Comparaisons de génomes avec gènes dupliqués : étude théorique et algorithmes

Sébastien Angibaud

UNIVERSITÉ DE NANTES

UFR SCIENCES ET TECHNIQUES

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE MATHÉMATIQUES

2010

Thèse de Doctorat de l'Université de Nantes

Spécialité : INFORMATIQUE

Présentée et soutenue publiquement par

Sébastien ANGIBAUD

le 7 octobre 2009

à l'UFR Sciences et Techniques, Université de Nantes

Comparaisons de génomes avec gènes dupliqués : étude théorique et algorithmes

Jury

Rapporteurs	:	Pr. David SANKOFF, Professeur	University of Ottawa, Canada
		Pr. Thierry LECROQ, Professeur	Université de Rouen
Examinateurs	:	Pr. Jens STOYE, Professeur	Universität Bielefeld, Allemagne
		M. Éric TANNIER, Chargé de recherche - INRIA	Université de Lyon I

Directeur de thèse : Pr. Irena RUSU
Co-directeur de thèse : Pr. Guillaume FERTIN
Laboratoire : LABORATOIRE D'INFORMATIQUE DE NANTES ATLANTIQUE.
2, rue de la Houssinière, BP 92 208 – 44 322 Nantes, CEDEX 3.

N° ED 0366-...

COMPARAISONS DE GÉNOMES AVEC GÈNES DUPLIQUÉS : ÉTUDE THÉORIQUE ET ALGORITHMES

Comparative genomics with duplicated genes : theoretical study and algorithms

Sébastien ANGIBAUD

⋈

favet neptunus eunti

Université de Nantes

Sébastien ANGIBAUD

Comparaisons de génomes avec gènes dupliqués : étude théorique et algorithmes

x+144 p.

Ce document a été préparé avec LaTeX2e et la classe `these-LINA` version v. 1.30 de l'association de jeunes chercheurs en informatique LOGIN, Université de Nantes. La classe `these-LINA` est disponible à l'adresse :

`http://login.irin.sciences.univ-nantes.fr/`

Impression : angibaud_these.tex – 3/2/2010 – 11:03

Révision pour la classe : these-LINA.cls,v 1.30 2005/08/16 17:01:41 mancheron Exp

Résumé

La génomique comparative étudie les similarités et/ou les dissimilarités entre génomes et permet d'établir des relations entre les espèces afin notamment de construire des phylogénies. Elle permet également de mettre en évidence des régions conservées au sein des génomes et de trouver ainsi des ensembles de gènes impliqués dans des processus biologiques conservés au cours de l'évolution. Dans ce mémoire, nous nous intéressons au calcul de mesures entre deux génomes en présence de gènes dupliqués, et plus particulièrement aux mesures à base de points de cassure, d'adjacences, d'intervalles communs et d'intervalles conservés.

Suivant une démarche informatique, nous proposons tout d'abord une étude avancée de la complexité algorithmique des problèmes rencontrés, en prouvant notamment pour la plupart d'entre eux soit leur NP-Complétude soit leur APX-Difficulté. Par la suite, nous exposons plusieurs méthodes de calcul de mesures entre deux génomes, à savoir (i) une approche exacte basée sur une transformation en un problème de contraintes à variables booléennes, (ii) une heuristique et (iii) une méthode hybride qui s'appuie sur la méthode exacte et l'heuristique proposées. Par une étude sur un jeu de données réel, nous montrons les qualités respectives de ces méthodes. Enfin, nous proposons un protocole de calcul des intervalles communs et mettons en évidence, par son utilisation et par un outil de visualisation, l'aspect fonctionnel de certains intervalles communs.

Mots-clés : Génomique comparative, Distances intergénomiques, Complexité algorithmique, Point de cassure, Adjacence, Intervalle commun, Intervalle conservé

Abstract

Comparative genomics consists in studying similarities/dissimilarities between genomes, and can be used to find relations between species in order to compute, for example, phylogenetic trees. Moreover, comparative genomics highlights conserved areas in genomes during the evolution and emphasizes sets of functionnal genes. In this thesis, we are interested in computing measures between two genomes with duplicated genes and, more particularly, we investigate breakpoint, adjacency, common interval and conserved interval based measures.

