	570
WGLOS-YPEST-KIM	1321	930	133	258	543	1367	910	28	429	561
XAXON-XCAMP	6900	5448	1291	161	2995	7078	5336	956	786	3438
XAXON-XFAST	3133	2430	315	388	1356	3217	2396	112	709	1476
XAXON-YPEST-CO92	2618	1604	65	949	997	2695	1586	20	1089	1181
XAXON-YPEST-KIM	2629	1590	155	884	993	2722	1566	12	1144	1183
XCAMP-XFAST	3093	2408	294	391	1343	3176	2368	102	706	1465
XCAMP-YPEST-CO92	2576	1612	67	897	990	2648	1594	22	1032	1168
XCAMP-YPEST-KIM	2583	1592	154	837	984	2669	1566	14	1089	1162
XFAST-YPEST-CO92	1922	1296	48	578	772	1972	1280	18	674	866
XFAST-YPEST-KIM	1932	1284	55	593	770	1991	1264	22	705	861
YPEST-CO92-YPEST-KIM	6750	5146	1015	589	2848	6893	5102	392	1399	3387

TABLE 3.10 – **Pré-couplage.** Après le pré-traitement, nous avons `nb_gènes` gènes sur les deux génomes comparés G_1 et G_2. Parmis ces `nb_gènes`, les gènes `triviaux` possédant qu'une occurrence sur chacun des génomes, peuvent être couplés directement. Notre étape de pré-couplage permet alors de coupler encore `pré-couplé` gènes. Il reste `restant` gènes à coupler sur l'ensemble des deux génomes. Finalement, le couplage qui optimise une mesure a `final_E` (respectivement `finale_M`) gènes pour le modèle exemplaire (respectivement pour le modèle maximum). Rappelons que pour le modèle intermédiaire, l'ensemble des couplages optimisant une mesure entre G_1 et G_2 contient des couplages de taille différente.

G_1 - G_2	OPT_E	OPT_M	OPT_{IA}	OPT_{IB}
Min	31,01%	30,42%	31,94%	31,34%
Max	96,16%	98,23%	79,45%	73,18%
Moyenne	48,89%	48,76%	48,79%	45,91%
BAPHI-ECOLI	70,63%	70,60%	70,92%	70,86%
BAPHI-HAEIN	37,12%	37,27%	37,67%	37,26%
BAPHI-PAERU	49,24%	48,72%	49,15%	49,35%
BAPHI-PMULT	41,65%	41,96%	42,38%	41,91%
BAPHI-SALTY	70,31%	70,28%	70,73%	70,59%
BAPHI-WGLOS	54,32%	54,28%	54,57%	54,32%
BAPHI-XAXON	45,07%	45,30%	45,63%	45,21%
BAPHI-XCAMP	45,07%	45,30%	45,63%	45,07%
BAPHI-XFAST	40,51%	40,60%	40,81%	40,51%
BAPHI-YPEST-CO92	67,82%	67,86%	68,30%	68,01%
BAPHI-YPEST-KIM	66,21%	66,29%	66,86%	66,47%
ECOLI-HAEIN	44,45%	45,23%	46,85%	45,35%
ECOLI-PAERU	40,20%	37,54%	42,41%	41,31%
ECOLI-PMULT	48,77%	48,46%	50,83%	49,55%
ECOLI-SALTY	94,12%	91,18%	X	X
ECOLI-WGLOS	66,85%	66,32%	67,02%	67,03%
ECOLI-XAXON	35,98%	33,52%	37,80%	36,59%
ECOLI-XCAMP	35,76%	33,18%	37,57%	36,24%
ECOLI-XFAST	37,96%	36,45%	39,50%	38,50%
ECOLI-YPEST-CO92	78,50%	74,50%	79,41%	X
ECOLI-YPEST-KIM	78,67%	74,18%	79,45%	X
HAEIN-PAERU	35,33%	35,09%	37,24%	35,93%
HAEIN-PMULT	60,26%	61,75%	62,47%	61,46%
HAEIN-SALTY	44,52%	44,82%	46,51%	45,66%
HAEIN-WGLOS	37,24%	37,25%	37,76%	37,82%
HAEIN-XAXON	31,40%	31,10%	33,47%	31,80%
HAEIN-XCAMP	31,30%	30,95%	32,97%	31,50%
HAEIN-XFAST	31,01%	30,42%	31,94%	31,34%
HAEIN-YPEST-CO92	43,74%	44,54%	45,84%	44,58%
HAEIN-YPEST-KIM	43,44%	44,15%	45,40%	44,43%
PAERU-PMULT	36,44%	35,36%	37,70%	36,91%
PAERU-SALTY	39,31%	37,10%	41,72%	40,40%
PAERU-WGLOS	49,70%	49,12%	50,30%	50,00%
PAERU-XAXON	40,03%	37,45%	41,78%	41,17%
PAERU-XCAMP	40,06%	37,04%	41,80%	40,99%
PAERU-XFAST	42,66%	41,41%	43,94%	43,76%
PAERU-YPEST-CO92	41,92%	40,37%	44,51%	43,30%
PAERU-YPEST-KIM	41,83%	39,71%	44,31%	43,18%
PMULT-SALTY	48,93%	48,73%	50,98%	49,88%
PMULT-WGLOS	41,69%	42,09%	42,55%	41,94%
PMULT-XAXON	33,06%	32,93%	35,33%	33,33%
PMULT-XCAMP	32,70%	32,44%	34,70%	33,06%
PMULT-XFAST	32,77%	32,68%	34,49%	33,59%
PMULT-YPEST-CO92	49,32%	49,09%	51,25%	50,17%
PMULT-YPEST-KIM	48,81%	48,56%	50,47%	49,41%
SALTY-WGLOS	67,15%	66,61%	67,31%	67,38%
SALTY-XAXON	35,44%	33,67%	38,23%	36,34%
SALTY-XCAMP	35,38%	33,31%	38,03%	36,22%
SALTY-XFAST	37,59%	36,31%	39,21%	38,37%
SALTY-YPEST-CO92	78,00%	74,81%	78,74%	X
SALTY-YPEST-KIM	78,02%	74,37%	79,02%	X
WGLOS-XAXON	41,91%	41,81%	42,21%	41,91%
WGLOS-XCAMP	42,00%	41,90%	42,21%	42,00%
WGLOS-XFAST	37,35%	37,39%	37,96%	37,65%
WGLOS-YPEST-CO92	66,85%	66,14%	67,02%	67,09%
WGLOS-YPEST-KIM	65,56%	64,88%	66,13%	65,88%
XAXON-XCAMP	96,16%	94,71%	X	X
XAXON-XFAST	72,27%	72,83%	73,87%	73,18%
XAXON-YPEST-CO92	37,31%	35,56%	39,74%	38,18%
XAXON-YPEST-KIM	37,06%	35,67%	39,49%	38,12%
XCAMP-XFAST	72,15%	72,35%	73,50%	72,88%
XCAMP-YPEST-CO92	37,27%	35,27%	39,66%	37,96%
XCAMP-YPEST-KIM	37,09%	35,46%	39,03%	37,98%
XFAST-YPEST-CO92	38,60%	37,30%	40,22%	39,23%
XFAST-YPEST-KIM	37,92%	36,59%	39,45%	38,40%
YPEST-CO92-YPEST-KIM	X	98,23%	X	X

TABLE 3.11 – **Le rapport $\mathtt{adj}(\mathcal{C})/|\mathcal{C}|$ en pourcentage pour les couplages $\mathcal{C}$ obtenus par $\mathbf{opt_E}$, $\mathbf{opt_M}$, $\mathbf{opt_{IA}}$ et $\mathbf{opt_{IP}}$** ; X représente les cas non résolus. Ce rapport nous permet d'observer la similarité entre deux génomes. Remarquons que nous avons des difficultés pour obtenir un résultat pour les paires de génomes similaires (c'est-à-dire les paires pour le lequel le pourcentage est supérieur à 74% pour le modèle maximum).

	Opt-IA		Opt-IP		Opt-IPA	
$G_1 - G_2$	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)
BAPHI-ECOLI	378	154	372	152	378	152
BAPHI-HAEIN	162	267	158	265	162	265
BAPHI-PAERU	230	237	227	232	230	232
BAPHI-PMULT	189	256	184	254	189	254
BAPHI-SALTY	377	155	372	154	377	154
BAPHI-WGLOS	203	168	201	168	203	168
BAPHI-XAXON	188	223	184	222	188	222
BAPHI-XCAMP	188	223	183	222	188	222
BAPHI-XFAST	162	234	158	231	162	231
BAPHI-YPEST-CO92	362	167	355	166	362	166
BAPHI-YPEST-KIM	349	172	343	172	349	172
ECOLI-HAEIN	551	624	507	610	545	610
ECOLI-PAERU	668	906	597	847	648	847
ECOLI-PMULT	670	647	612	622	659	622
ECOLI-SALTY	X	X	X	X	X	X
ECOLI-WGLOS	382	187	374	183	382	183
ECOLI-XAXON	437	718	390	675	427	675
ECOLI-XCAMP	432	717	386	678	424	678
ECOLI-XFAST	329	503	308	491	326	491
ECOLI-YPEST-CO92	1789	463	X	X	X	X
ECOLI-YPEST-KIM	1798	464	X	X	X	X
HAEIN-PAERU	337	567	309	550	334	550
HAEIN-PMULT	849	509	794	497	843	497
HAEIN-SALTY	553	635	515	612	547	612
HAEIN-WGLOS	165	271	163	267	165	267
HAEIN-XAXON	243	482	221	473	236	473
HAEIN-XCAMP	239	485	218	473	232	473
HAEIN-XFAST	207	440	194	424	205	424
HAEIN-YPEST-CO92	523	617	481	597	519	597
HAEIN-YPEST-KIM	518	622	479	598	514	598
PAERU-PMULT	380	627	347	592	377	592
PAERU-SALTY	658	918	585	862	642	862
PAERU-WGLOS	253	249	249	248	253	248
PAERU-XAXON	620	863	562	802	607	802
PAERU-XCAMP	609	847	557	801	598	801
PAERU-XFAST	410	522	389	499	408	499
PAERU-YPEST-CO92	685	853	604	790	666	790
PAERU-YPEST-KIM	681	855	598	786	X	X
PMULT-SALTY	677	650	620	622	667	622
PMULT-WGLOS	197	265	190	262	197	262
PMULT-XAXON	277	506	248	495	269	495
PMULT-XCAMP	270	507	245	495	264	495
PMULT-XFAST	238	451	221	436	236	436
PMULT-YPEST-CO92	654	621	602	597	645	597
PMULT-YPEST-KIM	644	631	588	601	636	601
SALTY-WGLOS	383	185	376	181	383	181
SALTY-XAXON	445	718	391	684	433	684
SALTY-XCAMP	440	716	389	684	429	684
SALTY-XFAST	329	509	310	497	328	497
SALTY-YPEST-CO92	1793	483	X	X	X	X
SALTY-YPEST-KIM	1800	477	X	X	X	X
WGLOS-XAXON	195	266	189	261	195	261
WGLOS-XCAMP	195	266	189	260	195	260
WGLOS-XFAST	164	267	160	264	164	264
WGLOS-YPEST-CO92	380	186	373	182	380	182
WGLOS-YPEST-KIM	367	187	361	186	367	186
XAXON-XCAMP	X	X	X	X	X	X
XAXON-XFAST	1077	380	1026	375	1076	375
XAXON-YPEST-CO92	432	654	386	624	425	624
XAXON-YPEST-KIM	432	661	385	624	423	624
XCAMP-XFAST	1065	383	1005	373	1062	373
XCAMP-YPEST-CO92	424	644	380	620	421	620
XCAMP-YPEST-KIM	420	655	379	618	415	618
XFAST-YPEST-CO92	327	485	306	473	326	473
XFAST-YPEST-KIM	318	487	298	477	316	477
YPEST-CO92-YPEST-KIM	X	X	X	X	X	X

TABLE 3.12 – **Nombre d'adjacences** $\mathtt{adj}(\mathcal{C})$ **et nombre de points de cassure** $\mathtt{pdc}(\mathcal{C})$ **pour les problèmes** $\mathbf{opt_{IA}}$**,** $\mathbf{opt_{IP}}$ **et** $\mathbf{opt_{IPA}}$**.** $\mathcal{C}$ représente toujours la solution retournée et X représente les cas non résolus.

$G_1 - G_2$	Exemplaire				Intermédiaire				Maximum			
	adj($\mathcal{C}$)		pdc($\mathcal{C}$)		adj($\mathcal{C}$)		pdc($\mathcal{C}$)		adj($\mathcal{C}$)		pdc($\mathcal{C}$)	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
BAPHI-ECOLI	276	368	152	244	272	378	152	261	272	377	156	261
BAPHI-HAEIN	135	157	265	287	135	162	265	296	135	161	270	296
BAPHI-PAERU	166	226	232	292	166	230	232	303	166	229	240	303
BAPHI-PMULT	158	182	254	278	158	189	254	289	158	188	259	289
BAPHI-SALTY	273	367	154	248	269	377	154	265	269	376	158	265
BAPHI-WGLOS	187	201	168	182	187	203	168	186	187	203	170	186
BAPHI-XAXON	152	183	222	253	152	188	222	262	152	188	226	262
BAPHI-XCAMP	155	183	222	250	155	188	222	259	155	188	226	259
BAPHI-XFAST	140	158	231	249	140	162	231	258	140	162	236	258
BAPHI-YPEST-CO92	255	352	166	263	252	362	166	279	252	361	170	279
BAPHI-YPEST-KIM	251	339	172	260	248	349	172	276	248	348	176	276
ECOLI-HAEIN	338	489	610	761	337	551	610	878	337	550	665	878
ECOLI-PAERU	334	570	847	1083	333	668	847	1400	333	651	1082	1400
ECOLI-PMULT	414	593	622	801	411	670	622	953	412	662	703	953
ECOLI-SALTY	1592	2465	153	1026	1518	X	X	1631	1521	2874	277	1630
ECOLI-WGLOS	279	371	183	275	274	382	183	298	275	380	192	297
ECOLI-XAXON	250	380	675	805	249	437	675	1018	249	425	842	1018
ECOLI-XCAMP	251	378	678	805	250	432	678	1015	250	420	845	1015
ECOLI-XFAST	215	301	491	577	214	329	491	674	214	324	564	674
ECOLI-YPEST-CO92	1042	1559	426	943	1012	1789	X	1325	1015	1744	596	1325
ECOLI-YPEST-KIM	1037	1560	422	945	1008	1798	X	1343	1011	1747	607	1343
HAEIN-PAERU	201	301	550	650	201	337	550	747	201	333	615	747
HAEIN-PMULT	614	755	497	638	606	849	497	768	606	849	525	768
HAEIN-SALTY	325	492	612	779	325	553	612	901	325	550	676	901
HAEIN-WGLOS	138	159	267	288	138	165	267	304	138	165	277	304
HAEIN-XAXON	167	217	473	523	167	243	473	607	167	241	533	607
HAEIN-XCAMP	168	216	473	521	168	239	473	600	168	238	530	600
HAEIN-XFAST	155	191	424	460	154	207	424	519	154	205	468	519
HAEIN-YPEST-CO92	342	465	597	720	339	523	597	832	339	522	649	832
HAEIN-YPEST-KIM	336	460	598	722	334	518	598	836	334	517	653	836
PAERU-PMULT	235	340	592	697	235	380	592	819	235	373	681	819
PAERU-SALTY	324	559	862	1097	322	658	862	1413	322	644	1091	1413
PAERU-WGLOS	185	246	248	309	185	253	248	326	185	251	259	325
PAERU-XAXON	346	536	802	992	340	620	802	1285	341	609	1016	1284
PAERU-XCAMP	347	536	801	990	342	609	801	1266	343	596	1012	1265
PAERU-XFAST	257	372	499	614	253	410	499	724	254	405	572	723
PAERU-YPEST-CO92	353	571	790	1008	351	685	790	1310	351	671	990	1310
PAERU-YPEST-KIM	345	566	786	1007	344	681	786	1322	344	662	1004	1322
PMULT-SALTY	399	597	622	820	399	677	622	974	400	670	704	974
PMULT-WGLOS	169	188	262	281	169	197	262	298	169	197	270	298
PMULT-XAXON	192	245	495	548	192	277	495	639	192	274	557	639
PMULT-XCAMP	189	241	495	547	189	270	495	633	189	267	555	633
PMULT-XFAST	176	213	436	473	176	238	436	539	176	234	481	539
PMULT-YPEST-CO92	428	582	597	751	424	654	597	895	424	648	671	895
PMULT-YPEST-KIM	416	574	601	759	414	644	601	901	414	639	676	901
SALTY-WGLOS	272	372	181	281	267	383	181	304	268	381	190	303
SALTY-XAXON	248	376	684	812	247	445	684	1041	247	434	854	1041
SALTY-XCAMP	248	375	684	811	247	440	684	1034	247	427	854	1034
SALTY-XFAST	208	300	497	589	207	329	497	687	207	325	569	687
SALTY-YPEST-CO92	1034	1560	439	965	996	1793	X	1351	998	1758	591	1351
SALTY-YPEST-KIM	1032	1562	439	969	996	1800	X	1369	998	1761	606	1369
WGLOS-XAXON	154	189	261	296	153	195	261	310	153	194	269	310
WGLOS-XCAMP	156	189	260	293	156	195	260	306	156	194	268	306
WGLOS-XFAST	135	158	264	287	135	164	264	300	135	163	272	300
WGLOS-YPEST-CO92	268	369	182	283	266	380	182	303	267	377	192	302
WGLOS-YPEST-KIM	262	356	186	280	260	367	186	300	261	364	196	299
XAXON-XCAMP	2233	2880	114	761	2115	X	X	1320	2118	3256	181	1319
XAXON-XFAST	774	980	375	581	755	1077	375	721	755	1075	400	720
XAXON-YPEST-CO92	249	372	624	747	248	432	624	932	248	420	760	932
XAXON-YPEST-KIM	245	368	624	747	244	432	624	938	244	422	760	938
XCAMP-XFAST	764	969	373	578	745	1065	373	720	745	1060	404	719
XCAMP-YPEST-CO92	249	369	620	740	248	424	620	919	248	412	755	919
XCAMP-YPEST-KIM	244	365	618	739	243	420	618	918	243	412	749	918
XFAST-YPEST-CO92	219	298	473	552	217	327	473	648	217	323	542	648
XFAST-YPEST-KIM	213	292	477	556	211	318	477	649	211	315	545	649
YPEST-CO92-YPEST-KIM	2104	X	X	743	1976	X	X	1401	1986	3327	59	1400

TABLE 3.13 – **Nombres *minimum* et *maximum* de points de cassure** `pdc` **et d'adjacences** `adj` pour les modèles exemplaire, intermédiaire et maximum.

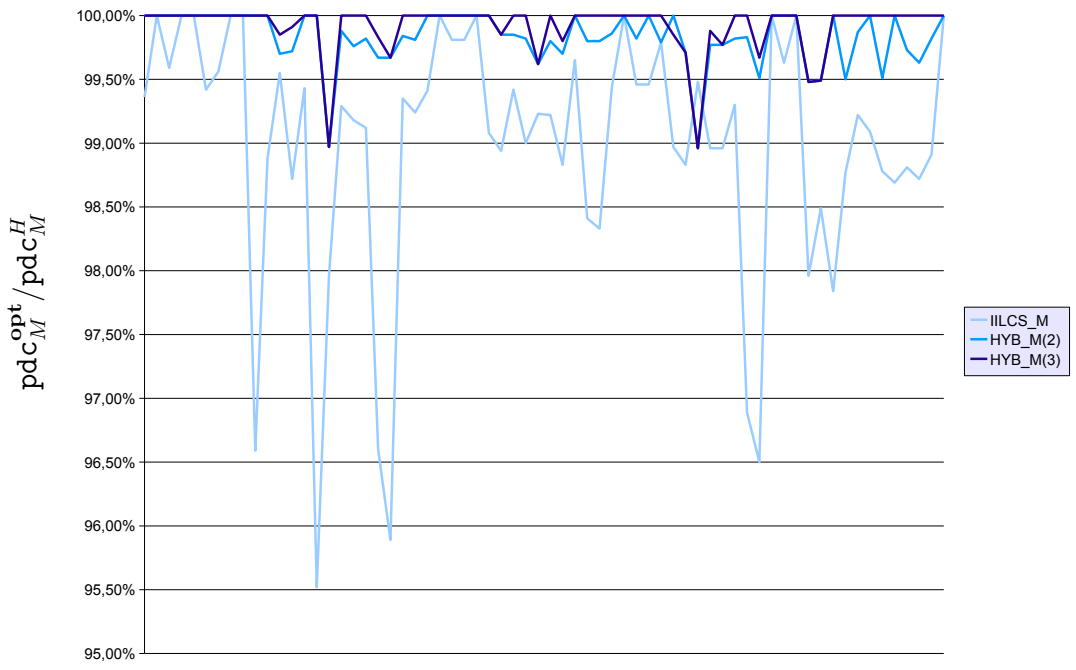
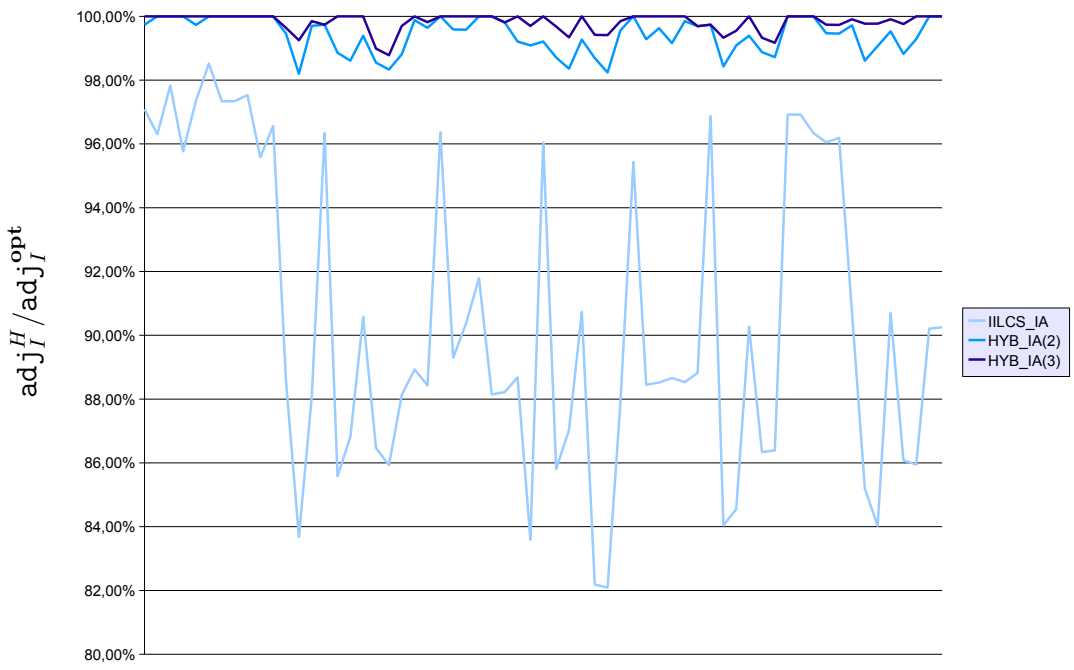


FIGURE 3.8 – **Représentation graphique du rapport $\mathtt{pdc}_M^{\mathbf{opt}}/\mathtt{pdc}_M^H$ pour le modèle maximum**, pour chacun des 66 résultats exacts que nous avons obtenu. H est l'heuristique étudiée (c'est-à-dire soit `IILCS_M`, soit `HYB_M(2)` soit `HYB_M(3)`), $\mathtt{pdc}_M^H$ est le nombre de points de cassure calculé par H et $\mathtt{pdc}_M^{\mathbf{opt}}$ est le nombre minimum de points de cassure pour le modèle maximum.

	OPT_M		IILCS_M		HYB_M(2)		HYB_M(3)	
$G_1 - G_2$	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)
BAPHI-ECOLI	377	156	376	157	377	156	377	156
BAPHI-HAEIN	161	270	161	270	161	270	161	270
BAPHI-PAERU	229	240	228	241	229	240	229	240
BAPHI-PMULT	188	259	188	259	188	259	188	259
BAPHI-SALTY	376	158	376	158	376	158	376	158
BAPHI-WGLOS	203	170	202	171	203	170	203	170
BAPHI-XAXON	188	226	187	227	188	226	188	226
BAPHI-XCAMP	188	226	188	226	188	226	188	226
BAPHI-XFAST	162	236	162	236	162	236	162	236
BAPHI-YPEST-CO92	361	170	355	176	361	170	361	170
BAPHI-YPEST-KIM	348	176	346	178	348	176	348	176
ECOLI-HAEIN	550	665	547	668	548	667	549	666
ECOLI-PAERU	651	1082	637	1096	648	1085	650	1083
ECOLI-PMULT	662	703	658	707	662	703	662	703
ECOLI-SALTY	2874	277	2861	290	2874	277	2874	277
ECOLI-WGLOS	380	192	376	196	378	194	378	194
ECOLI-XAXON	425	842	419	848	424	843	425	842
ECOLI-XCAMP	420	845	413	852	418	847	420	845
ECOLI-XFAST	324	564	319	569	323	565	324	564
ECOLI-YPEST-CO92	1744	596	1723	617	1742	598	1743	597
ECOLI-YPEST-KIM	1747	607	1721	633	1745	609	1745	609
HAEIN-PAERU	333	615	329	619	332	616	333	615
HAEIN-PMULT	849	525	845	529	848	526	849	525
HAEIN-SALTY	550	676	546	680	550	676	550	676
HAEIN-WGLOS	165	277	165	277	165	277	165	277
HAEIN-XAXON	241	533	240	534	241	533	241	533
HAEIN-XCAMP	238	530	237	531	238	530	238	530
HAEIN-XFAST	205	468	205	468	205	468	205	468
HAEIN-YPEST-CO92	522	649	516	655	522	649	522	649
HAEIN-YPEST-KIM	517	653	510	660	516	654	516	654
PAERU-PMULT	373	681	369	685	372	682	373	681
PAERU-SALTY	644	1091	633	1102	642	1093	644	1091
PAERU-WGLOS	251	259	249	261	250	260	250	260
PAERU-XAXON	609	1016	601	1024	607	1018	609	1016
PAERU-XCAMP	596	1012	584	1024	593	1015	594	1014
PAERU-XFAST	405	572	403	574	405	572	405	572
PAERU-YPEST-CO92	671	990	655	1006	669	992	671	990
PAERU-YPEST-KIM	662	1004	645	1021	660	1006	662	1004
PMULT-SALTY	670	704	666	708	669	705	670	704
PMULT-WGLOS	197	270	197	270	197	270	197	270
PMULT-XAXON	274	557	271	560	273	558	274	557
PMULT-XCAMP	267	555	264	558	267	555	267	555
PMULT-XFAST	234	481	233	482	233	482	234	481
PMULT-YPEST-CO92	648	671	641	678	648	671	647	672
PMULT-YPEST-KIM	639	676	631	684	637	678	637	678
SALTY-WGLOS	381	190	380	191	379	192	379	192
SALTY-XAXON	434	854	425	863	432	856	433	855
SALTY-XCAMP	427	854	418	863	425	856	425	856
SALTY-XFAST	325	569	321	573	324	570	325	569
SALTY-YPEST-CO92	1758	591	1739	610	1757	592	1758	591
SALTY-YPEST-KIM	1761	606	1739	628	1758	609	1759	608
WGLOS-XAXON	194	269	194	269	194	269	194	269
WGLOS-XCAMP	194	268	193	269	194	268	194	268
WGLOS-XFAST	163	272	163	272	163	272	163	272
WGLOS-YPEST-CO92	377	192	373	196	376	193	376	193
WGLOS-YPEST-KIM	364	196	361	199	363	197	363	197
XAXON-XCAMP	3256	181	3252	185	3256	181	3256	181
XAXON-XFAST	1075	400	1070	405	1073	402	1075	400
XAXON-YPEST-CO92	420	760	414	766	419	761	420	760
XAXON-YPEST-KIM	422	760	415	767	422	760	422	760
XCAMP-XFAST	1060	404	1055	409	1058	406	1060	404
XCAMP-YPEST-CO92	412	755	402	765	412	755	412	755
XCAMP-YPEST-KIM	412	749	403	758	410	751	412	749
XFAST-YPEST-CO92	323	542	316	549	321	544	323	542
XFAST-YPEST-KIM	315	545	309	551	314	546	315	545
YPEST-CO92-YPEST-KIM	3327	59	3327	59	3327	59	3327	59

TABLE 3.14 – **Le nombre d'adjacences** $\mathtt{adj}(\mathcal{C})$ **et le nombre de points de cassure** $\mathtt{pdc}(\mathcal{C})$ **pour le modèle maximum** : les résultats exacts et les résultats obtenus par les heuristiques IILCS_M, HYB_M(2) et HYB_M(3). $\mathcal{C}$ est un couplage maximum entre les génomes G_1 et G_2.

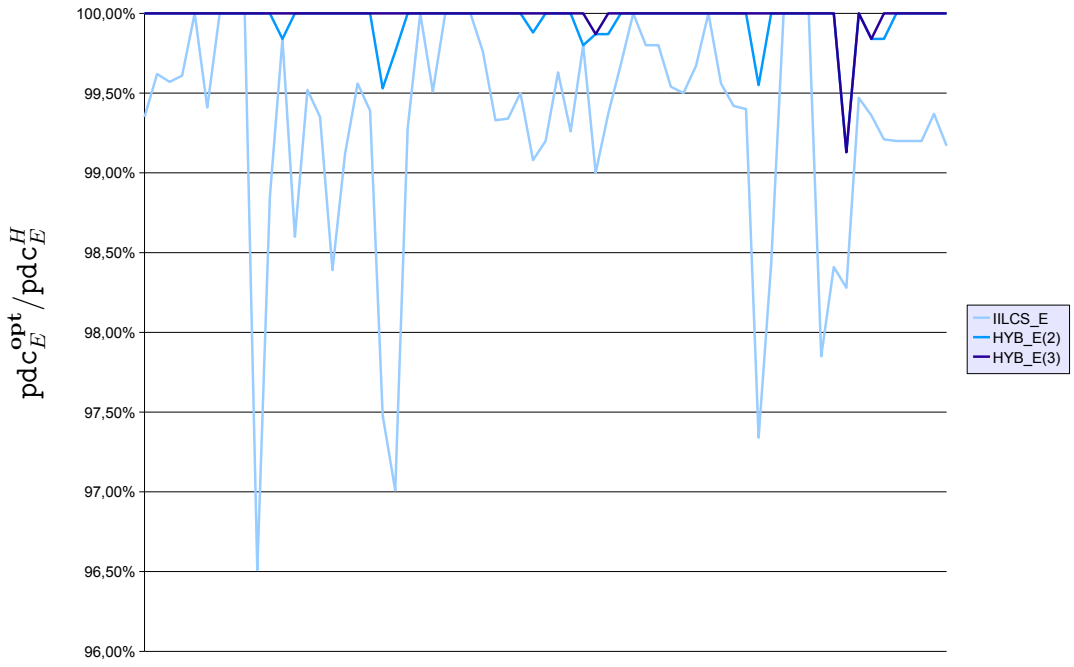


63 exécutions

FIGURE 3.9 – **Représentation graphique du rapport $\mathtt{adj}_I^H/\mathtt{adj}_I^{\mathbf{opt}}$ pour le modèle intermédiaire**, pour chacun des 63 résultats exacts que nous avons obtenu. H est l'heuristique étudiée (c'est-à-dire soit `IILCS_IA`, soit `HYB_IA(2)` soit `HYB_IA(3)`), $\mathtt{adj}_I^H$ est le nombre d'adjacences calculé par H et $\mathtt{adj}_I^{\mathbf{opt}}$ est le nombre maximum d'adjacences pour le modèle intermédiaire.

	OPT_{IA}		IILCS_IA		HYB_IA(2)		HYB_IA(3)	
$G_1 - G_2$	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)
BAPHI-ECOLI	378	154	367	153	377	156	378	154
BAPHI-HAEIN	162	267	156	266	162	268	162	266
BAPHI-PAERU	230	237	225	233	230	235	230	237
BAPHI-PMULT	189	256	181	255	189	257	189	255
BAPHI-SALTY	377	155	367	154	376	158	377	156
BAPHI-WGLOS	203	168	200	169	203	169	203	169
BAPHI-XAXON	188	223	183	222	188	225	188	224
BAPHI-XCAMP	188	223	183	222	188	225	188	223
BAPHI-XFAST	162	234	158	231	162	235	162	234
BAPHI-YPEST-CO92	362	167	346	172	362	168	362	168
BAPHI-YPEST-KIM	349	172	337	174	349	173	349	174
ECOLI-HAEIN	551	624	488	611	548	644	549	645
ECOLI-PAERU	668	906	559	858	657	911	663	913
ECOLI-PMULT	670	647	590	625	668	664	669	651
ECOLI-SALTY	X	X	2464	154	2882	247	2889	240
ECOLI-WGLOS	382	187	368	186	380	191	381	190
ECOLI-XAXON	437	718	374	681	434	723	437	715
ECOLI-XCAMP	432	717	375	681	428	720	432	724
ECOLI-XFAST	329	503	298	494	328	511	329	508
ECOLI-YPEST-CO92	1789	463	1547	438	1764	518	1771	506
ECOLI-YPEST-KIM	1798	464	1545	437	1769	523	1776	503
HAEIN-PAERU	337	567	297	554	332	575	336	571
HAEIN-PMULT	849	509	755	497	848	517	849	515
HAEIN-SALTY	553	635	489	615	551	648	552	649
HAEIN-WGLOS	165	271	159	267	165	273	165	272
HAEIN-XAXON	243	482	217	473	242	488	243	505
HAEIN-XCAMP	239	485	216	473	238	488	239	502
HAEIN-XFAST	207	440	190	425	207	436	207	438
HAEIN-YPEST-CO92	523	617	461	601	523	627	523	633
HAEIN-YPEST-KIM	518	622	457	601	517	628	517	620
PAERU-PMULT	380	627	337	595	377	620	380	620
PAERU-SALTY	658	918	550	871	652	931	656	922
PAERU-WGLOS	253	249	243	251	253	251	253	251
PAERU-XAXON	620	863	532	806	613	876	618	866
PAERU-XCAMP	609	847	530	807	602	864	605	861
PAERU-XFAST	410	522	372	499	408	524	410	515
PAERU-YPEST-CO92	685	853	563	798	677	854	681	857
PAERU-YPEST-KIM	681	855	559	793	668	850	677	846
PMULT-SALTY	677	650	595	624	675	665	676	664
PMULT-WGLOS	197	265	188	262	197	267	197	267
PMULT-XAXON	277	506	245	495	275	514	277	523
PMULT-XCAMP	270	507	239	497	268	511	270	531
PMULT-XFAST	238	451	211	438	236	447	238	453
PMULT-YPEST-CO92	654	621	579	600	653	635	654	646
PMULT-YPEST-KIM	644	631	572	603	643	641	642	630
SALTY-WGLOS	383	185	371	182	381	189	382	188
SALTY-XAXON	445	718	374	686	438	726	442	724
SALTY-XCAMP	440	716	372	687	438	724	438	715
SALTY-XFAST	329	509	297	500	328	516	329	511
SALTY-YPEST-CO92	1793	483	1548	451	1773	526	1781	512
SALTY-YPEST-KIM	1800	477	1555	446	1776	528	1785	515
WGLOS-XAXON	195	266	189	261	195	263	195	265
WGLOS-XCAMP	195	266	189	260	195	262	195	265
WGLOS-XFAST	164	267	158	264	164	267	164	266
WGLOS-YPEST-CO92	380	186	365	186	378	190	379	188
WGLOS-YPEST-KIM	367	187	353	189	365	194	366	192
XAXON-XCAMP	X	X	2877	117	3270	157	3271	154
XAXON-XFAST	1077	380	977	378	1074	394	1076	396
XAXON-YPEST-CO92	432	654	368	628	430	664	431	661
XAXON-YPEST-KIM	432	661	363	629	428	667	431	664
XCAMP-XFAST	1065	383	966	376	1063	393	1064	394
XCAMP-YPEST-CO92	424	644	365	624	419	657	423	651
XCAMP-YPEST-KIM	420	655	361	622	417	656	420	655
XFAST-YPEST-CO92	327	485	295	476	325	489	327	491
XFAST-YPEST-KIM	318	487	287	482	317	494	318	495
YPEST-CO92-YPEST-KIM	X	X	2815	32	3330	56	3330	56

TABLE 3.15 – **Le nombre d'adjacences** adj($\mathcal{C}$) **et le nombre de points de cassure** pdc($\mathcal{C}$) **pour le modèle intermédiaire** : les résultats exacts lorsque nous maximisons le nombre d'adjacences et les résultats obtenus par les heuristiques IILCS_IA, HYB_IA(2) et HYB_IA(3). $\mathcal{C}$ est un couplage entre les génomes G_1 et G_2 pour le modèle intermédiaire. X représente les cas non-résolus.



65 exécutions

FIGURE 3.10 – **Représentation graphique du rapport $\mathtt{pdc}_E^{\mathbf{opt}}/\mathtt{pdc}_E^H$ pour le modèle exemplaire**, pour chacun des 65 résultats exacts que nous avons obtenus. H est l'heuristique étudiée (c'est-à-dire soit `IILCS_E`, soit `HYB_E(2)` soit `HYB_E(3)`), $\mathtt{pdc}_E^H$ est le nombre de points de cassure calculé par H et $\mathtt{pdc}_E^{\mathbf{opt}}$ est le nombre minimum de points de cassure pour le modèle exemplaire.

	OPT_E		IILCS_E		HYB_E(2)		HYB_E(3)	
$G_1 - G_2$	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)
BAPHI-ECOLI	368	152	367	153	368	152	368	152
BAPHI-HAEIN	157	265	156	266	157	265	157	265
BAPHI-PAERU	226	232	225	233	226	232	226	232
BAPHI-PMULT	182	254	181	255	182	254	182	254
BAPHI-SALTY	367	154	367	154	367	154	367	154
BAPHI-WGLOS	201	168	201	168	201	168	201	168
BAPHI-XAXON	183	222	183	222	183	222	183	222
BAPHI-XCAMP	183	222	183	222	183	222	183	222
BAPHI-XFAST	158	231	158	231	158	231	158	231
BAPHI-YPEST-CO92	352	166	346	172	352	166	352	166
BAPHI-YPEST-KIM	339	172	337	174	339	172	339	172
ECOLI-HAEIN	489	610	488	611	489	610	489	610
ECOLI-PAERU	570	847	559	858	570	847	570	847
ECOLI-PMULT	593	622	590	625	593	622	593	622
ECOLI-SALTY	2465	153	2464	154	2465	153	2465	153
ECOLI-WGLOS	371	183	368	186	371	183	371	183
ECOLI-XAXON	380	675	373	682	379	676	380	675
ECOLI-XCAMP	378	678	374	682	378	678	378	678
ECOLI-XFAST	301	491	298	494	301	491	301	491
ECOLI-YPEST-CO92	1559	426	1545	440	1557	428	1559	426
ECOLI-YPEST-KIM	1560	422	1543	439	1559	423	1560	422
HAEIN-PAERU	301	550	297	554	301	550	301	550
HAEIN-PMULT	755	497	755	497	755	497	755	497
HAEIN-SALTY	492	612	489	615	492	612	492	612
HAEIN-WGLOS	159	267	159	267	159	267	159	267
HAEIN-XAXON	217	473	217	473	217	473	217	473
HAEIN-XCAMP	216	473	216	473	216	473	216	473
HAEIN-XFAST	191	424	190	425	191	424	191	424
HAEIN-YPEST-CO92	465	597	461	601	465	597	465	597
HAEIN-YPEST-KIM	460	598	457	601	459	599	460	598
PAERU-PMULT	340	592	338	594	340	592	340	592
PAERU-SALTY	559	862	552	869	558	863	559	862
PAERU-WGLOS	246	248	242	252	246	248	246	248
PAERU-XAXON	536	802	533	805	535	803	536	802
PAERU-XCAMP	536	801	531	806	536	801	536	801
PAERU-XFAST	372	499	372	499	372	499	372	499
PAERU-YPEST-CO92	571	790	565	796	571	790	570	791
PAERU-YPEST-KIM	566	786	561	791	565	787	566	786
PMULT-SALTY	597	622	595	624	597	622	597	622
PMULT-WGLOS	188	262	188	262	188	262	188	262
PMULT-XAXON	245	495	245	495	245	495	245	495
PMULT-XCAMP	241	495	240	496	241	495	241	495
PMULT-XFAST	213	436	211	438	213	436	213	436
PMULT-YPEST-CO92	582	597	579	600	582	597	582	597
PMULT-YPEST-KIM	574	601	572	603	574	601	574	601
SALTY-WGLOS	372	181	371	182	372	181	372	181
SALTY-XAXON	376	684	373	687	376	684	376	684
SALTY-XCAMP	375	684	372	687	375	684	375	684
SALTY-XFAST	300	497	297	500	300	497	300	497
SALTY-YPEST-CO92	1560	439	1551	448	1558	441	1560	439
SALTY-YPEST-KIM	1562	439	1553	448	1561	440	1562	439
WGLOS-XAXON	189	261	189	261	189	261	189	261
WGLOS-XCAMP	189	260	189	260	189	260	189	260
WGLOS-XFAST	158	264	158	264	158	264	158	264
WGLOS-YPEST-CO92	369	182	365	186	369	182	369	182
WGLOS-YPEST-KIM	356	186	353	189	356	186	356	186
XAXON-XCAMP	2880	114	2878	116	2879	115	2879	115
XAXON-XFAST	980	375	978	377	980	375	980	375
XAXON-YPEST-CO92	372	624	367	629	371	625	371	625
XAXON-YPEST-KIM	368	624	364	628	367	625	368	624
XCAMP-XFAST	969	373	966	376	969	373	969	373
XCAMP-YPEST-CO92	369	620	364	625	369	620	369	620
XCAMP-YPEST-KIM	365	618	361	622	365	618	365	618
XFAST-YPEST-CO92	298	473	294	477	298	473	298	473
XFAST-YPEST-KIM	292	477	287	482	292	477	292	477
YPEST-CO92-YPEST-KIM	X	X	2815	32	2815	32	2815	32

TABLE 3.16 – **Le nombre d'adjacences** adj($\mathcal{C}$) **et le nombre de points de cassure** pdc($\mathcal{C}$) **pour le modèle exemplaire** : les résultats exacts et les résultats obtenus par les heuristiques IILCS_E, HYB_E(2) et HYB_E(3). $\mathcal{C}$ est un couplage exemplaire entre les génomes G_1 et G_2. X représente les cas non-résolus.

$G_1 - G_2$	OPT_M		$ALEA_M$	
	adj($\mathcal{C}$)	pdc($\mathcal{C}$)	adj($\mathcal{C}$)	pdc($\mathcal{C}$)
BAPHI-ECOLI	377	156	323	210
BAPHI-HAEIN	161	270	151	280
BAPHI-PAERU	229	240	184	285
BAPHI-PMULT	188	259	175	272
BAPHI-SALTY	376	158	314	220
BAPHI-WGLOS	203	170	193	180
BAPHI-XAXON	188	226	165	249
BAPHI-XCAMP	188	226	161	253
BAPHI-XFAST	162	236	148	250
BAPHI-YPEST-CO92	361	170	296	235
BAPHI-YPEST-KIM	348	176	294	230
ECOLI-HAEIN	550	665	425	790
ECOLI-PAERU	651	1082	416	1317
ECOLI-PMULT	662	703	488	877
ECOLI-SALTY	2874	277	1921	1230
ECOLI-WGLOS	380	192	313	259
ECOLI-XAXON	425	842	294	973
ECOLI-XCAMP	420	845	305	960
ECOLI-XFAST	324	564	255	633
ECOLI-YPEST-CO92	1744	596	1219	1121
ECOLI-YPEST-KIM	1747	607	1205	1149
HAEIN-PAERU	333	615	252	696
HAEIN-PMULT	849	525	706	668
HAEIN-SALTY	550	676	380	846
HAEIN-WGLOS	165	277	151	291
HAEIN-XAXON	241	533	179	595
HAEIN-XCAMP	238	530	190	578
HAEIN-XFAST	205	468	172	501
HAEIN-YPEST-CO92	522	649	398	773
HAEIN-YPEST-KIM	517	653	393	777
PAERU-PMULT	373	681	290	764
PAERU-SALTY	644	1091	416	1319
PAERU-WGLOS	251	259	219	291
PAERU-XAXON	609	1016	428	1197
PAERU-XCAMP	596	1012	409	1199
PAERU-XFAST	405	572	316	661
PAERU-YPEST-CO92	671	990	436	1225
PAERU-YPEST-KIM	662	1004	430	1236
PMULT-SALTY	670	704	474	900
PMULT-WGLOS	197	270	180	287
PMULT-XAXON	274	557	221	610
PMULT-XCAMP	267	555	214	608
PMULT-XFAST	234	481	203	512
PMULT-YPEST-CO92	648	671	488	831
PMULT-YPEST-KIM	639	676	503	812
SALTY-WGLOS	381	190	324	247
SALTY-XAXON	434	854	304	984
SALTY-XCAMP	427	854	310	971
SALTY-XFAST	325	569	245	649
SALTY-YPEST-CO92	1758	591	1245	1104
SALTY-YPEST-KIM	1761	606	1237	1130
WGLOS-XAXON	194	269	171	292
WGLOS-XCAMP	194	268	178	284
WGLOS-XFAST	163	272	145	290
WGLOS-YPEST-CO92	377	192	312	257
WGLOS-YPEST-KIM	364	196	302	258
XAXON-XCAMP	3256	181	2516	921
XAXON-XFAST	1075	400	877	598
XAXON-YPEST-CO92	420	760	305	875
XAXON-YPEST-KIM	422	760	293	889
XCAMP-XFAST	1060	404	861	603
XCAMP-YPEST-CO92	412	755	299	868
XCAMP-YPEST-KIM	412	749	292	869
XFAST-YPEST-CO92	323	542	249	616
XFAST-YPEST-KIM	315	545	247	613
YPEST-CO92-YPEST-KIM	3327	59	2334	1052

TABLE 3.17 – **Nombres de points de cassure** pdc($\mathcal{C}$) **et d'adjacences** adj($\mathcal{C}$) **du couplage optimal** $\mathcal{C}$ **obtenu par** opt_M **et par l'heuristique** ALEA_M.

Chapitre 4

Comparaison de génomes partiellement ordonnés

Sommaire

De plus en plus de projets de séquençage et d'annotation de génomes sont réalisés [29, 24, 49, 70]. Ils permettent de déterminer l'ordre des gènes d'un génome. Malheureusement, certaines des techniques employées souffrent d'incertitudes dans la détermination de l'ordre de gènes voisins. En outre, lorsque plusieurs études sont effectuées pour un même génome, des conflits peuvent survenir lors de la synthèse des résultats. Par conséquent,

nous obtenons parfois une séquence de gènes partiellement ordonnés (voir Section 1.4 du Chapitre 1).

Pour obtenir un ordre total à partir d'un ordre partiel, Sankoff et al. [78, 89] ont proposé la stratégie suivante : confronter l'ordre partiel obtenu avec l'ordre total d'un génome proche et connu. Plus précisément, cette méthode recherche une extension linéaire du génome partiellement ordonnée qui optimise un critère de comparaison entre cette extension et l'ordre des gènes d'un génome proche, connu et totalement ordonné. L'optimisation est soit la minimisation d'une distance de réarrangements ou d'une mesure de dissimilarité, soit la maximisation d'une mesure de similarité (voir Section 1.2 du Chapitre 1). Par ailleurs nous pouvons, de façon similaire, calculer deux extensions linéaires qui optimisent une distance/mesure, à partir de deux génomes partiellement ordonnés.

Dans ce chapitre, nous présentons une solution algorithmique pour calculer, avec cette stratégie, l'ordre total d'un génome partiellement ordonné. Contrairement au Chapitre 3, nous ne prenons pas en compte les duplications de gènes : nous supposons qu'il n'y a qu'un exemplaire de gène par famille multigénique, sur chaque génome. Notre méthode est néanmoins similaire : nous utilisons une modélisation par programmation linéaire à variables booléennes (voir Section 2.3 Chapitre 2). L'ensemble des travaux présentés dans cette section est le fruit d'une collaboration avec Sébastien Angibaud, Guillaume Fertin et Stéphane Vialette [7].

Nous considérons uniquement les mesures de similarité suivantes : le nombre de points de cassure, le nombre d'adjacences et le nombre d'intervalles communs. Ces mesures ont été choisies comme travail préliminaire car elles sont les plus simples à calculer. En absence de duplications, la minimisation du nombre de points de cassures est équivalente à la maximisation du nombre d'adjacences. Par simplification, dans la suite de ce chapitre nous parlerons uniquement de la maximisation du nombre d'adjacences, en notant que tous les résultats obtenus seront aussi valable pour le nombre de points de cassure.

Les notations utilisées dans ce chapitre sont définies dans la Section 1.4 du Chapitre 2. Nous rappelons ici uniquement la définition d'un ordre partiel et des trois problèmes que nous allons étudier.

Définition 4.1 *Un* ordre partiel *est un ensemble d'éléments $\mathcal{E}$ et une relation binaire $\preceq$ tel que pour tout x, y et z dans $\mathcal{E}$:*

1. $x \preceq x$
2. $x \preceq y$ *et* $y \preceq x$ *implique* $x = y$, *et*
3. $x \preceq y$ *et* $y \preceq z$ *implique* $x \preceq z$.

Ces trois relations sont des propriétés de réflexivité, d'antisymétrie et de transitivité, respectivement.

La Figure 4.1 (a) page 129 représente deux ordres partiels (sous forme de DAG).

Les trois problèmes étudiés dans ce chapitre prennent en entrée des ordres partiels et renvoient des extensions linéaires de ces ordres partiels qui maximise une mesure de similarité.

Le problème MICL-1OP prend en entrée un unique ordre partiel P_1 (sans duplication) et renvoie une extension linéaire T_1 telle que le nombre d'intervalles communs entre T_1 et Id soit maximal.

MICL-1OP

- **Entrée** : Un ordre partiel P_1.
- **Sortie** : Une extension linéaire T_1 de P_1 qui maximise le nombre d'intervalles communs entre T_1 et Id.

Le problème MAL-1OP est similaire au problème MICL-1OP hormis que la mesure qui est maximisée est le nombre d'adjacences.

MAL-1OP

- **Entrée** : Un ordre partiel P_1.
- **Sortie** : Une extension linéaire T_1 de P_1 qui maximise le nombre d'adjacences entre T_1 et Id.

Le problème MAL-2OP est un problème plus général : il prend en entrée deux ordres partiels P_1 et P_2 (sans duplication) et renvoie deux extensions linéaires T_1 et T_2 de P_1 et P_2, respectivement, telles que le nombre d'adjacences soit maximisé entre T_1 et T_2. Nous pouvons ainsi comparer deux génomes dont l'ordre des gènes n'est que partiellement connu. Dans le cas où P_2 (ou P_1) est un ordre totalement ordonné, le problème MAL-2OP est le problème MAL-1OP.

MAL-2OP

- **Entrée** : Deux ordres partiels P_1 et P_2.
- **Sortie** : Deux extensions linéaires T_1 et T_2 de P_1 et P_2, respectivement, qui maximisent le nombre d'adjacences entre T_1 et T_2.

La Figure 4.1 illustre ces trois problèmes.

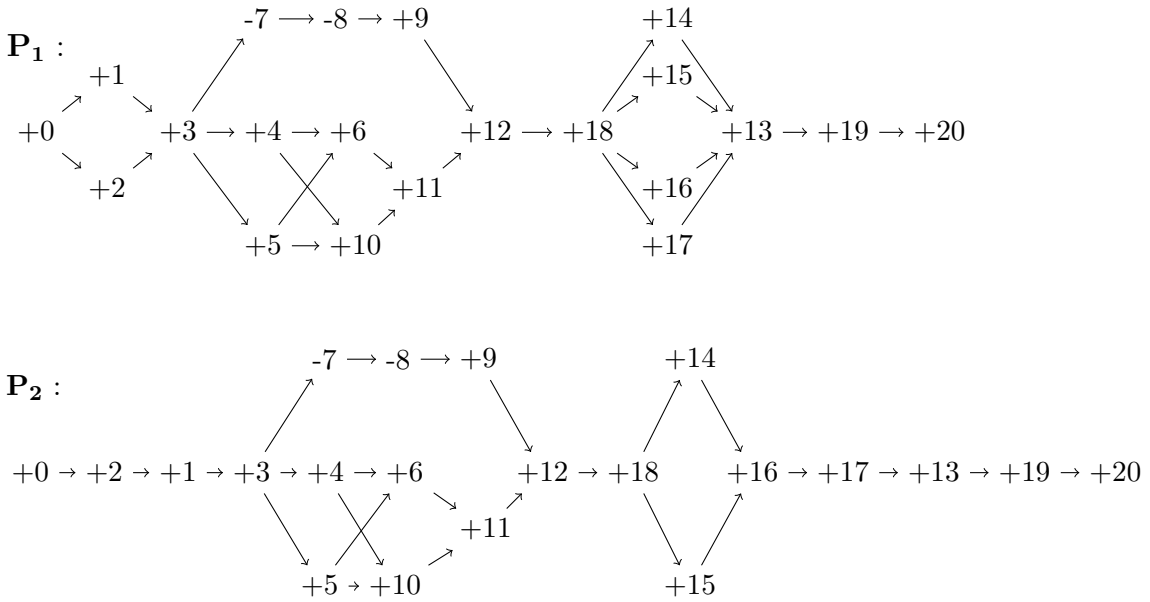
Nous notons dès à présent qu'afin de prendre en compte les intervalles communs et les adjacences aux extrémités des deux génomes, nous ajoutons artificiellement deux gènes non-dupliqués. En effet, pour tout ordre partiel P_x dont les gènes appartiennent à $[1 : n_x]$, le gène $+0$ est placé avant tous les gènes et le gène $+(n_x + 1)$ est placé après tous les gènes. De plus, sachant que nous allons comparer uniquement des ordres comprenant le même ensemble de gènes sans duplication, nous avons $n_1 = n_2$. Finalement, nous supposons par la suite que tous les ordres sont encadrés par les gènes $+0$ et $+n$, avec $n = n_x + 1$.

Ce chapitre est organisé de la façon suivante. Nous nous focalisons dans la Section 4.1 sur la résolution des problèmes MICL-1OP, MAL-1OP et MAL-2OP. Pour cela, nous proposons respectivement les programmes linéaires à variables booléennes `IC-1OP`, `Adjacence-1OP` et `Adjacence-2OP`. Ces 3 programmes prennent en entrée des ordres partiels : P_1 pour `IC-1OP` et `Adjacence-1OP` ; P_1 et P_2 pour `Adjacence-2OP`.

Dans la Section 4.1.1, nous commençons par présenter les similarités entre les trois programmes : définir une (ou deux pour `Adjacence-2OP`) extension(s) linéaire(s) pour un(deux) ordre(s) partiel(s). Nous montrons ensuite les spécificités de chaque programme : maximiser le nombre d'intervalles communs ou le nombre d'adjacences entre un ordre partiel et l'idendité ou entre deux ordres partiels. Dans la Section 4.1.2, nous donnons des règles de réduction de données pour réduire la taille des instances et ainsi accélérer le programme. Finalement, nous montrons comment adapter ces programmes pour les génomes dont le signe des gènes n'est pas connu (Sous-section 4.1.3).

La Section 4.2 est consacrée aux résultats expérimentaux sur des données non-signées et simulées par Blin et al. [15]. Les trois programmes seront étudiés à l'aide de ces données. De plus, nous évaluons les différentes règles de réduction que nous proposons et l'intérêt de `Adjacence-1OP` sur `Adjacence-2OP` pour le cas où P_2 est un ordre total. Finalement, nous comparons les deux mesures étudiées.

(a)



(b)

MICL-1PO renvoie T_1, avec 156 intervalles communs entre Id et T_1.
T_1 : +0 +1 +2 +3 +4 +5 +6 -7 -8 +9 +10 +11 +12 +18 +17 +16 +15 +14 +13 +19 +20

MAL-1PO renvoie T_1, avec 14 adjacences entre Id et T_1.
T_1 : +0 +1 +2 +3 +4 +5 +6 -7 -8 +9 +10 +11 +12 +18 +14 +15 +16 +17 +13 +19 +20

MAL-2PO renvoie T_1 et T_2, avec 20 adjacences entre T_1 et T_2.
T_1 : +0 +2 +1 +3 +5 +4 +10 -7 -8 +6 +11 +9 +12 +18 +14 +15 +16+ 17 +13 +19 +20
T_2 : +0 +2 +1 +3 +5 +4 +10 -7 -8 +6 +11 +9 +12 +18 +14 +15 +16+ 17 +13 +19 +20

FIGURE 4.1 – **(a)** Exemple d'instances pour les problèmes MICL-1OP, MAL-1OP et MAL-2OP : les ordres partiels P_1 et P_2 représenté par des DAG. **(b)** Résultats possibles pour chacun des trois problèmes.

4.1 Une approche exacte par programmation linéaire à variables booléennes

Nous développons dans cette section une approche générique exacte. L'idée principale est de modéliser nos problèmes en des programmes linéaires à variables booléennes [79] et d'utiliser des solveurs puissants pour obtenir des solutions optimales. Dans cette étude, tous les résultats ont été obtenus par le solveur `MiniSat+` [39].

4.1.1 Programmes linéaires à variables booléennes

Similarités entre les trois programmes

Soient P_1 et P_2 deux ordres partiels. Pour tout x appartenant à $\{1; 2\}$, nous cherchons une extension linéaire T_x de P_x résolvant MICL-1OP, MAL-1OP ou MAL-2OP. Pour les problèmes MICL-1OP et MAL-1OP, P_2 est l'ordre total Id. L'extension linéaire T_2 de P_2 est alors égale à P_2.

Les programmes sont divisés en deux parties : définir des extensions linéaires et maximiser une mesure. Pour la seconde partie, nous utilisons des variables et des contraintes spécifiques pour chaque programme. Mais pour définir une extension linéaire, nous utilisons un même ensemble de variables booléennes ($\mathcal{A}^x$) et de contraintes (**C.a**, **C.b** et **C.c**).

Pour définir un ordre total T_x qui soit une extension linéaire de P_x, nous utilisons l'ensemble de variables booléennes $\mathcal{A}^x = \{a^x_{\mathtt{g},i} \ : \ \mathtt{g} \in [0 : n] \text{ et } i \in [0 : n]\}$. Une variable $a^x_{\mathtt{g},i}$, appartenant à $\mathcal{A}^x$, est égale à 1 si et seulement si $T_x(\mathtt{g})$ est égale à i. Les contraintes de **C.a** à **C.c** (voir Figure 4.2) imposent que T_x soit une extension linéaire de P_x. Plus précisément et pour tous les programmes, nous avons les variables et les contraintes définies ci-dessous.

Variables :

- ⋆ Les variables $a^x_{\mathtt{g},i}$, $0 \leq \mathtt{g} \leq n$, $0 \leq i \leq n$, définissent l'ordre total T_x : $a^x_{\mathtt{g},i} = 1$ si et seulement si $T_x(\mathtt{g})$ est égale à i.

Contraintes :

- ⋆ La contrainte **(C.a)** garantit que chaque gène est assigné à exactement une position dans T_x ;
- ⋆ La contrainte **(C.b)** garantit que deux gènes ne sont pas assignés à la même position dans T_x ;
- ⋆ La contrainte **(C.c)** garantit que T_x est une extension linéaire de P_x : soient deux gènes $\mathtt{g}_1$ et $\mathtt{g}_2$ de P_x tel que $\mathtt{g}_1 \prec_x \mathtt{g}_2$, alors pour tout i et j, $0 \leq j < i \leq n$, la somme des variables $a^x_{\mathtt{g}_1,i}$ et $a^x_{\mathtt{g}_2,j}$ doit être différente de 2.

Définir une extension linéaire de l'ordre partiel P_x ($x \in \{1;2\}$)

Contraintes :

`C.a` $\forall\, 0 \leq \mathtt{g} \leq n, \sum_{0 \leq i \leq n} a^x_{\mathtt{g},i} = 1$

`C.b` $\forall\, 0 \leq i \leq n, \sum_{0 \leq \mathtt{g} \leq n} a^x_{\mathtt{g},i} = 1$

`C.c` $\forall\, 0 \leq \mathtt{g}_1 \leq n,\ 0 \leq \mathtt{g}_2 \leq n,\ \mathtt{g}_1 \prec_x \mathtt{g}_2,\ 0 < j \leq i \leq n,\ a^x_{\mathtt{g}_1,i} + a^x_{\mathtt{g}_2,j} \leq 1$

FIGURE 4.2 – Pour les 3 programmes (`IC-1OP`, `Adjacence-1OP`, `Adjacence-2OP`) les contraintes de `C.a` à `C.c` garantissent que T^G soit une extension linéaire de P_x ; $x \in \{1;2\}$.

Nous allons maintenant étudier les spécificités de chaque problème en commençant par `MICL-1OP`.

Maximiser le nombre d'intervalles communs entre un ordre partiel et l'identité

Le programme linéaire à variables booléennes `IC-1OP` que nous proposons ici calcule une extension linéaire T_1 de l'ordre partiel P_1 tel que le nombre d'intervalles communs entre T_1 et Id soit maximisé.
Nous rappelons que le signe des gènes n'est pas pertinent pour la mesure d'intervalles communs. Le programme `IC-1OP` est présenté dans la Figure 4.3 et est illustré par la Figure 4.4.

Soient $\mathtt{g}$, i et t trois indices dans $[0 : n]$. Après avoir défini l'ordre total T_1 avec l'ensemble des variables $\mathcal{A}^1$, nous définissons deux ensembles de variables booléennes : $\mathcal{B} = \{b_{\mathtt{g},i,t} \ :\ \mathtt{g} \in [0 : n],\ i \in [0 : n] \text{ et } t \in [0 : n]\}$ et $\mathcal{C} = \{c_{\mathtt{g},i,t} \ :\ \mathtt{g} \in [0 : n],\ i \in [0 : n] \text{ et } t \in [0 : n]\}$. Une variable $b_{\mathtt{g},i,t}$, appartenant à $\mathcal{B}$, est égale à 1 si et seulement si $T_1(\mathtt{g})$ est dans $[i : i+t]$. Une variable $c_{\mathtt{g},i,t}$, appartenant à $\mathcal{C}$, est égale à 1 si et seulement si, pour tout k dans $[0 : t]$, $T_1(\mathtt{g}+k)$ est dans $[i : i+t]$; c'est-à-dire l'ensemble des gènes $\mathtt{g}+k$ (k dans $[0 : t]$) est un intervalle de T_1. Dans ce cas, nous avons un intervalle commun entre T_1 et l'identité.

L'intervalle composé des gènes $\{\mathtt{g}; \mathtt{g}+1; \ldots; \mathtt{g}+t\}$ est commun à T_1 et à l'identité si et seulement si $\sum_{0 \leq \mathtt{g} \leq n} c_{\mathtt{g},i,t} = 1$. Sachant que $\sum_{0 \leq \mathtt{g} \leq n} c_{\mathtt{g},i,t} \leq 1$, en maximisant la somme des variables $c_{\mathtt{g},i,t}$ nous obtenons le nombre maximum d'intervalles communs entre T_1 et l'identité.

Les variables, la fonction objective et les contraintes impliquées sont maintenant présentées.

Variables :

- ⋆ Les variables $a^1_{\mathsf{g},i}$, $0 \leq \mathsf{g} \leq n$, $0 \leq i \leq n$;
- ⋆ Les variables $b_{\mathsf{g},i,t}$, $0 \leq \mathsf{g} \leq n$, $0 \leq i \leq n$, $0 \leq t \leq n-i$, indiquent si $T_1(\mathsf{g})$ est dans $[i : i+t]$: $b_{\mathsf{g},i,t} = 1$ si et seulement s'il existe j dans $[0 : t]$ telle que $T_1(\mathsf{g})$ soit égale à $i+j$;
- ⋆ Les variables $c_{\mathsf{g},i,t}$, $0 \leq \mathsf{g} \leq n$, $0 \leq i \leq n$, $0 \leq t \leq n-i$, indiquent que l'intervalle $T_1[T_1[i], T_1[i+t]]$ est composé des gènes $\{\mathsf{g}, \ldots, \mathsf{g}+t\}$.

Objectif :

- ⋆ Maximiser $\sum_{0 \leq \mathsf{g} \leq n} \sum_{0 \leq i \leq n} \sum_{0 \leq t \leq n-i} c_{\mathsf{g},i,t}$

Contraintes :

- ⋆ Les contraintes de **(C.01)** à **(C.03)** sont équivalentes aux contraintes de **(C.a)** à **(C.c)** avec x égal à 1 ;
- ⋆ La contrainte **(C.04)** garantit la valeur des variables $b_{\mathsf{g},i,t}$: $b_{\mathsf{g},i,t} = 1$ si et seulement s'il existe j dans $[0 : t]$ tel que $a_{\mathsf{g},i+j}$ soit égale à 1 ;
- ⋆ La contrainte **(C.05)** garantit la valeur des variables $c_{\mathsf{g},i,t}$: $c_{\mathsf{g},i,t} = 1$ si et seulement si $b_{\mathsf{g}+k,i,t}$ est égale à 1 pour tout k dans $[0 : t]$.

Maximiser le nombre d'adjacences entre un ordre partiel et l'identité

❑ Le programme linéaire à variables booléennes `Adjacence-IOP` que nous proposons ici calcule une extension linéaire T_1 d'un génome partiellement ordonné P_1 tel que le nombre d'adjacences entre T_1 et Id soit maximisé.
À l'opposé des intervalles communs, la définition d'une adjacence dépend du signe des gènes.
Puisque nous comparons T_1 avec l'identité, deux gènes g_1 et g_2 consécutifs sur T_1 forment une adjacence si seulement si $\mathsf{g}_2 = \mathsf{g}_1 + 1$. Par exemple $\mathsf{g}_1 = +4$ et $\mathsf{g}_2 = +5$ ou $\mathsf{g}_1 = -5$ et $\mathsf{g}_2 = -4$.
Le programme est présenté dans la Figure 4.5 et est illustré par la Figure 4.6.

Soient g et i deux indices dans $[0 : n]$. Après avoir défini l'ordre total T_1 avec l'ensemble des variables $\mathcal{A}^1$, nous définissons l'ensemble des variables booléennes $\mathcal{D} = \{d_{\mathsf{g},i} \ : \ \mathsf{g} \in [0 : n[\text{ et } i \in [0 : n]\}$.

Une variable $d_{\mathsf{g},i}$ dans $\mathcal{D}$ est égale à 1 si et seulement si

– $T_1(\mathsf{g})$ est égale à i, et

Programme `IC-1OP`

Objectif :

`Maximiser` $\sum_{0 \leq \mathtt{g} \leq n} \sum_{0 \leq i \leq n} \sum_{0 \leq t \leq n-i} c_{\mathtt{g},i,t}$

Contraintes :

`C.01` $\forall 0 \leq \mathtt{g} \leq n, \sum_{0 \leq i \leq n} a^1_{\mathtt{g},i} = 1$

`C.02` $\forall 0 \leq i \leq n, \sum_{0 \leq \mathtt{g} \leq n} a^1_{\mathtt{g},i} = 1$

`C.03` $\forall 0 \leq \mathtt{g}_1 \leq n,\ 0 \leq \mathtt{g}_2 \leq n,\ \mathtt{g}_1 \prec_1 \mathtt{g}_2,\ 0 < j \leq i \leq n, a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_2,j} < 1$

`C.04` $0 \leq \mathtt{g} \leq n,\ 0 \leq i \leq n,\ 0 \leq t \leq n-i, \sum_{0 \leq j \leq t} a^1_{\mathtt{g},i+j} - b_{\mathtt{g},i,t} = 0$

`C.05` $0 \leq \mathtt{g} \leq n,\ 0 \leq i \leq n,\ 0 \leq t \leq n-i,$
$\sum_{0 \leq k \leq t} b_{\mathtt{g}+k,i,t} - c_{\mathtt{g},i,t} < t+1$
et $\sum_{0 \leq k \leq t} b_{\mathtt{g}+k,i,t} - t.c_{\mathtt{g},i,t} \geq 0$

FIGURE 4.3 – Le programme `IC-1OP` renvoie le nombre maximum d'intervalles communs entre une extension linéaire d'un ordre partiel et l'identité.

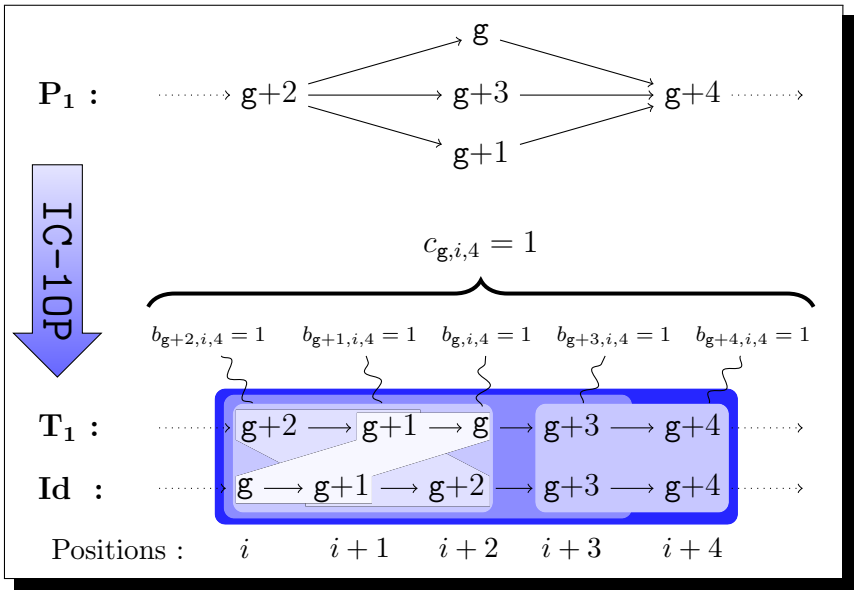


FIGURE 4.4 – **Illustration de** `IC-1OP`. Soit P_1 un ordre partiel. Les gènes aux positions comprises entre i et $i+4$ créent 6 intervalles communs non triviaux : $T_1[\mathtt{g}+2, \mathtt{g}+1]$, $T_1[\mathtt{g}+2, \mathtt{g}]$, $T_1[\mathtt{g}+2, \mathtt{g}+3]$, $T_1[\mathtt{g}+2, \mathtt{g}+4]$, $T_1[\mathtt{g}+1, \mathtt{g}]$ et $T_1[\mathtt{g}+3, \mathtt{g}+4]$. Donc les variables $c_{\mathtt{g}+1,i,1}$, $c_{\mathtt{g},i,2}$, $c_{\mathtt{g},i,3}$, $c_{\mathtt{g},i,4}$, $c_{\mathtt{g},i+1,1}$ et $c_{\mathtt{g}+3,i+3,1}$ sont égales à 1.

– nous avons une adjacence créée par g et $(\mathsf{g}+1)$.

Sachant qu'il n'existe pas de gène $n+1$, il est inutile de considérer une variable $d_{\mathsf{g},i}$ pour $\mathsf{g} = n$. Pour tout $\mathsf{g} \neq n$, g et $(\mathsf{g}+1)$ forment une adjacence si et seulement si $\sum_{0 \leq i \leq n} d_{\mathsf{g},i} = 1$. Sachant que $\sum_{0 \leq i \leq n} d_{\mathsf{g},i} \leq 1$, en maximisant la somme des variables $d_{\mathsf{g},i}$ nous obtenons le nombre maximum d'adjacences entre T_1 et l'identité.

Les variables, la fonction objective et les contraintes impliquées sont présentées ci-dessous.

Variables :

⋆ Les variables $a^1_{\mathsf{g},i}$, $0 \leq \mathsf{g} \leq n$, $0 \leq i \leq n$;
⋆ Les variables $d_{\mathsf{g},i}$, $0 \leq \mathsf{g} < n$, $0 \leq i \leq n$, définissent une adjacence formée par $(\mathsf{g}, \mathsf{g}+1)$ avec $T_1(\mathsf{g})$ égale à i : $d_{\mathsf{g},i} = 1$ si et seulement si $T_1(\mathsf{g})$ est égale à i et $T_1(\mathsf{g}+1)$ est égale à
– $i+1$ si $s_1(\mathsf{g}) = \text{“}+\text{”}$, $s_1(\mathsf{g}+1) = \text{“}+\text{”}$ et $i \neq n$,
– $i-1$ si $s_1(\mathsf{g}) = \text{“}-\text{”}$, $s_1(\mathsf{g}+1) = \text{“}-\text{”}$ et $i \neq 0$.

Objectif :

⋆ Maximiser $\sum\limits_{0 \leq \mathsf{g} < n} \sum\limits_{0 \leq i \leq n} d_{\mathsf{g},i}$

Contraintes :

⋆ Les contraintes de **(C'.01)** à **(C'.03)** sont équivalentes aux contraintes de **(C.a)** à **(C.c)** avec $x = 1$;
⋆ La contrainte **(C'.04)** garantit que les variables $d_{\mathsf{g},i}$ $(0 \leq \mathsf{g} < n,\ 0 < i < n)$, sont égales à 1 si et seulement si $a^1_{\mathsf{g},i} = 1$ et
– $a^1_{\mathsf{g}+1,i+1} = 1$ si $s_1(\mathsf{g}) = \text{“}+\text{”}$ et $s_1(\mathsf{g}+1) = \text{“}+\text{”}$
– $a^1_{\mathsf{g}+1,i-1} = 1$ si $s_1(\mathsf{g}) = \text{“}+\text{”}$ et $s_1(\mathsf{g}+1) = \text{“}-\text{”}$;
⋆ La contrainte **(C'.05)** garantit que les variables $d_{\mathsf{g},0}$ sont égales à 1 si et seulement si $a^1_{\mathsf{g},0} = 1$, $a^1_{\mathsf{g}+1,1} = 1$, $s_1(\mathsf{g}) = \text{“}+\text{”}$ et $s_1(\mathsf{g}+1) = \text{“}+\text{”}$;
⋆ La contrainte **(C'.06)** garantit que les variables $d_{\mathsf{g},n}$ sont égales à 1 si et seulement si $a^1_{\mathsf{g},n} = 1$, $a^1_{\mathsf{g}+1,n-1} = 1$, $s_1(\mathsf{g}) = \text{“}-\text{”}$ et $s_1(\mathsf{g}+1) = \text{“}-\text{”}$.

Programme `Adjacence-1OP`

Objectif :

`Maximiser` $\sum_{0 \leq \mathtt{g} < n} \sum_{0 \leq i \leq n} d_{\mathtt{g},i}$

Contraintes :

`C'.01` $\forall\, 0 \leq \mathtt{g} \leq n,\ \sum_{0 \leq i \leq n} a^1_{\mathtt{g},i} = 1$

`C'.02` $\forall\, 0 \leq i \leq n,\ \sum_{0 \leq \mathtt{g} \leq n} a^1_{\mathtt{g},i} = 1$

`C'.03` $\forall\, 0 \leq \mathtt{g}_1 \leq n,\ 0 \leq \mathtt{g}_2 \leq n,\ \mathtt{g}_1 \prec_1 \mathtt{g}_2,\ 0 < j \leq i \leq n,\ a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_2,j} \leq 1$

`C'.04` $\forall\, 0 \leq \mathtt{g} < n, 0 < i < n,$
si $s_1(\mathtt{g}) =$ “ $+$ ” et $s_1(\mathtt{g}+1) =$ “ $+$ ” alors $a^1_{\mathtt{g},i} + a^1_{\mathtt{g}+1,i+1} - d_{\mathtt{g},i} \leq 1$
et $3.d_{\mathtt{g},i} - a^1_{\mathtt{g},i} - a^1_{\mathtt{g}+1,i+1} \leq 1$
si $s_1(\mathtt{g}) =$ “ $-$ ” et $s_1(\mathtt{g}+1) =$ “ $-$ ” alors $a^1_{\mathtt{g},i} + a^1_{\mathtt{g}+1,i-1} - d_{\mathtt{g},i} \leq 1$
et $3.d_{\mathtt{g},i} - a^1_{\mathtt{g},i} - a^1_{\mathtt{g}+1,i+1} \leq 1$
sinon $d_{\mathtt{g},i} = 0$

`C'.05` $\forall\, 0 \leq \mathtt{g} < n,$
si $s_1(\mathtt{g}) =$ “ $+$ ” et $s_1(\mathtt{g}+1) =$ “ $+$ ” alors $a^1_{\mathtt{g},0} + a^1_{\mathtt{g}+1,1} - d_{\mathtt{g},0} \leq 1$
et $3.d_{\mathtt{g},0} - a^1_{\mathtt{g},0} - a^1_{\mathtt{g}+1,1} \leq 1$
sinon $d_{\mathtt{g},0} = 0$

`C'.06` $\forall\, 0 \leq \mathtt{g} < n,$
si $s_1(\mathtt{g}) =$ “ $-$ ” et $s_1(\mathtt{g}+1) =$ “ $-$ ” alors $a^1_{\mathtt{g},n} + a^1_{\mathtt{g}+1,n-1} - d_{\mathtt{g},n} \leq 1$
et $3.d_{\mathtt{g},n} - a^1_{\mathtt{g},n} - a^1_{\mathtt{g}+1,n-1} \leq 1$
sinon $d_{\mathtt{g},n} = 0$

FIGURE 4.5 – Le programme `Adjacence-1OP` renvoie le nombre maximum d'adjacences entre une extension linéaire d'un ordre partiel et l'identité.

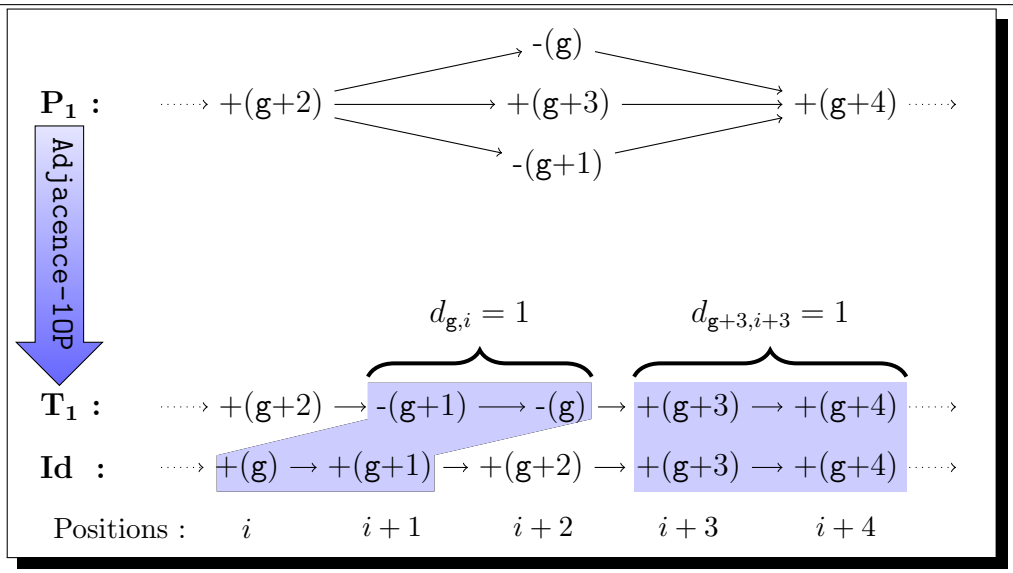


FIGURE 4.6 – **Illustration de `Adjacence-1OP`.** Soit P_1 un ordre partiel. Les gènes ayant une position comprise entre i et $i+4$ forment 2 adjacences : les paires de gènes $(\mathtt{g}, \mathtt{g}+1)$ et $(\mathtt{g}+3, \mathtt{g}+4)$. Les variables $d_{\mathtt{g},i}$ et $d_{\mathtt{g}+3,i+3}$ sont donc égales à 1.

Maximiser le nombre d'adjacences entre deux ordres partiels

Le programme linéaire à variables booléennes `Adjacence-2OP` que nous proposons ici calcule deux extensions linéaires T_1 et T_2 de deux génomes partiellement ordonnées P_1 et P_2, respectivement, tel que le nombre d'adjacences entre T_1 et T_2 soit maximisé.
Tout comme pour le problème `MAL-1OP`, les signes des gènes sont significatifs. Mais, contrairement à `MAL-1OP`, nous ne connaissons pas à l'avance la relation entre deux gènes formant une adjacence ce qui rend le problème plus complexe. Le programme est présenté Figure 4.7 et est illustré par la Figure 4.8.

Soient $\mathtt{g}_1$, i, j et $\mathtt{g}_2$ quatre indices dans $[0:n]$. Après avoir défini les ordres totaux T_1 et T_2 avec l'ensemble des variables $\mathcal{A}^1$ et $\mathcal{A}^2$, nous définissons les adjacences à l'aide de l'ensemble des variables booléennes $\mathcal{E} = \{e_{\mathtt{g}_1,i,j,\mathtt{g}_2} \ : \ \mathtt{g}_1 \in [0:n], i \in [0:n], j \in [0:n] \text{ et } \mathtt{g}_2 \in [0:n]\}$. Toutes les variables $e_{\mathtt{g}_1,i,j,\mathtt{g}_2}$ appartenant à $\mathcal{E}$ dépendent de quatre indices : deux gènes $\mathtt{g}_1$ et $\mathtt{g}_2$ et deux positions i et j. La variable $e_{\mathtt{g}_1,i,j,\mathtt{g}_2}$ est égale à 1 si et seulement si

- les gènes $\mathtt{g}_1$ et $\mathtt{g}_2$ forment une adjacence et
- $T_1(\mathtt{g}_1)$ est égale à i, $T_2(\mathtt{g}_1)$ est égale à j et $T_1(\mathtt{g}_2)$ est égale à $i+1$.

La sous-séquence de T_1 de deux gènes commençant par g_1, créée une adjacence si et seulement si $\sum_{0\leq i\leq n}\ \sum_{0\leq j\leq n}\ \sum_{0\leq \mathsf{g}_2\leq n} e_{\mathsf{g}_1,i,j,\mathsf{g}_2} = 1$. Sachant que $\sum_{0\leq \mathsf{g}_2\leq n} e_{\mathsf{g}_1,i,j,\mathsf{g}_2} \leq 1$, en maximisant la somme des variables $e_{\mathsf{g}_1,i,j,\mathsf{g}_2}$, nous obtenons le nombre maximum d'adjacences entre T_1 et T_2.

Les variables, la fonction objective et les contraintes impliquées sont maintenant présentées.

Variables :

- ⋆ Les variables $a^1_{\mathsf{g},i}$, $0 \leq \mathsf{g} \leq n$, $0 \leq i \leq n$
- ⋆ Les variables $a^2_{\mathsf{g},i}$, $0 \leq \mathsf{g} \leq n$, $0 \leq i \leq n$
- ⋆ Les variables $e_{\mathsf{g}_1,i,j,\mathsf{g}_2}$, $0 \leq \mathsf{g}_1 \leq n$, $0 \leq \mathsf{g}_2 \leq n$, $0 \leq i < n$, $0 \leq j \leq n$ définissent une adjacence formée par $(\mathsf{g}_1, \mathsf{g}_2)$ tel que $T_1(\mathsf{g}_1)$ soit égale à i, $T_2(\mathsf{g}_1)$ soit égale à j et $T_1(\mathsf{g}_2)$ soit égale à $i+1$: $e_{\mathsf{g}_1,i,j,\mathsf{g}_2} = 1$ si et seulement si
  1. $T_1(\mathsf{g}_1)$ est égale à i,
  2. $T_2(\mathsf{g}_1)$ est égale à j,
  3. $T_1(\mathsf{g}_2)$ est égale à $i+1$, et
  4. $T_2(\mathsf{g}_2)$ est égale à
     - $j+1$ si $s_1(\mathsf{g}_1) = s_2(\mathsf{g}_1)$ et $s_1(\mathsf{g}_2) = s_2(\mathsf{g}_2)$ et $j \neq n$,
     - $j-1$ si $s_1(\mathsf{g}_1) = -s_2(\mathsf{g}_1)$ et $s_1(\mathsf{g}_2) = -s_2(\mathsf{g}_2)$ et $j \neq 0$.

Objectif :

- ⋆ Maximiser $\displaystyle\sum_{0\leq \mathsf{g}_1\leq n}\ \sum_{0\leq i\leq n}\ \sum_{0\leq j\leq n}\ \sum_{0\leq \mathsf{g}_2\leq n} e_{\mathsf{g}_1,i,j,\mathsf{g}_2}$

Les contraintes :

- ⋆ Les contraintes de **(C".01)** à **(C".06)** sont équivalentes aux contraintes de **(C.a)** à **(C.c)** pour x égale à 1 puis égale à 2.
- ⋆ La contrainte **(C".07)** garantit la valeur des variables $e_{\mathsf{g}_1,i,j,\mathsf{g}_2}$ pour $i \neq n$ et $j \notin \{0, n\}$: $e_{\mathsf{g}_1,i,j,\mathsf{g}_2} = 1$ si et seulement si les variables $a^1_{\mathsf{g}_1,i}$, $a^2_{\mathsf{g}_1,j}$, $a^1_{\mathsf{g}_2,i+1}$ sont égales à 1 et
  - $a^2_{\mathsf{g}_2,j+1}$ est égale à 1 si $s_1(\mathsf{g}_1) = s_2(\mathsf{g}_1)$ et $s_1(\mathsf{g}_2) = s_2(\mathsf{g}_2)$;
  - $a^2_{\mathsf{g}_2,j-1}$ est égale à 1 si $s_1(\mathsf{g}_1) = -s_2(\mathsf{g}_1)$ et $s_1(\mathsf{g}_2) = -s_2(\mathsf{g}_2)$.
- ⋆ La contrainte **(C".08)** garantit la valeur des variables $e_{\mathsf{g}_1,i,j,\mathsf{g}_2}$ pour $i \neq n$ et $j = 0$: $e_{\mathsf{g}_1,i,0,\mathsf{g}_2} = 1$ si et seulement si $a^1_{\mathsf{g}_1,i}$, $a^2_{\mathsf{g}_1,0}$, $a^1_{\mathsf{g}_2,i+1}$ et $a^2_{\mathsf{g}_2,1}$ sont égales à 1, $s_1(\mathsf{g}_1) = s_2(\mathsf{g}_1)$ et $s_1(\mathsf{g}_2) = s_2(\mathsf{g}_2)$.
- ⋆ La contrainte **(C".09)** garantit que la valeur des variables $e_{\mathsf{g}_1,i,j,\mathsf{g}_2}$ pour $i \neq n$ et $j = n$: $e_{\mathsf{g}_1,i,n,\mathsf{g}_2} = 1$ si et seulement si $a^1_{\mathsf{g}_1,i}$, $a^2_{\mathsf{g}_1,n}$, $a^1_{\mathsf{g}_2,i+1}$ et $a^2_{\mathsf{g}_2,n-1}$ sont égales à 1, $s_1(\mathsf{g}_1) = -s_2(\mathsf{g}_1)$ et $s_1(\mathsf{g}_2) = -s_2(\mathsf{g}_2)$.

Le programme `Adjacence-2OP`

Objectif :

`Maximiser` $\sum_{0\leq \mathtt{g}_1\leq n} \sum_{0\leq i\leq n} \sum_{0\leq j\leq n} \sum_{0\leq \mathtt{g}_2\leq n} e_{\mathtt{g}_1,i,j,\mathtt{g}_2}$

Contraintes :

`C''.01` $\forall 0 \leq \mathtt{g} \leq n, \sum_{0\leq i\leq n} a^1_{\mathtt{g},i} = 1$

`C''.02` $\forall 0 \leq i \leq n, \sum_{0\leq \mathtt{g}\leq n} a^1_{\mathtt{g},i} = 1$

`C''.03` $\forall 0 \leq \mathtt{g}_1 \leq n, 0 \leq \mathtt{g}_2 \leq n, \mathtt{g}_1 \prec_1 \mathtt{g}_2, 0 < j \leq i \leq n, a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_2,j} \leq 1$

`C''.04` $\forall 0 \leq \mathtt{g} \leq n, \sum_{0\leq i\leq n} a^2_{\mathtt{g},i} = 1$

`C''.05` $\forall 0 \leq i \leq n, \sum_{0\leq \mathtt{g}\leq n} a^2_{\mathtt{g},i} = 1$

`C''.06` $\forall 0 \leq \mathtt{g}_1 \leq n, 0 \leq \mathtt{g}_2 \leq n, \mathtt{g}_1 \prec_2 \mathtt{g}_2, 0 < j \leq i \leq n, a^2_{\mathtt{g}_1,i} + a^2_{\mathtt{g}_2,j} \leq 1$

`C''.07` $\forall 0 \leq \mathtt{g}_1 \leq n, 0 \leq i \leq n, 1 \leq j < n, 0 \leq \mathtt{g}_2 \leq n,$
si $s_1(\mathtt{g}_1) = s_2(\mathtt{g}_1)$ et $s_1(\mathtt{g}_2) = s_2(\mathtt{g}_2)$
alors $a^1_{\mathtt{g}_1,i} + a^2_{\mathtt{g}_1,j} + a^1_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,j+1} - 4.e_{\mathtt{g}_1,i,j,\mathtt{g}_2} \geq 0$
et $a^1_{\mathtt{g}_1,i} + a^2_{\mathtt{g}_1,j} + a^1_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,j+1} - 4.e_{\mathtt{g}_1,i,j,\mathtt{g}_2} \leq 3$
sinon si $s_1(\mathtt{g}_1) = -s_2(\mathtt{g}_1)$ et $s_1(\mathtt{g}_2) = -s_2(\mathtt{g}_2)$
alors $a^1_{\mathtt{g}_1,i} + a^2_{\mathtt{g}_1,j} + a^1_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,j-1} - 4.e_{\mathtt{g}_1,i,j,\mathtt{g}_2} \geq 0$
et $a^1_{\mathtt{g}_1,i} + a^2_{\mathtt{g}_1,j} + a^1_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,j-1} - 4.e_{\mathtt{g}_1,i,j,\mathtt{g}_2} \leq 3$
sinon $e_{\mathtt{g}_1,i,j,\mathtt{g}_2} = 0$

`C''.08` $\forall\ 0 \leq \mathtt{g}_1 \leq n, 0 \leq i \leq n, 0 \leq \mathtt{g}_2 \leq n,$
si $s_1(\mathtt{g}_1) = s_2(\mathtt{g}_1)$ et $s_1(\mathtt{g}_2) = s_2(\mathtt{g}_2)$
alors $a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_1,0} + a^2_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,1} - 4.e_{\mathtt{g}_1,i,0,\mathtt{g}_2} \geq 0$
et $a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_1,0} + a^2_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,1} - 4.e_{\mathtt{g}_1,i,0,\mathtt{g}_2} \leq 3$
sinon $e_{\mathtt{g}_1,i,0,\mathtt{g}_2} = 0$

`C''.09` $\forall\ 0 \leq \mathtt{g}_1 \leq n, 0 \leq i \leq n, 0 \leq \mathtt{g}_2 \leq n,$
si $s_1(\mathtt{g}_1) = -s_2(\mathtt{g}_1)$ et $s_1(\mathtt{g}_2) = -s_2(\mathtt{g}_2)$
alors $a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_1,n} + a^2_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,n-1} - 4.e_{\mathtt{g}_1,i,n,\mathtt{g}_2} \geq 0$
et $a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_1,n} + a^2_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,n-1} - 4.e_{\mathtt{g}_1,i,n,\mathtt{g}_2} \leq 3$
sinon $e_{\mathtt{g}_1,i,n,\mathtt{g}_2} = 0$

FIGURE 4.7 – Le programme `Adjacence-2OP` renvoie le nombre maximum d'adjacences entre deux extensions linéaires de deux ordres partiels.

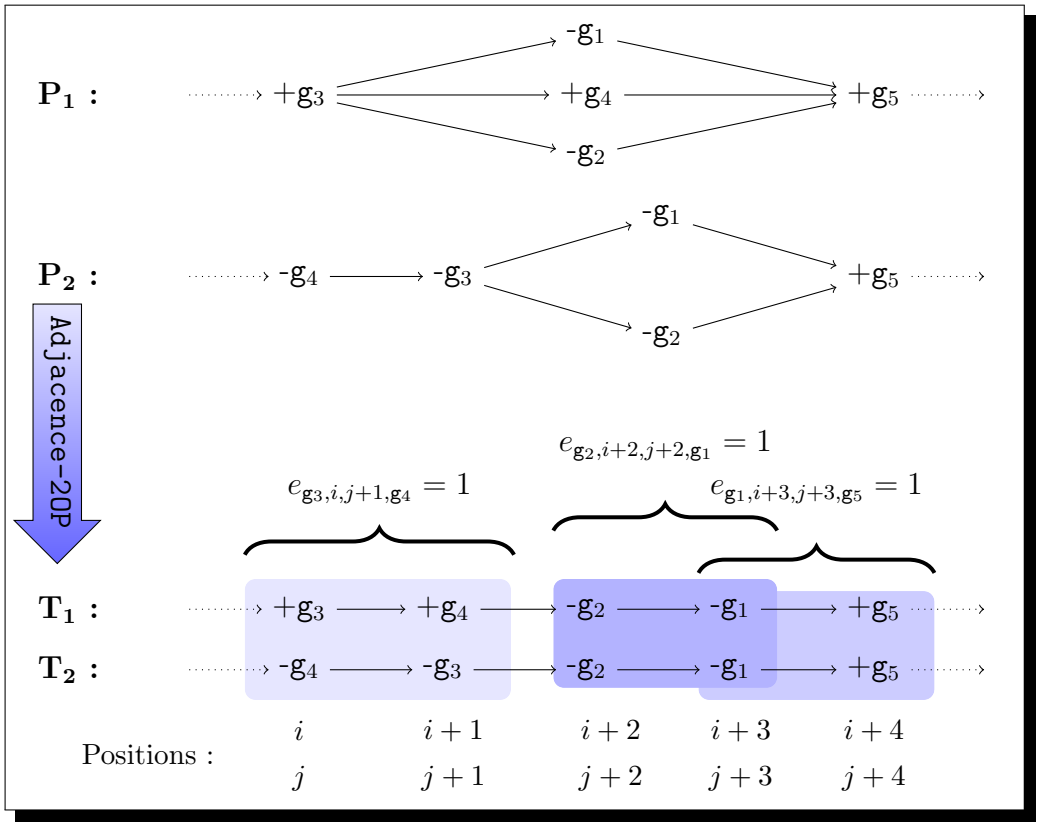


FIGURE 4.8 – **Illustration de `Adjacence-2OP`.** Soit P_1 et P_2 deux ordres partiels. Les gènes aux positions comprises entre i et $i+4$ forment 3 adjacences avec les paires de gènes (g_1, g_2), (g_1, g_5) et (g_3, g_4). Les variables $e_{g_3,i,j+1,g_4}$, $e_{g_2,i+2,j+2,g_1}$ et $e_{g_1,i+3,j+3,g_5}$ sont donc égales à 1.

4.1.2 Réduction des programmes

Le programme `IC-1OP` possède $O(n^3)$ variables et $O(n^3)$ contraintes. Le programme `Adjacence-1OP` possède $O(n^2)$ variables et $O(n^2)$ contraintes. Le programme `Adjacence-2OP` possède $O(n^4)$ variables et $O(n^4)$ contraintes. Nous rappelons que résoudre un programme linéaire à variables booléennes est un problème **NP**-complet [44]. Le temps de résolution de ces programmes est donc au moins exponentiel. C'est pourquoi, comme pour les programmes présentés au Chapitre 3, nous proposons des règles permettant, nous l'espérons, d'accélérer la résolution des programmes.

Ces règles permettent, d'une part de supprimer des variables et les contraintes associées, d'autre part, de calculer la valeur de certaines variables avant la résolution du programme linéaire à variables booléennes. Il est important de noter que le temps nécessaire à l'ajout de ces règles est négligeable par rapport au temps de résolution. En effet l'ajout de ces règles durant l'écriture du programme est effectué en temps polynomial alors que la résolution du programme est exponentiel en temps.

Nous supposons que x est dans $\{1; 2\}$ et que g, g_1, g_2, i, j et t sont dans $[0 : n]$.

Règles 1 : Positions restreintes

♦ Pour tout gène g, $T_x(\mathsf{g})$ appartient à $\mathsf{pos}_x(\mathsf{g})$. En accord avec cette remarque, la valeur de certaines des variables correspondant à g est calculée en prétraitement.

- Les variables $a^x_{\mathsf{g},i}$ sont égales à 0 si i n'est pas dans $\mathsf{pos}_x(\mathsf{g})$;
- Les variables $b_{\mathsf{g},i,t}$ sont égales à 0 si $T_1(\mathsf{g})$ ne peut pas être dans $[i : i + t]$;
- Les variables $b_{\mathsf{g},i,t}$ sont égales à 1 si $T_1(\mathsf{g})$ est obligatoirement dans $[i : i + t]$;
- Les variables $c_{\mathsf{g},i,t}$ sont égales à 0 s'il existe k dans $[0 : t]$ tel que $T_1(\mathsf{g} + k)$ ne peut pas être dans $[i : i + t]$;
- Les variables $c_{\mathsf{g},i,t}$ sont égales à 0 s'il existe un gène qui n'appartient pas à $[\mathsf{g}, \mathsf{g} + t]$ mais que sa position est obligatoirement dans $[i : i + t]$;
- Les variables $c_{\mathsf{g},i,t}$ sont égales à 1 si pour tout k dans $[0 : t]$ $T_1(\mathsf{g} + k)$ est obligatoirement dans $[i : i + t]$;
- Les variables $d_{\mathsf{g},i}$ sont égales à 0 si i n'est pas dans $\mathsf{pos}_1(\mathsf{g})$;
- Les variables $e_{\mathsf{g}_1,i,j,\mathsf{g}_2}$ sont égales à 0 si i n'est pas dans $\mathsf{pos}_1(\mathsf{g}_1)$, $i+1$ n'est pas dans $\mathsf{pos}_1(\mathsf{g}_2)$ ou
  - $i + 1$ n'est pas dans $\mathsf{pos}_2(\mathsf{g}_2)$ si $s_1(\mathsf{g}_1) = s_1(\mathsf{g}_2)$
  - $i - 1$ n'est pas dans $\mathsf{pos}_2(\mathsf{g}_2)$ si $s_1(\mathsf{g}_1) = -s_1(\mathsf{g}_2)$

♦ Dans le cas particulier où un gène g est une cheville ($\mathsf{pos}_1(\mathsf{g})$ n'a qu'un élément), d'autres valeurs de variables sont calculées en prétraitement.

- Les variables $a^x_{\mathsf{g},i}$ sont égales à 1 si g est une *i-cheville*$_x$.
- Les variables $d_{\mathsf{g},i}$ sont égales à 1 si le gène g est une *i-cheville*$_1$ et le gène $(\mathsf{g} + 1)$

est
  - une $(i+1)$-*cheville*$_2$ dans les cas où $s_1(\mathsf{g}) = \text{“}+\text{”}$ et $s_1(\mathsf{g}+1) = \text{“}+\text{”}$,
  - une $(i-1)$-*cheville*$_2$ dans les cas où $s_1(\mathsf{g}) = \text{“}-\text{”}$ et $s_1(\mathsf{g}+1) = \text{“}-\text{”}$.
- La variable $e_{\mathsf{g}_1,i,j,\mathsf{g}_2}$ est égale à 1 si le gène g_1 est une i-*cheville*$_1$ et une j-*cheville*$_2$, le gène g_2 est une $(i+1)$-*cheville*$_1$ et
  - une $(j+1)$-*cheville*$_2$ dans les cas où $s_1(\mathsf{g}_1) = s_1(\mathsf{g}_2)$,
  - une $(j-1)$-*cheville*$_2$ dans les cas où $s_1(\mathsf{g}_1) = -s_1(\mathsf{g}_2)$.

♦ La contrainte `C'.06` définit l'adjacence formée par le couple de gènes $(\mathsf{g}, \mathsf{g}+1)$, avec $s_1(\mathsf{g}) = \text{“}-\text{”}$ et $s_1(\mathsf{g}+1) = \text{“}-\text{”}$, aux positions n et $n-1$, respectivement. Sachant que le gène à la position n est $+n$, l'adjacence définie par la contrainte `C'06` ne sera jamais possible. Donc les variables $d_{\mathsf{g},n}$, pour g dans $[0 : n[$ sont égales à zéro et la contrainte `C'.06` est inutile.

Règles 2 : Réductions spécifiques au programme `IC-1OP`
♦ Par définition, tout intervalle de taille 1 ou n est commun à T_1 et T_2. Par conséquent, pour tout $0 \leq \mathsf{g} \leq n$, les sommes $\sum_{0 \leq i \leq n} c_{\mathsf{g},i,0}$ et que $\sum_{0 \leq i \leq n} c_{\mathsf{g},i,n}$ sont égales à 1. Donc, si nous ajoutons $n+1$ à l'objectif, nous pouvons éliminer toutes les variables $c_{\mathsf{g},i,0}$ et $c_{\mathsf{g},i,n}$ de l'objectif et dans toutes les contraintes associées à ces variables.

♦ Une seconde réduction consiste à supprimer certaines variables $c_{\mathsf{g},i,t}$, et leurs contraintes associées, lorsque nous sommes assurés que $c_{\mathsf{g},i,t}$ est égale à 0. En effet, s'il existe un entier $k \in [0 : t]$ tel que $pos_1(\mathsf{g}+k)$ ne peut pas appartenir à $[i : i+t]$ (c'est-à-dire un gène $\mathsf{g}+k$ ne peut pas être entre i et $i+t$ dans l'extension linéaire), alors nous sommes assurés que l'intervalle composé des gènes $\{\mathsf{g}; \mathsf{g}+1; \ldots; \mathsf{g}+t\}$ n'est pas un intervalle commun. La variable $c_{\mathsf{g},i,t}$ est donc égale à 0.

Règle 3 : Réduction spécifique à `Adjacence-1OP`
♦ Nous allons cette fois imposer un début d'extension linéaire sans perdre l'optimalité de la mesure maximisée. En effet, si entre deux chevilles, il n'existe pas d'ordre total permettant d'obtenir au moins une adjacence, alors nous pouvons choisir une extension linéaire quelconque entre ces deux chevilles.

Plus précisément, soient g_1 et g_2 deux gènes tel que g_1 est i-*cheville*$_1$ et g_2 est j-*cheville*$_1$ $(i < j)$. Si pour tout gène g_3 tel que $\mathsf{pos}_1(\mathsf{g}_3)$ soit inclut dans $[i : j]$ nous avons :
- $s_1(\mathsf{g}_3) \neq s_1(\mathsf{g}_3+1)$,
- $s_1(\mathsf{g}_3) = \text{“}+\text{”}$ et $s_1(\mathsf{g}_3+1) = \text{“}+\text{”}$ et $\nexists k \in \mathsf{pos}_1(\mathsf{g}_3)$ tel que $(k+1) \in \mathsf{pos}_1(\mathsf{g}_3+1)$,
- ou $s_1(\mathsf{g}_3) = \text{“}-\text{”}$ et $s_1(\mathsf{g}_3+1) = \text{“}-\text{”}$ et $\nexists k \in \mathsf{pos}_1(\mathsf{g}_3)$ tel que $(k+1) \in \mathsf{pos}_1(\mathsf{g}_3+1)$

alors aucune adjacence peut être créée par les gènes dont les positions sont comprises entre i et j, entre une extension linéaire de P_1 et l'identité. Donc nous pouvons définir un

ordre total quelconque pour les gènes $\mathtt{g}_3$ sans affecter le nombre d'adjacences.

Règle 4 : Réduction spécifique à `Adjacence-2OP`
♦ La **règle 3** peut être généralisée pour le problème `Adjacence-2OP` : entre deux chevilles $\mathtt{g}_1$ et $\mathtt{g}_2$ de P_1 (respectivement P_2) nous pouvons avoir une extension linéaire quelconque si aucune adjacence n'est possible pour les gènes appartenant à $[\mathtt{g}_1, \mathtt{g}_2]_1$ (respectivement $[\mathtt{g}_1, \mathtt{g}_2]_2$).

4.1.3 Cas de gènes non-signés

Parfois, les données d'ordre partiel ne possèdent pas d'information concernant le signe des gènes (c'est-à-dire le brin sur lequel sont situés les gènes). Les gènes sont alors représentés par des entiers naturels. Pour la mesure d'intervalles communs, le signe des gènes n'est pas significatif. Donc le programme `IC-1OP` n'a pas besoin d'être adapté.

Mais la définition d'adjacence est différente pour le cas non-signé. En effet, dans ce cas, dans T_1, deux gènes $\mathtt{g}_1$ et $\mathtt{g}_2$ forment une adjacence si et seulement si nous avons $\mathtt{g}_1$ $\mathtt{g}_2$ ou $\mathtt{g}_2$ $\mathtt{g}_1$ dans T_2. Par exemple entre la séquence signée $+0\ +1\ +3\ +2\ +4$ et l'identité signée $+0\ +1\ +2\ +3\ +4$, nous avons 1 adjacence mais dans le cas non-signé (entre la séquence non-signée 0 1 3 2 4 et l'identité non-signé 0 1 2 3 4) nous avons 2 adjacences. Par définition, le nombre d'adjacences dans le cas non-signé est supérieur ou égal au nombre d'adjacences dans le cas signé.

L'idée principale pour les programmes `Adjacence-1OP` et `Adjacence-2OP` est la même pour le cas non-signé mais les contraintes `C'.04`, `C'.05`, `C''.07`, `C''.08` et `C''.09` doivent être remplacées par les contraintes : `C'''.04`, `C'''.05`, `C'''.07`, `C'''.08` et `C'''.09`, respectivement (voir Figure 4.9). Dans le même ordre d'idée que précédemment, des simplifications peuvent être appliquées (détails omis).

Cas de gènes non-signés : nouvelles contraintes

Contraintes :

`C'''.04` $\forall\ 0 \leq \mathtt{g} < n, 1 \leq i < n,$
$a^1_{\mathtt{g},i} + a^1_{\mathtt{g}+1,i+1} + a^1_{\mathtt{g}+1,i-1} - d_{\mathtt{g},i} \leq 1$
$3 * d_{\mathtt{g},i} - a^1_{\mathtt{g},i} - a^1_{\mathtt{g}+1,i+1} - a^1_{\mathtt{g}+1,i+1} \leq 1$

`C'''.05` $\forall\ 0 \leq \mathtt{g} < n,$
$a^1_{0,0} + a^1_{1,1} - d_{0,0} \leq 1$
$3 * d_{0,0} - a^1_{0,0} - a^1_{1,1} \leq 1$

`C'''.07` $\forall 0 \leq \mathtt{g}_1 \leq n, 1 \leq i \leq n, 1 \leq j < n, 0 \leq \mathtt{g}_2 \leq n,$
$a^1_{\mathtt{g}_1,i} + a^2_{\mathtt{g}_1,j} + a^1_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,j-1} + a^2_{\mathtt{g}_2,j+1} - 4.e_{\mathtt{g}_1,i,j,\mathtt{g}_2} \leq 3$
$a^1_{\mathtt{g}_1,i} + a^2_{\mathtt{g}_1,j} + a^1_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,j-1} + a^2_{\mathtt{g}_2,j+1} - 4.e_{\mathtt{g}_1,i,j,\mathtt{g}_2} \geq 0$

`C'''.08` $\forall\ 0 \leq \mathtt{g}_1 \leq n, 0 \leq i \leq n, 0 \leq \mathtt{g}_2 \leq n,$
$a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_1,0} + a^2_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,1} - 4.e_{\mathtt{g}_1,i,0,\mathtt{g}_2} \leq 3$
$a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_1,0} + a^2_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,1} - 4.e_{\mathtt{g}_1,i,0,\mathtt{g}_2} \geq 0$

`C'''.09` $\forall\ 0 \leq \mathtt{g}_1 \leq n, 0 \leq i \leq n, 0 \leq \mathtt{g}_2 \leq n,$
$a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_1,} + a^2_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,n-1} - 4.e_{\mathtt{g}_1,i,n,\mathtt{g}_2} \leq 3$
$a^1_{\mathtt{g}_1,i} + a^1_{\mathtt{g}_1,n} + a^2_{\mathtt{g}_2,i+1} + a^2_{\mathtt{g}_2,n-1} - 4.e_{\mathtt{g}_1,i,n,\mathtt{g}_2} \geq 0$

FIGURE 4.9 – **Cas non-signé.** Pour les programmes `Adjacence-1OP` et `Adjacence-2OP` quelques contraintes doivent être modifiées dans le cas de gènes non-signés.

4.2 Résultats expérimentaux

Pour tester nos algorithmes, nous utilisons des données simulées. Ceci nous permet ainsi d'évaluer les performances de nos programmes en fonction de plusieurs caractéristiques inhérentes aux ordres partiels. Nous notons que les données réelles disponibles actuellement [1] sont de petites tailles (environ 30 gènes) et ne permettent donc pas, pour l'instant, d'évaluer correctement nos programmes.

Pour tous les calculs présentés dans ce chapitre, le solveur de programme linéaire utilisé est le logiciel `MiniSat+` [39]. Le choix d'utiliser ce solveur se justifie par le fait que ces temps de calcul sont significativement plus faibles par rapport aux temps de calculs des autres solveurs testés. Cette expérimentation a été effectuée sur un Quadri Intel(R) Xeon(TM) CPU 3,00 Ghz avec 16GB de mémoire tournant sous Linux.

L'ensemble des résultats obtenus est disponible à l'adresse www.lri.fr/∼thevenin.

L'ensemble des données de référence est tiré de [15]. Les auteurs ont généré des ordres partiels P_x en fonction de trois paramètres : la *taille* n, le *rapport d'ordre* p qui détermine le nombre d'attenants dans l'expression et la *règle de distribution de gène* q qui définit à la probabilité de points de cassure possibles par comparaison avec l'identité.

Nous utilisons 19 [2] instances différentes pour chaque triplet de paramètres (n, p, q) avec n dans $\{30; 40; 50; 60; 70; 80; 90\}$, p dans $\{0,7; 0,9\}$ et q dans $\{0,4; 0,6; 0,8\}$ ce qui induit 798 $(= 19 * 7 * 2 * 3)$ génomes.

D'une part, nous pouvons noter que plus p est grand, plus la largeur de l'ordre partiel est grande et plus le nombre d'extensions linéaires possibles est grand. Donc nous nous attendons à avoir plus d'intervalles communs et d'adjacences mais aussi des temps de calcul plus importants. D'autre part, plus q est grand, plus le nombre d'intervalles communs et d'adjacences sera grand. Nous verrons, par la suite, si la valeur de q influe bien sur la mesure et s'il agit sur le temps de calcul. Finalement, plus la taille des génomes est importante, plus le temps de calcul sera considérable.

Les résultats des trois programmes, sur cet ensemble de données, sont présentés dans la Sous-section 4.2.1. Nous évaluons deux critères : le temps (temps de l'écriture et de la résolution du programme linéaire) et la mesure (le nombre d'intervalles communs ou le nombre d'adjacences) optimisée par les extensions linéaires calculées. La largeur des ordres partiels donnés en entrée sera aussi considérée. Notons déjà que pour les 798 génomes étudiés, la largeur est comprise entre 1 et 22 et est en moyenne 6.

Nous étudions par ailleurs les avantages du programme `Adjacence-1OP` comparé à ceux d'`Adjacence-2OP` lorsque nous comparons un ordre partiel avec l'identité. Puis, nous évaluons la **règle 3** de simplification, vue dans la Sous-section 4.2.2. Finalement,

1. http ://www.gramene.org
2. 20 génomes cités dans l'article mais seulement 19 communiqués pas les auteurs.

dans la Sous-section 4.2.3, nous comparons les deux mesures.

4.2.1 Résultats pour les trois programmes

IC-1OP. Nous n'avons actuellement pas testé notre programme `IC-1OP` pour les 798 ordres partiels mais uniquement pour les 570 ordres partiels correspondant aux triplets suivants : pour $p = 0,9$ les triplets tel que $n \in [30; 40; 50; 60; 70; 80; 90\}$ et $\{0,4; 0,6; 0,8\}$ pour $p = 0,7$ les triplets tel que $n \in [30; 40; 50\}$ et $\{0,4; 0,6; 0,8\}$. Les **règles 1** et **2** ont été prises en compte lors de la création de ces programmes.

Parmis les 570 ordres partiels étudiés, nous avons obtenu 494 résultats (87%) - voir Tableau 4.1. Pour les 76 autres ordres, `MiniSat+` n'a pas résolu les programmes correspondants car le nombre de contraintes et de variables est trop grand (plus de 450 000 contraintes).

Au vue des résultats, nous remarquons que sans surprise, si p est faible (probabilité d'avoir des gènes attenants faible) ou si la largeur est grande alors le temps de calcul est plus important. Le paramètre q (probabilité d'avoir des adjacences) ne semble pas affecter le temps de calcul. Du point de vue de la mesure, si p est faible, si la largeur est grande ou si q est faible le nombre d'intervalles communs est plus important.

Adjacence-1OP Pour tester notre programme `Adjacence-1OP`, qui maximise le nombre d'adjacences entre un ordre partiel et l'identité, nous utilisons les 19 génomes correspondants à chaque triplet (n, p, q) avec $n \in \{30; 40; 50; 60; 70; 80; 90\}$, $p \in \{0,7; 0,9\}$ et $q \in \{0,4; 0,6; 0,8\}$. Les **règles 1** et **3** ont été prises en compte lors de la création de ces programmes.

Nous obtenons 779 résultats sur 798 ($97,5\%$) et ce en un peu moins de 2 mois 1/2 (plus 52 jours pour un unique calcul - voir ci-dessous) ; voir Tableau 4.2. Seulement 174 calculs ont mis plus d'une minute dont 71 qui ont mis plus d'une heure. Parmis ces 71 calculs, 14 ont mis plus d'une journée (voir Figure 4.10). Pour l'ensemble des génomes, la phase d'écriture du programme linéaire dure en moyenne 1 seconde.

Un de ces 779 calculs a nécessité près de 52 jours de calcul. Les paramètres de l'ordre partiel correspondant sont $n = 90$, $p = 0,7$ et $q = 0,4$, sa largeur est 17 et le nombre d'adjacences obtenu est 42. Ce temps de calcul est exceptionnel (nous avons "seulement" 16 jours de calcul au maximum pour les autres calculs). C'est pourquoi, afin de ne pas biaiser l'ensemble des résultats, ce calcul n'est pas pris en compte dans la suite de cette section (en particulier dans le Tableau récapitulatif 4.2 et la Figure 4.10).

Pour les 70 autres cas où le temps de calcul est supérieur à 1 heure, nous notons que les génomes correspondant ont une taille supérieure ou égale à 50, un rapport d'ordre p généralement égal à $0,7$ (58 fois sur 70) et une règle de distribution q qui influe peu (21 fois 0,4, 27 fois 0,6 et 22 fois 0,8). Leur largeur est comprise entre 5 à 22 et est en moyenne 11,6 (contre 6 pour l'ensemble des 798 génomes).

Génomes						Résultats de IC-1OP		
Taille	p	q	Largeur	Critère	Largeur	Largeur	Intervalles Communs	Temps
*	*	*	*	Somme	2612	1954	39253	53 jours
				Moyenne	4,6	4	79,5	2 h 34
				Min	1	1	37	< 1 s
				Max	18	13	182	13 jours 13
				Nb-res	570	494	494	494
30	*	*	*	Somme	493	358	979	4 h 40
				Moyenne	4,3	3,65	8,6	3 min
				Min	1	1	1	< 1 s
				Max	14	10	19	2 h 36
				Nb-res	114	98	98	98
40	*	*	*	Somme	580	358	5708	2 jours 13
				Moyenne	5	4	64	41 min
				Min	1	1	46	3 s
				Max	16	10	127	2 jours 04
				Nb-res	114	89	89	89
50	*	*	*	Somme	645	478	7760	35 jours
				Moyenne	5,6	4,8	78,38	8 h 30
				Min	2	2	57	6 s
				Max	18	13	182	13 jours 13
				Nb-res	114	99	99	99
60	*	*	*	Somme	209	202	4518	2 jours 17
				Moyenne	3,67	3,6	80,7	1 h 09
				Min	2	2	67	10 s
				Max	7	6	111	42 h 31
				Nb-res	57	56	56	56
70	*	*	*	Somme	214	184	4839	6 jours 18
				Moyenne	3,75	3,5	93	3 h 07
				Min	2	2	78	16 s
				Max	7	6	135	5 jours 03
				Nb-res	57	52	52	52
80	*	*	*	Somme	212	189	5595	11 h 15
				Moyenne	3,7	3,57	105,6	12 min 44
				Min	2	2	87	36 s
				Max	7	7	161	4 h 29
				Nb-res	57	53	53	53
90	*	*	*	Somme	259	195	5418	5 jours 04
				Moyenne	4,54	4	112,88	2 h 32
				Min	2	2	98	52 s
				Max	9	8	163	3 jours 16
				Nb-res	57	48	48	48

TABLE 4.1 – **Résumé des résultats obtenus par `IC-1OP`.** Voir page 158 pour un mode d'emploi du tableau.

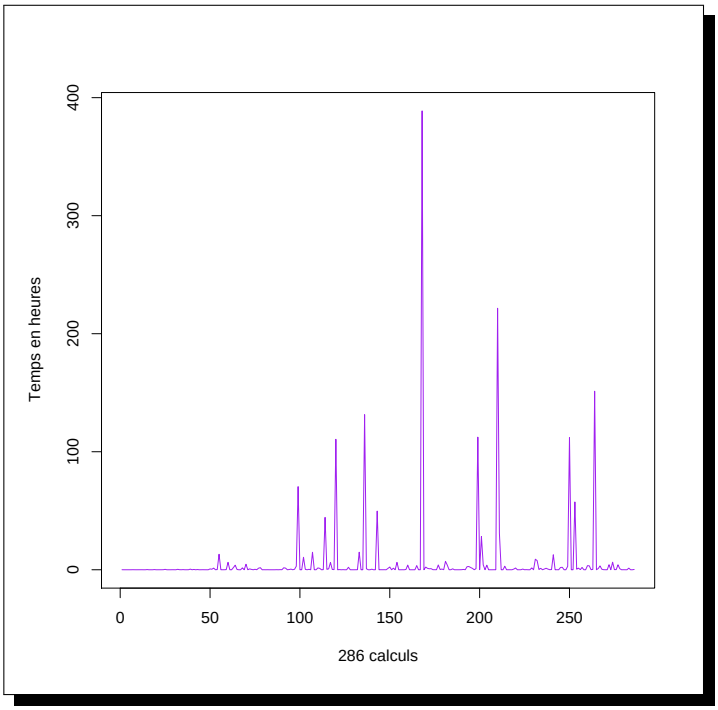


FIGURE 4.10 – **Temps de calcul (en heures) de** `Adjacence-1OP`. Seul les 286 (sur 788) calculs qui ont nécessité au moins 5 secondes sont indiqués. L'ordre des calculs en abscisse est arbitraire.

Les graphes de la Figure 4.14 (voir Section-Annexe 4.4) permettent d'observer l'influence de trois paramètres (p, q et largeur) sur le temps de calcul. La moyenne du temps de calcul pour l'ensemble des génomes d'une taille donnée est indiquée sous forme d'histogramme. Nous remarquons que le paramètre p est le plus influant sur le temps de calcul. Ce dernier est nettement plus important lorsque $p = 0,7$ plutôt que lorsque $p = 0,9$, c'est-à-dire lorsque les ordres partiels ont moins d'attenants. La largeur influe elle aussi sur le temps : plus elle est grande, plus le temps de calcul est important. Par contre le paramètre q ne semble pas influer sur le temps.

Les graphes de la Figure 4.15 (voir Section-Annexe 4.4) permettent d'observer l'influence de trois paramètres (p, q et largeur) sur le nombre d'adjacences maximal. La moyenne du nombre d'adjacences maximal pour l'ensemble des génomes d'une taille donnée est indiquée sous forme d'histogramme. Nous remarquons que le nombre d'adjacences dépend principalement de la taille des génomes. De plus, il est plus important lorsque p est petit, c'est-à-dire lorsque plus d'extensions linéaires sont possibles. Le nombre d'adjacences maximal est aussi proportionnel à la largeur (qui dépend principalement de p). Nous observons, et c'est inhérent de par la définition de q, que le nombre maximum d'adjacences est inversement proportionnel à q.

Pour 19 génomes le programme linéaire `Adjacence-1OP` n'a pu être résolu par `MiniSat+` car le nombre de contraintes et de variables est trop grand pour ce solveur. Pour ces 19 génomes, nous observons que leur taille est supérieure ou égale à 70, que p est égal à $0,7$ et que q n'est pas spécifique (9 fois $0,4$, 3 fois $0,6$ et 8 fois $0,8$). Leur largeur est importante : en moyenne $17,5$ (de 11 à 29) contre 6 pour l'ensemble des génomes.

Afin d'exécuter ces 19 calculs, une solution serait d'utiliser un autre solveur. Les tests effectués avec le solveur `CPLEX` pour les génomes de taille 30, montre que le temps de calcul de `Adjacence-1OP` est alors nettement plus important (une différence de quelques secondes à plusieurs heures, pour chacune des comparaisons). C'est pourquoi, nous n'avons pas testé ces 19 génomes avec ce solveur. Rappelons que les programmes linéaires à variables booléennes présentés dans le Chapitre 3 étaient eux résolu plus rapidement avec le solveur `CPLEX` qu'avec le solveur `MiniSat+`. Ceci montre clairement que le choix de solveur est une étape fondamentale dans la résolution de programme linéaire. Ils possèdent chacun des spécificités qui influent énormément sur le temps de calcul.

En plus de simuler les génomes utilisés pour notre propre expérimentation, Blin *et al.* proposent un algorithme de programmation dynamique pour résoudre MAL-1OP. Cet algorithme ainsi que les résultats obtenus avec ces génomes simulés sont présentés dans [15]. Il est cependant difficile de comparer leurs temps de calcul avec les notres puisque l'ordinateur utilisé est différent. Nous pouvons tout de même remarquer que, tout comme nous, leur temps de calcul est nettement plus important lorsqu'il s'agit d'étudier des génomes dont la largeur est importante (supérieur à 10).

Ceci n'est guère étonnant. En effet lorsque la largeur d'un génome est grande l'espace des solutions à étudier (c'est-à-dire le nombre d'extensions linéaires possibles) est plus important. Quelle que soit la méthode employée pour résoudre MAL-1OP (ou de manière équivalente MICL-1OP et MAL-2OP), nous pouvons donc nous attendre à une résolution plus longue.

Adjacence-2OP Pour chaque triplet (n, p, q) étudié, nous comparons les 19 génomes deux à deux. Actuellement, nous avons obtenu les résultats pour les 12 triplets suivants : $\{30; 0,7; 0,8\}$, $\{30; 0,9; 0,4\}$, $\{30; 0,9; 0,6\}$, $\{30; 0,9; 0,8\}$, $\{40; 0,9; 0,4\}$, $\{40; 0,9; 0,6\}$, $\{40; 0,9; 0,8\}$, $\{50; 0,9; 0,6\}$, $\{50; 0,9; 0,8\}$, $\{60; 0,9; 0,6\}$, $\{60; 0,9; 0,8\}$, $\{70; 0,9; 0,8\}$ (voir Tableau 4.4 de la Section-annexe 4.4). Pour chaque triplet nous comparons les ordres partiels deux à deux, nous avons donc 171 comparaisons possibles pour chaque triplet. Notons que la **règle 5**, permettant de calculer en prétraitement certaines variables, n'a pas encore été programmée. Seule la **règle 1** est prise en compte lors de l'écriture de ces programmes linéaires à variables booléennes.

Avant d'étudier les 11 triplets pour lesquels $p = 0,9$, étudions les résultats obtenus pour le triplet $\{30; 0,7; 0,8\}$. Le temps de calcul est nettement plus important lorsque $p = 0,7$ (2 jours et 21 heures pour le triplet $\{30; 0,7; 0,8\}$ contre 40 minutes pour le triplet $\{30; 0,9; 0,8\}$). De plus, moins de résultats ont été obtenus (53% résultats obtenus pour le triplet $\{30; 0,7; 0,8\}$ contre 100% pour le triplet $\{30; 0,9; 0,8\}$).

Pour les 11 triplets pour lesquels $p = 0,9$, nous avons 1881 comparaisons à effectuer Nous parvenons à obtenir 1668 ($88,6\%$) résultats en un peu plus de 38 jours. Pour les 213 calculs non achevés, deux types de problèmes sont apparus : soit le programme linéaire possède trop de variables et contraintes pour le solver `MiniSat+`, soit la mémoire demandée est trop importante pour le Dual Intel de 2GB de mémoire utilisé. Pour résoudre le premier problème, nous pouvons utiliser d'autres solvers, sachant que le solveur `CPLEX` ne permet pas non plus de résoudre ces programmes linéaires. Pour le second problème, des calculs sont en train d'être réalisés afin d'obtenir de nouveaux résultats à l'aide du Quadri Intel à 16GB de mémoire. Dans les deux cas, une solution serait de définir des programmes linéaires plus performants (en utilisant des règles de réduction par exemple).

La largeur influe clairement sur le temps de calculs. En effet, lorsque la somme des largeurs des deux ordres partiels P_1 et P_2 est inférieur à 5, la moyenne du temps de calculs est 14 secondes contre 56 minutes lorsque cette somme est supérieur à 5 (voir Tableau 4.3). Le paramètre q influe seulement sur le nombre d'adjacences : plus q est petit, plus le nombre d'adjacences est important. Plus la taille est importante, plus les temps de calculs et la demande en mémoire sont importants.

Génomes						Résultats de Adjacence-1OP		
Taille	p	q	Largeur	Critère	Largeur	Largeur	Adjacence	Temps
*	*	*	*	Somme	4851	4500	9475	73 jours 2
				Moyenne	6,1	5,8	12,2	2 jours 15
				Min	1	1	0	< 1 s
				Max	28	22	39	16 jours 4
				Nb-res	798	778	778	778
30	*	*	*	Somme	493	493	979	20 min 34
				Moyenne	4,3	4,3	8,6	11 s
				Min	1	1	1	< 1 s
				Max	14	14	19	9 min 28
				Nb-res	114	114	114	114
40	*	*	*	Somme	580	580	1144	2 h 18
				Moyenne	5	5	10	1 min 13
				Min	1	1	0	< 1 s
				Max	16	16	25	31 min 23
				Nb-res	114	114	114	114
50	*	*	*	Somme	645	645	1320	1 jour 15
				Moyenne	5,6	5,6	11,6	20 min 45
				Min	2	2	1	< 1 s
				Max	18	18	34	13 h 07
				Nb-res	114	114	114	114
60	*	*	*	Somme	690	690	1409	11 jours 7
				Moyenne	6	6	12,3	2 h 23
				Min	2	2	1	< 1 s
				Max	19	19	30	4 jours 14
				Nb-res	114	114	114	114
70	*	*	*	Somme	747	701	1553	24 jours 10
				Moyenne	6,5	6,2	13,8	5 h 14
				Min	2	2	2	< 1 s
				Max	29	22	39	16 jours 4
				Nb-res	114	112	112	112
80	*	*	*	Somme	825	678	1471	17 jours 15
				Moyenne	7,2	6,4	13,9	3 h 59
				Min	2	2	1	< 1 s
				Max	28	18	33	9 jours 5
				Nb-res	114	106	106	106
90	*	*	*	Somme	871	713	1601	16 jours 11
				Moyenne	7,6	6,8	15,4	3 h 47
				Min	2	2	2	< 1 s
				Max	23	17	38	6 jours 7
				Nb-res	114	104	104	104
*	0,7	*	*	Somme	3441	3090	5940	71 jours 5
				Moyenne	8,6	8,1	15,7	4 h 30
				Min	2	2	2	< 1 s
				Max	29	22	39	16 jours 4
				Nb-res	399	379	379	379
*	0,9	*	*	Somme	1410	1410	3535	1 jour 21
				Moyenne	3,5	3,5	8,8	6 min 46
				Min	1	1	0	< 1 s
				Max	9	9	30	7 h 8
				Nb-res	399	399	399	399
*	*	0,4	*	Somme	1664	1506	4277	14 jours 20
				Moyenne	6,3	5,8	16,6	1 jour 23
				Min	1	1	4	< 1 s
				Max	28	19	39	5 jours 11
				Nb-res	266	257	257	257
*	*	0,6	*	Somme	1580	1530	2917	16 jours 9
				Moyenne	5,9	5,8	11,1	1 jour 29
				Min	1	1	1	1 s
				Max	19	17	31	4 jours 16
				Nb-res	266	263	263	263
*	*	0,8	*	Somme	1607	1464	2281	41 jours 21
				Moyenne	6	5,7	8,8	3 h 53
				Min	2	2	0	< 1 s
				Max	29	22	31	16 jours 4
				Nb-res	266	258	258	258
*	*	*	<11	Somme	3414	3414	7537	7 jours 3
				Moyenne	4,9	4,9	10,8	14 min 48
				Min	1	1	0	< 1 s
				Max	10	10	30	2 jours 9
				Nb-res	696	696	696	696
*	*	*	>10	Somme	1437	1086	1938	65 jours 23
				Moyenne	14,1	13,2	23,6	19 h 18
				Min	11	11	12	3 s
				Max	29	22	39	16 jours 4
				Nb-res	102	82	82	82

TABLE 4.2 – **Résumé des résultats obtenus par `Adjacence-1OP`.** Voir page 158 pour un mode d'emploi du tableau.

Somme des largeurs	2	3	4	5	6	7	8	9	10	11
Temps moyen	1	0,6	0,6	23,2	198	3785	3142	8063	29622	14573

TABLE 4.3 – **Temps moyens en seconde de** `Adjacence-2OP` en fonction de la somme de la largeur des deux ordres partiels donnés en instance.

Comparaison entre `Adjacence-1OP` et `Adjacence-2OP` Dans le but d'évaluer `Adjacence-1OP`, nous comparons ses temps de calcul avec ceux de `Adjacence-2OP` pour les cas où P_2 est l'identité. Pour les 19 instances de chaque triplet (n, p, q) tel que n soit dans $\{30; 40\}$, p dans $\{0,7; 0,9\}$ et q dans $\{0,4; 0,6; 0,8\}$, la somme des temps de calcul est de 2 heures 30 (minimum = 0,1 seconde et maximum = 15 minutes 25) pour `Adjacence-1OP` et de 67 heures (minimum = 0,1 seconde et maximum = 14 heures 42) pour `Adjacence-2OP`.

Il est clair que `Adjacence-1OP` est nettement plus avantageux que `Adjacence-2OP` du point de vue du temps de calcul (une diminution de $96,3\%$). À l'aide de la Figure 4.11, nous remarquons que les deux programmes ont des temps de calculs importants pour les mêmes génomes. Nous pouvons en déduire que les deux programmes sont confrontés aux mêmes difficultés de résolution. Ceci n'est pas étonnant au vue des similarités des variables et des contraintes des programmes `Adjacence-1OP` et `Adjacence-2OP`.

4.2.2 Contribution d'une règle de réduction

Dans la Section 4.1.2, nous proposons différentes règles permettant de fixer les valeurs de variables avant même la résolution du programme. Nous espérons ainsi diminuer le temps de calcul global.

Afin d'évaluer l'intérêt de la **règle 3**, nous comparons les temps de résolution des programmes lors de l'utilisation ou l'absence d'utilisation de cette règle. Cette évaluation peut nous indiquer s'il est avantageux de l'ajouter.

Pour évaluer la **règle 3**, nous comparons le temps de calcul avec et sans cette règle. Pour les 19 instances de chaque triplet (n, p, q) (n dans $\{30; 40\}$, p dans $\{0,7; 0,9\}$ et q dans $\{0,4; 0,6; 0,8\}$) nous obtenons tous les résultats. La règle s'applique à 38 ordres partiels sur les 228 étudiés. Elle permet de définir des ordres totaux entre deux chevilles séparées en moyenne par 3 gènes (de 2 à 7 gènes).

Le temps d'écriture des 228 programmes linéaires à variables booléennes n'est pas modifié par l'ajout de cette règle. Le temps de résolution de `Adjacence-1OP` est de 2 heures et 29 minutes lorsque nous n'avons pas la **règle 3** et de 2 heures et 28 minutes avec cette règle. Pour les 38 cas où la règle peut être appliquée, le temps de calcul est de 26 secondes avec la règle et 31 secondes sans.

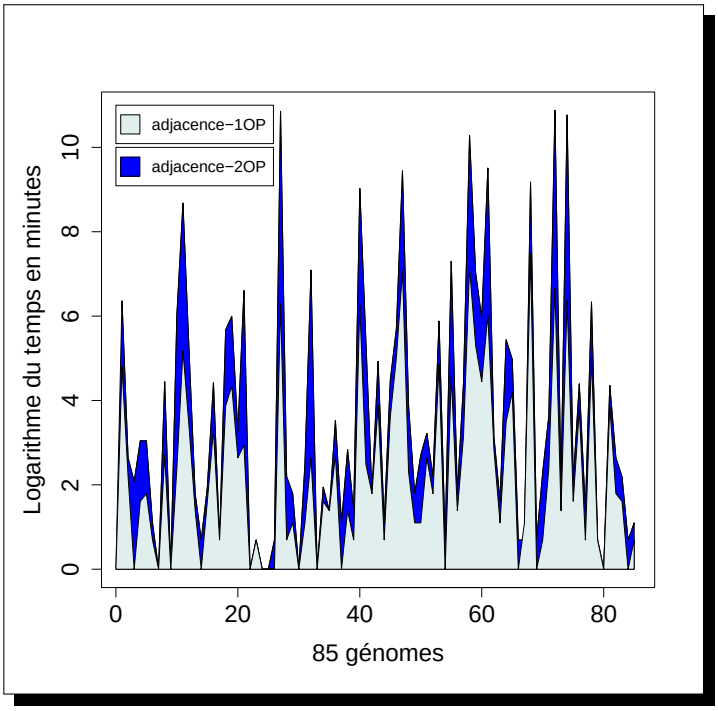


FIGURE 4.11 – **Les temps de calcul de** `Adjacence-1OP` **et de** `Adjacence-2OP` sont indiqués à échelle logarithmique. Les triplets (n, p, q) pris en compte sont ceux pour lesquels n est dans $\{30; 40\}$, p est dans $\{0,7; 0,9\}$ et q est dans $\{0,4; 0,6; 0,8\}$. Seuls les calculs pour lequel le temps est supérieur à 1 seconde, pour les deux programmes, sont indiqués : 85 calculs sur 228. L'ordre des calculs en abscisse est arbitraire.

Il semblerait donc que la **règle 3** ne permette pas significativement d'accélérer la résolution du programme `Adjacence-1OP`. Mais il serait judicieux d'évaluer l'impact de cette règle pour le cas de génomes où elle pourrait être appliquée entre des chevilles espacées par plus de gènes, en particulier si l'anti-chaîne maximale entre ces deux chevilles est grande.

4.2.3 Comparaison des deux mesures

Nous avons étudié parallèlement deux mesures : le nombre d'intervalles communs et le nombre d'adjacences. Nous désirons savoir maintenant si ces mesures apportent des informations différentes.

Nous notons tout d'abord que pour toute adjacence, formée entre T_1 et T_2, nous avons un intervalle commun de taille 2. En effet, pour toute paire de gènes $(\mathtt{g}_1, \mathtt{g}_2)$ formant une adjacence entre T_1 et T_2, nous avons la sous-séquence $\mathtt{g}_1\ \mathtt{g}_2$ ou $-\mathtt{g}_2\ -\mathtt{g}_1$ dans les deux extensions linéaires ; par conséquent il y a un intervalle commun de taille 2 entre T_1 et T_2.

De plus, nous pouvons avoir un nombre minimal d'adjacence (0) et un nombre maximal d'intervalles communs ($\sum_{0 \leq i \leq n}(i)$) entre un ordre total avec n gènes et l'identité. En effet, nous avons 15 intervalles communs et 0 adjacence entre la séquence $+0\ -1\ +2\ -3\ +4$ et l'identité.

Pour comparer ces deux mesures, nous proposons deux méthodes.

Premièrement, nous pouvons comparer directement les mesures optimales. En particulier, nous regardons si le nombre d'intervalles communs et le nombre d'adjacences sont importants pour les mêmes génomes. La Figure 4.12 indique le nombre d'intervalles communs maximal (obtenu par `IC-1OP`) et le nombre d'adjacences maximal (optenu par `Adjacence-1OP`) entre l'identité et un génome tel que n est dans $\{30; 40\}$, p est dans $\{0,7; 0,9\}$ et q est dans $\{0,4; 0,6; 0,8\}$.

Clairement, et c'est logique, le nombre d'intervalles communs est plus important. Dans la majorité des cas, les nombres d'intervalles communs et d'adjacences maximaux sont importants pour les mêmes génomes partiellement ordonnés. Mais il est intéressant de noter que pour certains génomes ce n'est pas le cas. En effet, leur nombre d'intervalles communs est très nettement supérieur à la moyenne du nombre d'intervalles communs maximal alors que ce phénomène n'est pas présent pour leur nombre d'adjacences. Il serait intéressant de voir si une telle différence existe pour des génomes réels. Cela pourrait mettre en avant des ensembles de gènes qui restent physiquement proches dans les génomes (nombre d'intervalles communs important) mais dont l'ordre ou le signe/brin des gènes à l'intérieur de cet ensemble est quelconque (nombre d'adjacences faible).

Deuxièmement, nous pouvons calculer une mesure entre les extensions linéaires obtenues *via* l'optimisation de l'autre mesure. Par exemple, en comparant P_1 à l'identité à l'aide de `IC-1OP` nous obtenons T_1. Puis, nous calculons le nombre d'adjacences entre T_1 et l'identité.

Mais avant cela, nous notons tout d'abord que maximiser le nombre d'intervalles communs entre P_1 et P_2 peut être obtenu par différentes extensions linéaires. Cette diversité au niveau de ces dernières, apporte une diversité dans le nombre d'adjacences qui sera alors calculé (voir Figure 4.16 (a)). De même, un nombre maximum d'adjacences entre P_1 et P_2 peut être obtenu par diverses extensions linéaires et donc conduire à différents nombre d'intervalles communs (voir Figure 4.16 (b)). Il faut donc être prudent lors de l'analyse des résultats.

Pour cette méthode de comparaison de mesure, un travail préliminaire a été effectué pour les génomes tels que n est dans $\{30; 40\}$, p est dans $\{0,7; 0,9\}$ et q est dans $\{0,4; 0,6; 0,8\}$. Nous comparons le nombre d'intervalles communs obtenus de façon optimale, grâce à `IC-1OP`, avec celui obtenu par `Adjacence-1OP` lors de l'optimisation du nombre d'adjacences (voir Figure 4.13). Nous pouvons remarquer que le nombre d'intervalles communs calculé par `Adjacence-1OP` est légèrement plus faible pour une majorité des génomes étudiés. Par contre, il arrive que le nombre d'intervalles soit très différents, en particulier pour les génomes pour lesquels le rapport d'ordre p est égal à $0,7$ et d'autant plus lorsque leur taille est égale à 40 (et non à 30). De plus, notons que dans 90% des cas le programme `IC-1OP` renvoie des extensions liénaires tel que nous obtenons aussi le nombre maximum d'adjacences. Par contre, le programme `Adjacence-1OP` renvoie uniquement dans 16% des cas le nombre d'intervalles communs maximum.

Le Tableau 4.5 (Section-Annexe 4.4) précise les nombres d'intevalles communs obtenus par `Adjacence-1OP` pour les 778 génomes pour lequels un résultat a été obtenu.

Nous avons donc vu que l'optimisation du nombre d'intervalles communs ou du nombre d'adjacences conduit à des résultats différents. Dans le cadre de la génomique comparative, il serait nécessaire maintenant de comparer ces deux mesures à l'aide de génomes réels.

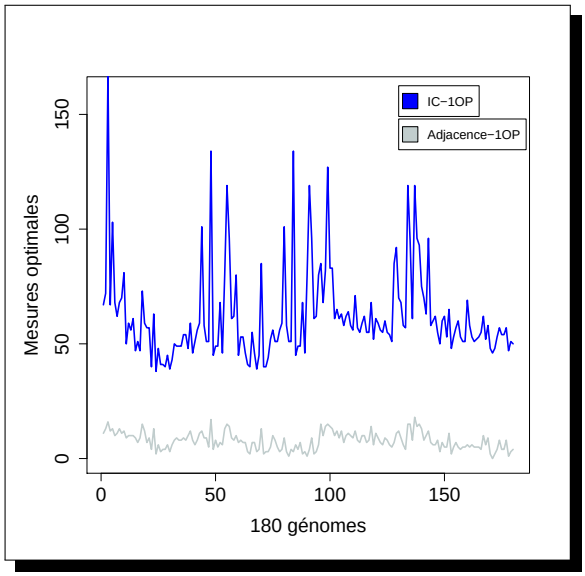


FIGURE 4.12 – **Mesures optimisées par `Adjacence-1OP` et `IC-1OP`** pour les triplets (n, p, q) tels que n est dans $\{30; 40\}$, p est dans $\{0,7; 0,9\}$ et q est dans $\{0,4; 0,6; 0,8\}$. L'ordre des génomes en abscisse est arbitraire.

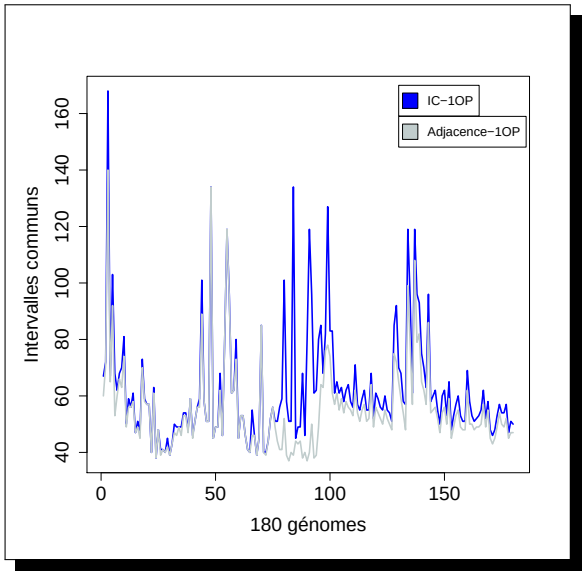


FIGURE 4.13 – **Nombres d'intervalles communs obtenus par `IC-1OP` et `Adjacence-1OP`** pour les triplets (n, p, q) tels que n est dans $\{30; 40\}$, p est dans $\{0,7; 0,9\}$ et q est dans $\{0,4; 0,6; 0,8\}$. L'ordre des génomes en abscisse est arbitraire.

4.3 Conclusions

Dans ce chapitre, nous avons introduit une méthode générique pour comparer de façon exacte deux genomes dont l'ordre des gènes n'est que partiellement connu. Cette méthode utilise la programmation linéaire à variables booléennes. Plus précisément, nous écrivons un programme qui permet d'obtenir des extensions linéaires de génomes partiellement ordonnés grâce à l'optimisation d'une mesure. Notre approche générale peut être adaptée à différentes mesures de (dis)similarité (intervalles conservés, MAD, etc).

Actuellement, nous avons appliqué cette méthode pour calculer le nombre d'intervalles communs entre un ordre partiel et l'identité et calculer le nombre d'adjacences entre deux ordres partiels (avec une amélioration si un ordre partiel est un ordre total). Nous rappelons que lorsque nous comparons des génomes sans duplication, maximiser le nombre d'adjacences est équivalent à minimiser le nombre de points de cassure.

Des expérimentations ont été entreprises sur des données simulées montrant la validité de notre approche. En effet, nous avons obtenu de nombreux résultats, en particulier pour les programmes basés sur l'optimisation du nombre d'adjacences entre un génome partiellement ordonné et un génome totalement ordonné. Nous avons pu remarquer que la difficulté des problèmes MICL-1OP, MAL-1OP et MAL-2OP est fortement dépendante de la largeur des génomes. Par contre, le temps de résolution semble indépendant de la similarité entre les génomes.

Ce travail est dans une phase préliminaire et de nombreux travaux sont encore à effectuer :

- Tester nos programmes avec des données réelles (tirées du site Gramene[3] [62] par exemple, voir Section-Annexe 4.4) ;
- Évaluer l'ensemble des règles que nous avons proposé dans la Section 4.1.2 ;
- Pour chacun des trois programmes, déterminer d'autres règles fortes et pertinentes pour accélérer le processus en évitant la génération d'un trop grand nombre de variables et de contraintes ;
- Généraliser le programme `IC-1OP` pour optimiser le nombre d'intervalles commun entre deux ordres partiels afin de résoudre MICL-2OP ;
- Définir des heuristiques et les évaluer grâce aux résultats exacts obtenus jusqu'ici ;
- Utiliser une approche similaire pour calculer d'autres mesures et/ou prendre en compte les duplications de gènes.

3. http ://www.gramene.org

4.4 Annexes

4.4.1 Lecture des Tableaux 4.1, 4.2 et 4.4

Voici quelques explications pour mieux comprendre le Tableau 4.1, 4.2 et 4.4.

Les résultats sont regroupés par série de 5 lignes. Une série correspond aux résultats obtenus à partir de l'ensemble des génomes suivant les même critères. Ces critères sont indiqués par les 4 premières colonnes : la taille, le rapport d'ordre p, la règle de distribution q et la largeur.

Pour chaque de ces critères
- soit la valeur est spécifiée : elle est indiquée dans une des 4 premières colonnes,
- soit la valeur est quelconque : une étoile est placée dans la colonne correspondante.

Pour chaque série, une ligne représente
- soit une somme,
- soit une moyenne,
- soit un minimum,
- soit un maximum,
- soit un nombre de résultats à obtenir (partie de gauche) ou obtenus (partie de droite).

Les colonnes correspondent de gauche à droite :
- aux 4 critères
- à la légende
- à la largeur de tous les génomes correspondant à la série,
- à la largeur, le nombre d'adjacences obtenu par `Adjacence-1OP` et le temps de calcul de `Adjacence-1OP` pour l'ensemble des génomes de la série pour lequel un résultat a été obtenu par `Adjacence-1OP`.

`Exemple :` En moyenne, les 798 génomes ont une largeur égale à 6. Le nombre maximal d'adjacences, pour les génomes de taille 30, a été obtenu au maximum en 9 minutes et 28 secondes. Le programme `Adjacence-1OP` a permis d'obtenir le nombre maximum d'adjacences pour 379 génomes dont $p = 0,7$ (sur les 399 génomes dont $p = 0,7$).

4.4.2 Détails des résultats de `Adjacence-1OP` et `Adjacence-2OP`

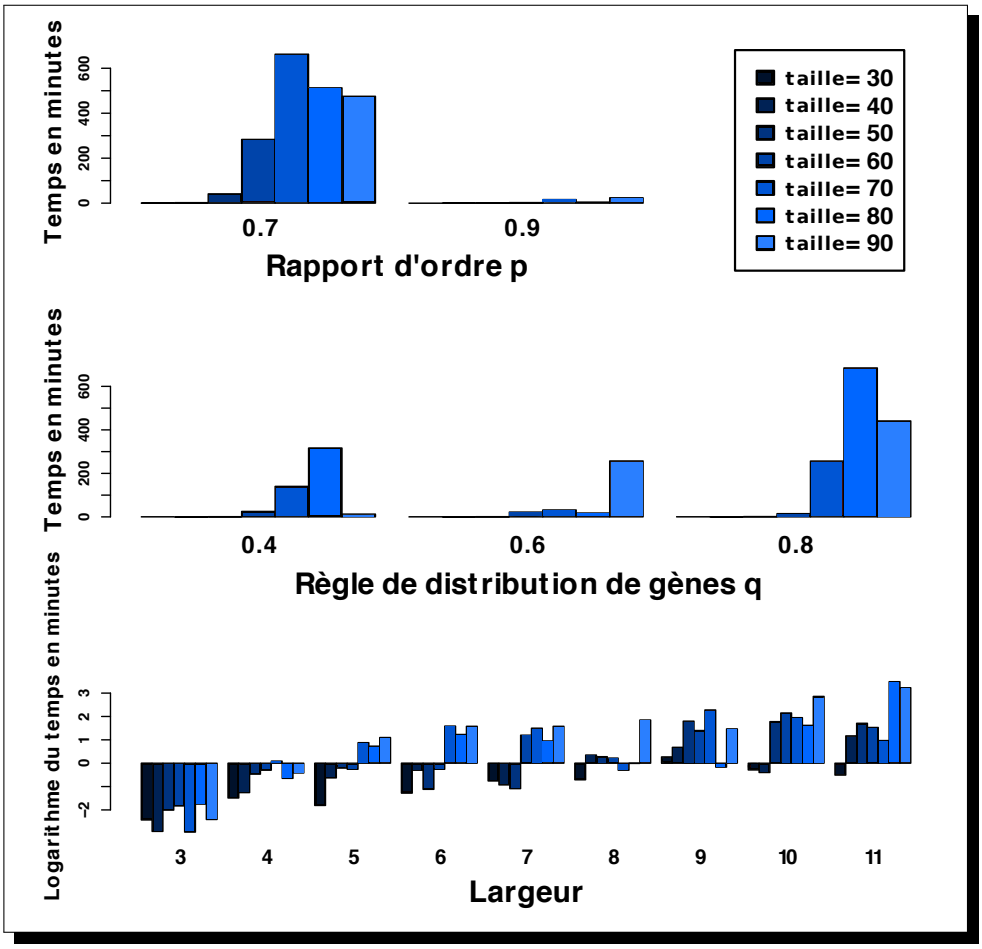


FIGURE 4.14 – **L'influence de p, de q et de la largeur sur le temps de calcul de `Adjacence-1OP`.** Le temps moyen est indiqué sous forme d'histogramme en fonction de la taille des génomes.

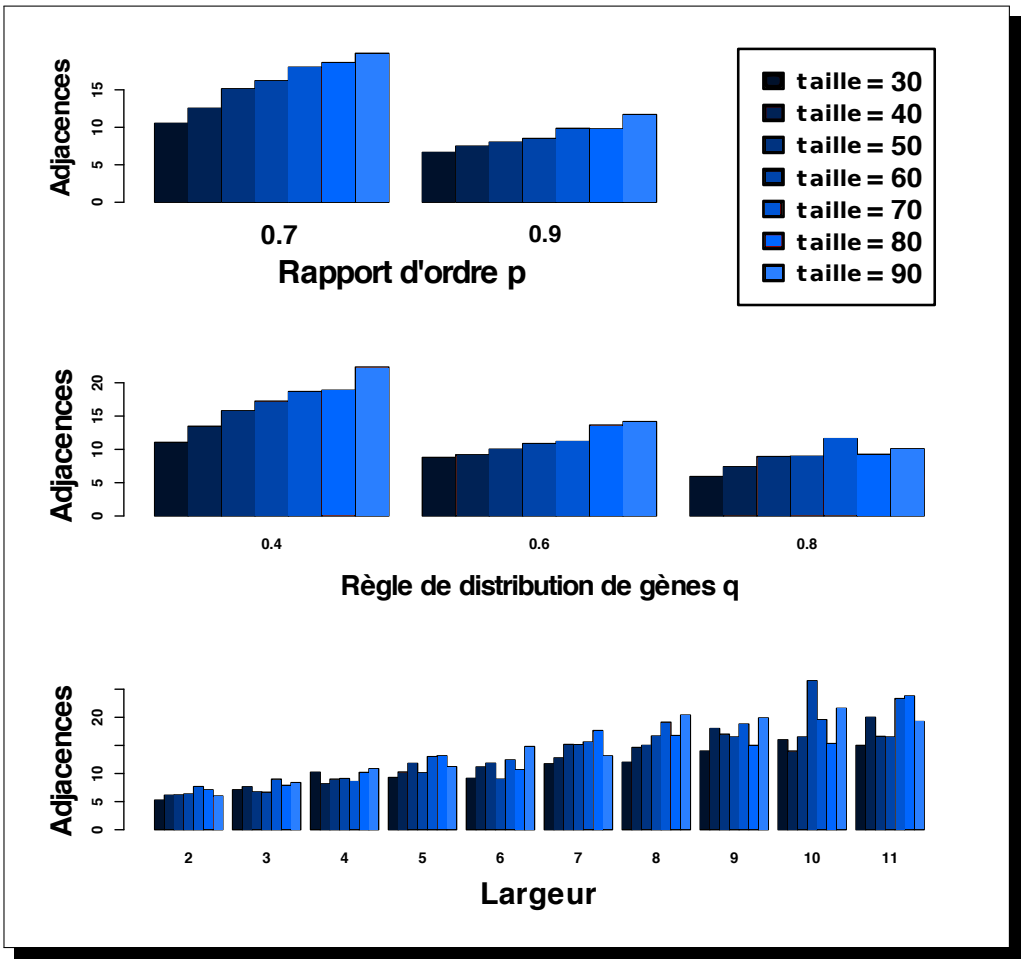


FIGURE 4.15 – **L'influence de p, de q et de la largeur sur le nombre maximal d'adjacences obtenu par** `Adjacence-1OP`**.** La valeur moyenne du nombre d'adjacences maximal est indiquée sous forme d'histogramme en fonction de la taille des génomes.

Génomes					Résultats de Adjacence-2OP			
Taille	p	q	Critère	Largeur	Largeur	Adjacence	Intervalle commun	Temps
30	0,7	0,8	Somme	2196	832	947	4698	2 jours 12
			Moyenne	6,4	4,6	10,5	52,2	40 min 24
			Min	3	2	3	39	< 1 s
			Max	11	14	18	76	1 jour 8
			Nb-res	171	90	90	90	90
*	0,9	*	Somme	11888	10053	12573	101183	38 jours 7
			Moyenne	3,17	3,05	7,6	62,22	33 min
			Min	1	1	0	35	< 1 s
			Max	7	7	21	121	6 jours 14
			Nb-res	1881	1668	1668	1668	1668
30	0,9	0,4	Somme	756	749	1257	8744	4 h 5
			Moyenne	2,2	2,2	7,4	51,4	1 min 26
			Min	1	1	1	37	< 1 s
			Max	4	4	16	121	2 h 12
			Nb-res	171	170	170	170	170
30	0,9	0,6	Somme	1008	954	1213	7896	6 jours 19
			Moyenne	2,9	2,9	7,4	47,8	59 min 14
			Min	2	2	1	37	< 1 s
			Max	7	7	19	98	4 jours 1
			Nb-res	171	165	165	165	165
30	0,9	0,8	Somme	900	900	828	7129	40 min
			Moyenne	2,6	2,6	4,8	41,2	14 s
			Min	2	2	1	35	< 1 s
			Max	5	5	11	53	18 min 53
			Nb-res	171	171	171	171	171
40	0,9	0,4	Somme	1134	1046	1705	10647	18 jours
			Moyenne	3,3	3,2	10,5	65,7	2 h 40
			Min	2	2	4	51	< 1 s
			Max	6	6	21	102	6 jours 14
			Nb-res	171	162	162	162	162
40	0,9	0,6	Somme	936	912	1078	9185	6 h 3
			Moyenne	2,7	2,7	6,4	54,7	2 min 9
			Min	1	1	0	46	< 1 s
			Max	5	5	15	73	3 h 34
			Nb-res	171	168	168	168	168
40	0,9	0,8	Somme	1098	1050	1170	9045	1 jour 3
			Moyenne	3,2	3,2	7	54,8	9 min 54
			Min	2	2	1	47	< 1 s
			Max	5	5	15	68	5 h 48
			Nb-res	171	165	165	165	165
50	0,9	0,6	Somme	1116	893	1083	9432	1 jour 4
			Moyenne	3,3	3,1	7,6	66	12 min
			Min	2	2	1	57	< 1 s
			Max	6	6	19	84	8 h
			Nb-res	171	143	143	143	143
50	0,9	0,8	Somme	1116	940	1111	9798	5 jours 18
			Moyenne	3,3	3,1	7,4	64,9	55 min
			Min	2	2	1	57	< 1 s
			Max	6	6	17	80	3 jours 9
			Nb-res	171	151	151	151	151
60	0,9	0,6	Somme	1124	934	1107	10848	1 jour 17
			Moyenne	3,6	3,3	7,7	75,9	17 min 9
			Min	2	2	1	67	< 1 s
			Max	7	6	18	93	8 h 43
			Nb-res	171	143	143	143	143
60	0,9	0,8	Somme	1314	1028	1246	10436	2 jours 8
			Moyenne	3,8	3,8	9,2	76,7	25 min
			Min	2	2	1	67	< 1 s
			Max	6	6	18	91	12 h 14
			Nb-res	171	136	136	136	136
70	0,9	0,8	Somme	1386	647	775	8023	22 h 30
			Moyenne	4	3,4	8,2	85,3	14 min 21
			Min	2	2	2	78	1 s
			Max	7	6	19	98	4 h 27
			Nb-res	171	94	94	94	94

TABLE 4.4 – **Résumé des résultats obtenus par** `Adjacence-2OP` La série $\{*; 0,9; *\}$ correspond aux génomes des 11 triplets étudiés pour lequel $p = 0,9$. La colonne `Intervalle commun` correspond au nombre d'intervalles communs pour les extensions linéaires obtenues par la maximisation du nombre d'ajdacences. Ce tableau est construit d'une façon similaire à celui du Tableau 4.2 (voir Section 4.4.1).

4.4.3 Comparaisons des deux mesures

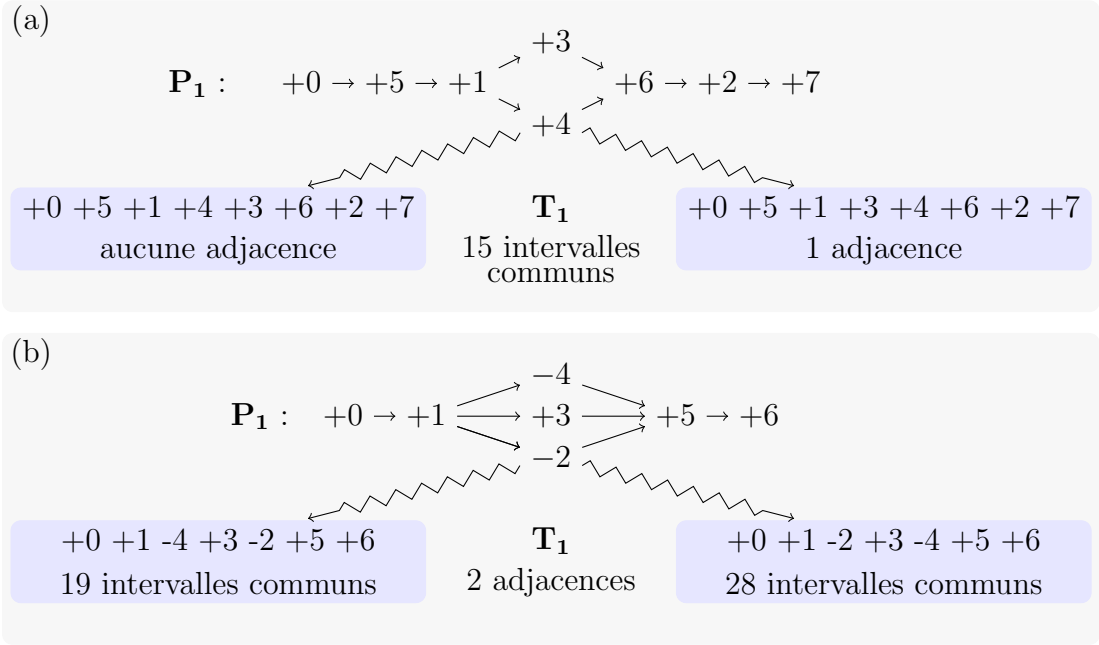


FIGURE 4.16 – **La maximisation d'une mesure peut être obtenue par différentes extensions. (a)** Deux extensions linéaires de P_1 peuvent donner le nombre maximum d'intervalles communs optimal (15) mais donner un nombre d'adjacences différents (0 ou 1). **(b)** Au moins deux extensions linéaires de P_1 peuvent donner le nombre maximum d'adjacences optimal (2) mais donner un nombre d'intervalles communs différents (19 ou 28).

Génomes					Résultats de Adjacence-1OP	
Taille	p	q	Largeur	Critère	Adjacence	Intervalle commun
*	*	*	*	Moyenne	12,2	84,6
				Min	0	37
				Max	39	211
				Nb-res	778	778
30	*	*	*	Moyenne	8,6	56,4
				Min	1	37
				Max	19	140
				Nb-res	114	114
40	*	*	*	Moyenne	10	61,4
				Min	0	43
				Max	25	113
				Nb-res	114	114
50	*	*	*	Moyenne	11,6	74,7
				Min	1	54
				Max	34	153
				Nb-res	114	114
60	*	*	*	Moyenne	12,3	84
				Min	1	64
				Max	30	463
				Nb-res	114	114
70	*	*	*	Moyenne	13,8	97
				Min	2	75
				Max	39	169
				Nb-res	112	112
80	*	*	*	Moyenne	13,9	105,9
				Min	1	84
				Max	33	160
				Nb-res	106	106
90	*	*	*	Moyenne	15,4	117
				Min	2	95
				Max	38	211
				Nb-res	104	104
*	0,7	*	*	Moyenne	15,7	90
				Min	2	38
				Max	39	211
				Nb-res	379	379
*	0,9	*	*	Moyenne	8,8	79,2
				Min	0	37
				Max	30	153
				Nb-res	399	399
*	*	0,4	*	Moyenne	16,6	99,3
				Min	4	45
				Max	39	211
				Nb-res	257	257
*	*	0,6	*	Moyenne	11,1	80,1
				Min	1	39
				Max	31	141
				Nb-res	263	263
*	*	0,8	*	Moyenne	8,8	74,4
				Min	0	37
				Max	31	127
				Nb-res	258	258
*	*	*	<11	Moyenne	10,8	81
				Min	0	37
				Max	30	163
				Nb-res	696	696
*	*	*	>10	Moyenne	23,6	120,4
				Min	12	73
				Max	39	211
				Nb-res	82	59

TABLE 4.5 – **Comparaison des mesures d'intervalles communs et d'adjacences.** Une série de 5 lignes correspond aux 19 génomes suivant 4 mêmes critères (taille, p, q et largeur) et dont un résultat a été obtenu par `Adjacence-1OP`.

4.4.4 Gramene

Le site Gramene [62] (http ://www.gramene.org) offre accés, entre autre, à une base de données de génomes. Certains de ces génomes sont partiellement ordonnés. Pour un même chromosome, plusieurs cartes physiques peuvent être stockées, comme par exemple pour le sorgho (voir Figure 4.17).

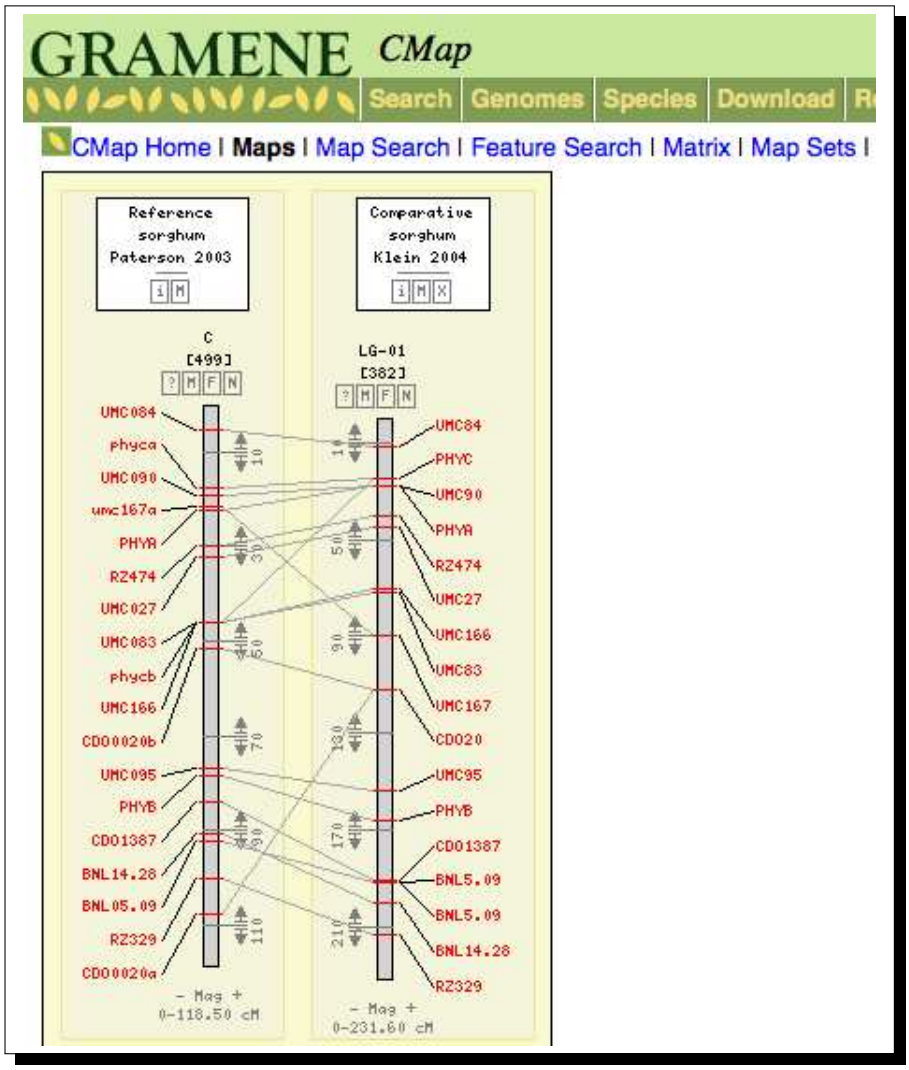


FIGURE 4.17 – **Gramene : Chromosome de sorgho.** Deux cartes physiques d'un chromosome de sorgho ("sorghum" en anglais) - http ://www.gramene.org/db/cmap/.

Conclusion

Nous avons présenté dans ce manuscrit de nouveaux résultats dans le domaine de la génomique comparative.

Dans un premier temps, nous avons étudié la comparaison de génomes possédant des gènes dupliqués.

En étudiant les problèmes MCEN, MCMN et MCIN,

- nous avons montré que minimiser le nombre de points de cassure, pour les modèles exemplaire et intermédiaire, n'est pas un problème approximable même si aucun gène n'est présent plus de deux fois dans un des deux génomes ;
- nous avons montré que maximiser le nombre d'adjacences, pour le modèle exemplaire, n'est pas un problème approximable même si aucun gène n'est présent plus de deux fois dans un des deux génomes ;
- nous avons proposé un algorithme exponentiel permettant de décider s'il existe un couplage exemplaire entre deux génomes tel que le nombre de points de cassure est nul, dans le cas où chaque gène est présent au plus deux fois dans chaque génome ;
- nous avons prouvé que décider s'il existe un couplage maximal entre deux génomes tel que le nombre de points de cassure est nul est résoluble en temps polynomial.

Nous avons ensuite proposé des algorithmes exacts et des heuristiques permettant, soit de maximiser le nombre d'adjacences, soit de minimiser le nombre de points de cassure sous trois modèles (exemplaire, intermédiaire et maximum). Nous avons évalué nos algorithmes sur 12 génomes de γ-protéobactéries. Nous avons obtenu, dans la plupart des cas, les résultats exacts et nous avons pu montrer que nos heuristiques sont rapides et donnent de bon résultats. En particulier, l'heuristique hybride pour le paramètre $k = 3$ semble être un bon compromis entre efficacité et rapidité. Ainsi, nous avons pu évaluer les trois modèles, en particulier le modèle intermédiaire, en fonction de ces deux mesures (nombre d'adjacences et nombre de points de cassure).

Les modèles exemplaire et maximum sont, d'un point de vue biologique, des modèles extrêmes et non réalistes. C'est pourquoi nous avons introduit le modèle intermédiaire : c'est une première tentative pour s'éloigner des modèles théoriques que sont les modèles

exemplaire et maximum. Les temps de calculs sont clairement supérieurs pour ce nouveau modèle. De plus les mesures étudiées peuvent être, quasiment, obtenues de façon indépendante *via* les modèles exemplaire et maximum. En effet, le nombre de points de cassure minimal pour le modèle intermédiaire est identique au nombre minimal optenu pour le modèle exemplaire ; le nombre d'adjacences maximal pour le modèle maximum est, en moyenne, 99% le nombre d'adjacences maximal pour le modèle intermédiaire. Pourtant, nous pensons que les degrés de liberté autorisés par le modèle intermédiaire donnent un avantage certain pour des applications pratiques. Tout d'abord, en optimisant une mesure, il donne de meilleurs résultats pour la deuxième mesure. De plus, il permet d'obtenir un couplage qui décrit un génome vraisemblable pour l'ancêtre commun des deux génomes étudiés.

Dans un deuxième temps, nous avons étudié les génomes dont l'ordre des gènes est partiellement connu. Ces génomes sont représentés par des ordres partiels. Nous avons proposé des programmes calculant, de façon exacte, des extensions linéaires de ces ordres grâce à une optimisation de mesure de similarité :

- le programme `IC-1OP` prend en entrée un ordre partiel et un ordre total et renvoie, parmi toutes les extensions possibles, une de celles qui permet d'avoir le nombre maximal d'intervalles communs entre une extension linéaire de l'ordre partiel et l'ordre total ;
- le programme `Adjacence-1OP` prend en entrée un ordre partiel et un ordre total et renvoie, parmi toutes les extensions possibles, une de celles qui permet d'avoir le nombre maximal d'adjacences entre une extension linéaire de l'ordre partiel et l'ordre total ;
- le programme `Adjacence-2OP` prend en entrée deux ordres partiels et renvoie parmi toutes les extensions linéaires possibles celles qui permettent d'avoir le nombre d'adjacences maximal.

Ces trois programmes ont été évalués à l'aide de génomes partiellement ordonnés simulés, nous permettant ainsi d'étudier l'influence des paramètres inhérents aux ordres partiels. Nous avons alors pu noter que la largeur des ordres partiels est le paramètre qui influe le plus sur le temps de calcul alors que la similarité entre les deux génomes n'affecte pas ce temps.

Le travail à effectuer dans un futur proche est la réalisation d'un plus grand nombre de tests pour le programme `IC-1OP` afin de posséder suffisamment de résultats pour une évaluation plus complète. De plus, et en particulier pour la mesure d'adjacences, nous pouvons maintenant proposer et évaluer des heuristiques.

À travers les problèmes exposés dans ce manuscrit, nous avons présenté une méthode générique permettant de comparer des génomes. Cette méthode utilise une modélisation par programmation linéaire à variables booléennes. Rappelons que le but est d'utili-

ser des solveurs efficaces pour la résolution de ce type de programme. Nous avons utilisé deux solveurs : `MiniSat+` et `CPLEX`. Par notre étude, nous avons pu observer que ces deux solveurs sont efficaces pour des problèmes différents. En effet, pour les programmes `Couplage-Maximum-Adjacences`, et les deux programmes équivalents pour les modèles exemplaire et intermédiaire, le solveur `CPLEX` résout plus de cas et ce plus rapidement que le solveur `MiniSat+`. Inversement, pour la résolution des programmes `IC-1PO`, `Adjacance-1PO` et `Adjacence-2OP`, le solveur `MiniSat+` est nettement plus performant.

Même si nos problèmes sont exponentiels, et donc voués à subir une explosion combinatoire lors de leur résolution, il serait bon de tester d'autres solveurs afin d'obtenir de nouveaux résultats, en particulier pour les problèmes d'ordres partiels. En effet, notre approche exacte, par programmation linéaire à variables booléennes, n'a pas pour but de fournir un outil pratique (même s'il l'est pour de petits génomes) mais plutôt un outil permettant d'une part la mise au point d'heuristiques efficaces et rapides, d'autre part l'étude des mesures et des modèles de la génomique comparative.

Dans le même ordre d'idée, les programmes proposés peuvent être améliorés par l'ajout de régles qui permettent, soit de diminuer le nombre de variables, soit de commencer à résoudre le problème (débuter un couplage ou une extension).

L'intérêt de cette méthode générique est qu'elle peut s'adapter pour la résolution de problèmes similaires : optimisation d'autres mesures ou plus directement résolution de `MICL-2OP`. Nous l'avons par ailleurs utilisée pour maximiser le nombre MAD entre deux génomes possédant des gènes dupliqués. Le travail préliminaire n'a actuellement pas donné de résultats probants (les temps de résolution sont trop excessifs).

De plus, mais cela semble plus difficile, nous pourrions définir de nouveaux programmes linéaires à variables booléennes permettant de comparer des génomes avec duplication dont l'ordre des gènes n'est que partiellement connu.

À noter aussi que nous pourrions facilement adapter nos programmes afin de pouvoir étudier les génomes circulaires. Par ailleurs, une étude intéressante serait de prendre en compte plusieurs chromosomes simultanément.

D'autre part, comme nous l'avons vu pour tous ces problèmes, plusieurs solutions sont possibles. Ceci nous pousse à vouloir donner la possibilité de choisir parmis toutes ces solutions, optimisant une même mesure, celle qui optimise au mieux une deuxième mesure. Comme nous l'avons vu, cela ne semble pas compliqué à réaliser mais plutôt coûteux en temps.

Nous avons donc proposé des programmes efficaces permettant d'obtenir des mesures de (dis)similarité entre deux génomes. Ainsi nous avons pu évaluer différentes caractéristiques inhérentes aux problèmes posés (modèles de couplage choisis, mesures utilisées) et aux génomes donnés en instances (taille, largeur, ressemblance). Il s'agit maintenant, pour poursuivre cette étude, d'utiliser ces outils sur des données réelles (en particulier pour les génomes partiellement ordonnés). Ainsi, nous pourrions évaluer l'intérêt

de chaque mesure et de chaque modèle. En particulier, il serait intéressant d'étudier les arbres phylogénétiques que nous pouvons obtenir à l'aide des mesures calculées.

Liste des tableaux

Table des figures

Index

Bibliographie

[1] A. Ahmadian, M. Ehn, and S. Hober. Pyrosequencing : history, biochemistry and future. *Clinica chimica acta*, 363 :83–94, 2006. 25

[2] S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. A pseudo-boolean programming approach for computing the breakpoint distance between two genomes with duplicate genes. In *Proceedings of the 5th RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG)*, volume 4751 of *Lecture Notes in Computer Science*, pages 16–29. Springer, September 2007. 38, 72, 82

[3] S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. Efficient tools for computing the number of breakpoints and the number of adjacencies between two genomes with duplicate genes. *Journal of Computational Biology*, 15(8) :1093–1115, 2008. 82

[4] S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. On the approximability of comparing genomes with duplicates. *Journal of Graph Algorithms and Applications*, 13(1) :19–53, 2008. 36, 38, 39, 73

[5] S. Angibaud, G. Fertin, I. Rusu, and S. Vialette. How pseudo-boolean programming can help genome rearrangement distance computation. In G. Bourque and N. El-Mabrouk, editors, *Proceedings of the Fourth RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG)*, volume 4205, pages 75–86, Canada, September 2006. Springer. 39

[6] S. Angibaud, G. Fertin, I. Rusu, and S. Vialette. A general framework for computing rearrangement distances between genomes with duplicates. *Journal of Computational Biology*, 14(4) :379–393, 2007. 84, 93, 96, 99, 109

[7] Sebastien Angibaud, Guillaume Fertin, Annelyse Thevenin, and Stephane Vialette. Pseudo-boolean programming for partially ordered genomes. In Springer, editor, *Comparative Genomics*, LNBI, Budapest, Hungary, September 2009. Springer-Verlag. 126

[8] S. Aubourg, V. Brunaud, C. Bruyère, M. Cock, R. Cooke, A. Cottet, A. Couloux, P. Déhais, G. Deléage, L. Drouard, A. Duclert, M. Echeverria, A. Eschbach, D. Falconet, G. Filippi, C. Gaspin, C. Geourjon, J.-M. Grienenberger, B. Guermann,

G. Houlné, E. Jamet, F. Lechauve, O. Leleu, P. Leroy, r. Mache, C. Meyer, H. Nedjari, I. Negrutiu, V. Orsini, E. Peyretaillade, C. Pommier, J. Raes, J.-L. Risler, S. Rivière, S. Rombauts, P. Rouzé, M. Schneider, P. Schwob, I. Small, G. Soumayet-Kampetenga, D. Stankovski, C. Toffano, M. Tognolli, M.Caboche, and A. Lecharny. The GENEFARM project : structural and functional annotation of arabidopsis gene and protein families by a network of experts. *Nucleic Acids Research*, 33 :641–646, 2005. 23

[9] D.A. Bader, B.M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8 :483–491, 2001. 30

[10] V. Bafna and P.A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2) :224–240, 1998. 30

[11] A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. In *Proceedings of the 9th International Computing and Combinatorics Conference (COCOON '03)*, volume 2697 of *Lecture Notes in Computer Science*, pages 68–79, 2003. 31, 32

[12] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *ESA '02 : Proceedings of the 10th Annual European Symposium on Algorithms*, pages 200–210, London, UK, 2002. Springer-Verlag. 30

[13] P. Berman and M. Karpinski. On some tighter inapproximability results (extended abstract). In *Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, page 705. Springer Berlin / Heidelberg, 1999. 30

[14] D. Le Berre and A. Parrain. À propos de l'extension d'un solveur SAT pour traiter des contraintes pseudo-booléennes. In *Troisième Journées Francophones de Programmation par Contraintes (JFPC07)*, pages 38–47, 2007. 65

[15] G. Blin, E. Blaisy, D. Hermelinz, P. Guillon, M. Blanchette, and N. El-Mabrouk. Gene maps linearization using genomic rearrangement distances. *Journal of Computational Biology*, 14(4) :394–407, 2007. 39, 44, 45, 46, 47, 128, 145, 149

[16] G. Blin, C. Chauve, and G. Fertin. The breakpoint distance for signed sequences. In *Proceedings of the first Algorithms and Computational Methods for Biochemical and Evolutionary Networks (CompBioNets)*, pages 3–16. KCL publications, 2004. 38, 72

[17] G. Blin, C. Chauve, and G. Fertin. Genes order and phylogenetic reconstruction : Application to γ-proteobacteria. In *Proceedings of the 3rd RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG'05)*, volume 3678 of *Lecture Notes in Comp. Sci.*, pages 11–20, 2005. 72, 96, 97

[18] G. Blin, C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Comparing genomes with duplications : A computational complexity point of view. *ACM/IEEE Transactions Computational Biology and Bioinformatics*, 4(4) :4523–534, 2007. 72

[19] G. Blin, G. Fertin, F. Sikora, and S. Vialette. The exemplar breakpoint distance for non-trivial genomes cannot be approximated. In S. Das and R. Uehara, editors, *3rd Annual Workshop on Algorithms and Computation (WALCOM'09)*, volume 5431, pages 357–368, Kolkata, India, 2009. Springer-Verlag. 73, 76

[20] M. Bogard, N. Ameziane, and J. Lamoril. Microarray d'ADN et profils d'expression des gènes : Première partie : concept, fabrication et mise en œuvre. *Immunoanalyse et Biologie Spécialisée*, 23(2) :71–88, April 2008. 25

[21] D. Bryant. The complexity of calculating exemplar distances. In *Comparative Genomics : Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pages 207–212. Kluwer, 2000. 34, 36, 38, 71, 72, 84

[22] A. Caprara. Sorting by reversals is difficult. In *Proceedings of the first annual international conference on Computational molecular biology*, pages 75–83, Santa Fe, New Mexico, United States, 1997. Association for Computing Machinery. 30

[23] M.-S. Chang and F.-H. Wang. Efficient algorithms for the maximum weight clique and maximum weight independent set problems on permutation graphs. *Information Processing Letters*, 43(6) :293–295, 1992. 81

[24] The French-Italian Public Consortium For Grapevine Genome Characterization. The grapevine genome sequence suggests ancestral hexaploidization in major angiosperm phyla. *Nature*, 449 :463–468, September 2007. 125

[25] C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Genomes containing duplicates are hard to compare. In *Computational Science – ICCS 2006*, volume 3992 of *Lecture Notes in Computer Science*, pages 783–790. Springer Berlin / Heidelberg, 2006. 39

[26] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions Comput. Biol. Bioinformatics*, 2(4) :302–315, 2005. 36

[27] Z. Chen, B. Fu, J. Xu, B. Yang, Z. Zhao, and B. Zhu. Non-breaking similarity of genomes with gene repetitions. In *Proceedings of the 18th Annual Symposium on Combinatorial Pattern Matching (CPM'07)*, volume 4580 of *Lecture Notes in Computer Science*, pages 119–130, 2007. 39, 84

[28] Z. Chen, B. Fu, and B. Zhu. The approximability of the exemplar breakpoint distance problem. In *Proceedings of the Second International Conference on Aspects in Information and Management 2006*, volume 4041 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2006. 36, 38, 73, 74, 79, 110

[29] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431 :931–945, October 2004. 125

[30] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971. 57

[31] G.B. Dantzig. Origins of the simplex method. In S.G. Nash, editor, *A History of Scientific Computing*, pages 141–151, 1990. 64

[32] B. Davey and H. Priestley. *Introduction to Lattices and order*. Cambridge University Press, 2nd éd. edition, 2002. 41

[33] J. Delgado, I. Lynce, and V. Manquinho. Computing the Summed Adjacency Disruption number between two genomes with duplicate genes using pseudo-boolean optimization. In *RECOMB-CG*, 2009. 112

[34] Z. Dias and J. Meidanis. Sorting by prefix transpositions. In *String Processing and Information Retrieval*, volume 2476 of *Lecture Notes in Computer Science*, pages 463–468. Springer Berlin / Heidelberg, 2002. 30

[35] T. Dobzansky and A.H. Sturtevant. Inversions in the third chromosome of wild races of drosophila pseudoobscura, and their use in the study of the history of the species. In *Proceedings of the National Academy of Sciences*, USA, 1936. 28, 29

[36] T. Dobzansky and A.H. Sturtevant. Inversions in the chromosomes of drosophila pseudoobscura. *Genetics*, 23 :28–64, 1938. 28, 29

[37] R. Drmanac, I. Labat, I. Brukner, and R. Crkvenjakov. Sequencing of megabase plus DNA by hybridization : theory of the method. *Genomics*, 4 :114–128, 1989. 25

[38] F. Droesbeke, M. Hallin, and C.L. Lefevre. *Programmation linéaire par l'exemple*. Ellipses, 1986. 62

[39] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT translating pseudo-boolean constraints into SAT translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2 :1–25, 2006. 65, 67, 99, 130, 145

[40] I. Elias and T. Hartman. A 1.375-approximation algorithm for sorting by transpositions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4) :369–379, 2006. 30

[41] H. Fernau. Parameterized algorithmics : A graph-theoretic approach. Habilitationsschrift, 2005. 78

[42] G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. MIT Press, August 2009. 29, 36, 44

[43] Z. Fu and T. Jiang. Computing the breakpoint distance between partially ordered genomes. *Journal of Bioinformatics and Computational*, 17(16), October 2006. 44, 45

[44] M.R. Garey and D.S. Johnson. *Computers and Intractability : a guide to the theory of NP-completeness*. W.H. Freeman, San Franciso, 1979. 53, 55, 64, 74, 141

[45] W. Gilbert and A. Maxam. The nucleotide sequence of the lac operator. *Proceedings of the National Academy of Sciences*, 70 :3581–3584, 1973. 24

[46] A. Goffeau, B.G. Barrell, H. Bussey, R.W. Davis, B. Dujon, H. Feldmann, F. Galibert, J.D. Hoheisel, C .Jacq, M. Johnston, E.J. Louis, H.W. Mewes, Y. Murakami, P. Philippsen, H. Tettelin, and S.G. Oliver. Life with 6000 genes. *Science*, 274(5287) :563–567, 1996. 26, 34

[47] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980. 80

[48] R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5) :275–278, 1958. 64, 65

[49] R.E. Green, A-S. Malaspinas, J. Krause, A. W. Briggs, P.L.F. Johnson, C. Uhler, M. Meyer, J.M. Good, T. Maricic, U. Stenzel, K. Prüfer, M. Siebauer, H.A. Burbano, M. Ronan, J. M. Rothberg, M. Egholm, P. Rudan, D. Brajković, ěl. Kućan, I. Gušić, M. Wikström, L. Laakkonen, J. Kelso, M. Slatkin, and S. Pääbo. A complete neandertal mitochondrial genome sequence determined by high-throughput sequencing. *Cell*, 134(3) :416–426, 2008. 125

[50] S. Hannenhalli and P.A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Foundations of Computer Science, 1995. Proceedings, 36th Annual Symposium on Foundation of Computer Science*, pages 581–592, New York, October 1995. 30

[51] S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip : polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46 :1–27, January 1999. 30

[52] Arabidopsis Genome initiative. Analysis of the genome sequence of the flowering plant arabidopsis thaliana. *Nature*, 408(6814) :796–815, 2000. 26, 34

[53] B.N. Jackson, S. Aluru, and P.S. Schnable. Consensus genetic maps : a graph theoretic approach. In *Computational Systems Bioinformatics Conference*, pages 35–43, August 2005. 41

[54] B.N. Jackson, P.S. Schnable, and S. Aluru. Consensus genetic maps as median orders from inconsistent sources. *Computational Biology and Bioinformatics*, 5(2) :161–171, 2008. 41

[55] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4 :373–395, 1984. 64

[56] J.D. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13 :180–210, 1995. 31

[57] L. Khachyan. Polynomial algorithm in linear programming. *Doklady Akademii Nauk U.S.S.R.*, 244 :1093–1096, 1979. 64

[58] J. Lamoril, N. Ameziane, J.C. Deybach, P. Bouizegarène, and M. Bogard. Les techniques de séquençage de l'ADN : une révolution en marche. première partie. *Immunoanalyse et Biologie Spécialisée*, 23(5) :260–279, 2008. 24

[59] E. Lerat, V. Daubin, and N.A. Moran. From gene tree to organismal phylogeny in prokaryotes : the case of γ-proteobacteria. *PLoS Biology*, 1(1) :101–109, 2003. 47, 96

[60] E. Lerat, V. Daubin, H. Ochman, and N.A.A. Moran. Evolutionary origins of genomic repertoires in bacteria. *PLoS Biology*, 3(5), April 2005. 96

[61] W. Li, Z. Gu, H. Wang, and A. Nekrutenko. Evolutionary analyses of the human genome. *Nature*, 409 :847–849, February 2001. 34

[62] C. Liang, P. Jaiswal, C. Hebbard, S. Avraham, E.S. Buckler, T. Casstevens, B. Hurwitz, S. McCouch J. Ni, A. Pujar, D. Ravenscroft, L. Ren, W. Spooner, I. Tecle, J. Thomason, C.W. Tung, X. Wei, I. Yap, K. Youens-Clark, D. Ware, and L. Stein. Gramene : a growing plant comparative genomics resource. *Nucleic Acids Research*, 36 :D947–D953, November 2007. 157, 164

[63] A. Louis, E. Ollivier, J.C Aude, and J.L. Risler. Massive sequence comparisons as a help in annotating genomic sequences. *Genome Research*, 11(7) :1296–1303, July 2001. 23

[64] V.M. Manquinho and J. Marques-Silva. On using cutting planes in pseudo-boolean optimization. *Satisfiability, Boolean Modeling and Computation*, 2 :209–219, March 2006. 65

[65] M. Marron, K.M. Swenson, and B.M.E. Moret. Genomic distances under deletions and insertions. *Theoretical Computer Science*, 325(3) :347–360, 2004. 93

[66] C. Mathé, M.F. Sagot, T. Schiex, and P. Rouzé. Survey and summary : Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research*, 30(19) :4103–4117, 2002. 26

[67] M. Nakano, N. Nakai, H. Kurita, J. Komatsu, K. Takashima, and S. Katsura et al. Single-molecule reverse transcription polymerase chain reaction using water-in-oil emulsion. *Journal of Bioscience and Bioengineering*, 99 :293–295, 2005. 25

[68] C.T. Nguyen, Y.C. Tay, and L. Zhang. Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics*, 21(10) :2171–2176, May 2005. 38

[69] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Prentice-Hall, 1982. 60

[70] M. Picardeau, D.M. Bulach, C. Bouchier, R.L. Zuerner, N Zidane, P.J. Wilson, S. Creno, E.S. Kuczek, S. Bommezzadri, J.C. Davis, A. McGrath, M.J. Johnson, C. Boursaux-Eude, T. Seemann, Z. Rouy, R.L. Coppel, J.I. Rood, A. Lajus, J.K. Davies, C. Médigue, and B. Adler. Genome sequence of the saprophyte leptospira biflexa provides insights into the evolution of leptospira and the pathogenesis of leptospirosis. *PLoS ONE*, 3(2) :e1607, 2008. 125

[71] F. Sanger, G.M.Air, B.G. Barrell, N.L. Brown, A.R. Coulson, C.A. Fiddes, C.A.Hutchison, P.M. Slocombe, and M. Smith. Nucleotide sequence of bacteriophage phi x174 DNA. *Nature*, 265(5596) :687–695, February 1977. 26

[72] F. Sanger, S. Nicklen, and A.R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12) :5463–5467, December 1977. 24

[73] D. Sankoff. Mechanismes of genome evolution : models and inference. *Bulletin of the International Statistical Institute*, 47 :461–475, 1989. 30

[74] D. Sankoff. Edit distance for genome comparison based on non-local operations. In *Combinatorial Pattern Matching*, volume 644/1992 of *Lecture Notes in Computer Science*, pages 121–135. Springer Berlin / Heidelberg, 1992. 36, 38

[75] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11) :909–917, November 1999. 34, 35, 72

[76] D. Sankoff and M. Blanchette. The median problem for breakpoints in comparative genomics. In T. Jiang and D.T. Lee, editors, *Proceedings of the Third Internaional Computing and Combinatorics Conference (COCOON)*, volume 1276 of *Lecture Notes in Computer Science*, pages 251–263. Springer-Verlag, August 1997. 31

[77] D. Sankoff and L. Haque. Power boosts for cluster tests. In *Proceedings of the 3rd RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG'05)*, volume 3678 of *Lecture Notes in Computer Science*, pages 11–20. Springer Berlin / Heidelberg, 2005. 32

[78] D. Sankoff, C. Zheng, and A. Lenert. Reversals of fortune. In *Proceedings of the 3rd RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG'05)*, volume 3678 of *Lecture Notes in Computer Science*, pages 131–141. Springer Berlin / Heidelberg, 2005. 42, 126

[79] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998. 84, 130

[80] H.M. Sheini and K.A. Sakallah. Pueblo : A hybrid pseudo-boolean SAT solver. *Satisfiability, Boolean Modeling and Computation*, 2 :155–179, 2006. 65

[81] J. Tang and B.M.E Moret. Phylogenetic reconstruction from gene-rearrangement data with unequal gene content. In *Proceedings of Workshop on Algorithms and Data Structures (WADS'03)*, volume 2748 of *Lecture Notes in Computer Science*, pages 37–46, 2003. 72

[82] E. Tannier and M.-F. Sagot. Sorting by reversals in subquadratic time. In *Combinatorial Pattern Matching*, volume 3109 of *Lecture Notes In Computer Science*, pages 1–13. Springer Verlag, 2004. 29

[83] R.L. Tatusov, N.D. Fedorova, J.D. Jackson, A.R. Jacobs, B. Kiryutin, E.V. Koonin, D.M. Krylov, R. Mazumder, S.L. Mekhedov, A.N. Nikolskaya, B.S. Rao, S. Smirnov, A.V. Sverdlov, S. Vasudevan, Y.I. Wolf, J.J. Yin, and D.A. Natale. The COG database : an updated version includes eukaryotes. *BMC Bioinformatics*, 4(41), September 2003. 23

[84] A.M. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, volume 42 of *2*, pages 230–265, 1936. 55

[85] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2) :290–309, 2000. 31

[86] G.A Watterson, W.J. Ewens, T.E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theoretical Biology*, 99 :1–7, 1982. 29, 30

[87] I. V. Yapa, D. Schneiderb, J. Kleinbergc, D. Matthewsd, S. Cartinhourb, and S. R. McCouch. A graph-theoretic approach to comparing and integrating genetic, physical and sequence-based maps. *Genetics*, 165 :2235–2247, December 2003. 41

[88] C. Zheng, A. Lenert, and D. Sankoff. Reversals distance for partially ordered genomes. *Bioinformatics*, 1(1) :1–7, June 2003. 42, 44

[89] C. Zheng, A. Lenert, and D. Sankoff. Reversal distance for partially ordered genomes. *Bioinformatics*, 21 :502–508, 2005. 41, 126