We first present some theoretical results by proving the NP-Completeness or the APX-Hardness of most of the studied problems. We then propose several methods to compute distances between two genomes: (i) an exact approach based on a transformation into a pseudo-boolean problem (ii) a heuristic (iii) a hybrid method using the exact method and the above mentioned heuristic. We next show their respective qualities on real data. Finally, we propose a general protocol to compute common intervals and highlight their functional aspects.

Keywords: Comparative genomics, Genomic distances, Computational complexity, Breakpoint, Adjacency, Common interval, Conserved interval

Remerciements

Mes premiers remerciements sont sans conteste pour Irena et Guillaume qui m'ont tous les deux accompagné pendant ces trois années de thèse. Merci d'avoir toujours été disponibles et d'avoir suivi avec rigueur mon travail, en perdant parfois beaucoup d'encre rouge.

Je tiens à remercier chaleureusement David Sankoff et Thierry Lecroq d'avoir bien voulu être rapporteurs de ma thèse, ainsi que Jens Stoye et Éric Tannier d'avoir accepté d'être membre de mon jury de thèse. Qu'ils en soient vivement remerciés.

Je remercie également Stéphane Vialette, Annelyse Thévenin et Damien Éveillard pour les différents travaux réalisés ensemble.

Félicitons Julien qui a réussi à me supporter depuis la licence et pour les nombreux plis réalisés ensemble (et pas seulement aux cartes). Merci également à Florian, Malcom et Laurent, pour leur bonne humeur du midi.

À Sandrine qui m'écoute encore quand je parle de problèmes APX-DifficuleS2,

À mes parents, pour m'avoir permis d'arriver jusqu'ici,

Je dis MERCI.

Sommaire

« Ce qui donne à un individu sa valeur génétique, ce n'est pas la qualité propre de ses gènes. C'est qu'il n'a pas la même collection de gènes que les autres. »

— François GROS et François JACOB, Extrait des Sciences de la vie et société (1980).

Introduction

Depuis une vingtaine d'années, les technologies modernes ont révolutionné la biologie en fournissant une multitude d'informations. En particulier, les méthodes de séquençage de l'ADN ont considérablement évolué, offrant ainsi des données à traiter de plus en plus nombreuses. En 1995, le premier génome cellulaire *Haemophilus influenzae* est séquencé, suivi l'année suivante par le premier eucaryote, la levure *Saccharomyces cerevisiae*. Le 14 avril 2003, le consortium international public HGP (Human Genome Project), regroupant dix-huit pays, publie la première carte d'un génome humain. À ce jour, plus de 1000 génomes complets ont été publiés et près de 5000 projets sont en cours, ce qui montre un accroissement extrêmement fort de telles études, poussant même certains à qualifier cette expansion de *Loi de Moore de la génomique*. De plus, de nouvelles techniques, comme *454 Sequencing System* ou encore *Solexa Illumina sequencing*, permettront des séquençages à moindres efforts, de plus en plus rapides et à des coûts bien moins élevés que les méthodes traditionnelles. Devant ce nombre croissant de données, les biologistes se tournent naturellement vers la bio-informatique afin de traiter, par l'outil informatique, les informations de plus en plus nombreuses et volumineuses (par exemple, la séquence d'ADN du génome humain contient plus de trois milliards de paires de bases ce qui correspondant à environ 750 méga-octets de mémoire à stocker). Tout particulièrement, la génomique comparative se révèle être un outil intéressant pour les biologistes, car elle permet une meilleure compréhension du monde vivant et des mécanismes biologiques.

La génomique comparative étudie les relations entre deux ou plusieurs génomes d'espèces ou de souches différentes, et comporte de nombreuses thématiques.

Citons tout d'abord les études portant sur l'histoire des espèces. Parmi ces études, certaines ont pour objectif la recherche de scénarios évolutifs entre deux génomes, ceci en considérant diverses opérations élémentaires (transposition, suppression, insertion, fusion, translocation...). D'autres travaux s'orientent vers l'inférence de génomes ancestraux, avec notamment les problèmes du *génome médian* [BP02, SB97, SSK96] et du *genome halving* [EMNS98]. L'identification des régions hautement conservées au sein des génomes lors de l'évolution (synténie) est également un domaine important de la génomique comparative, et permet l'inférence de fonctions biologiques associées à ces régions conservées. Enfin, de nombreuses recherches portent quant à elles sur le calcul de distance entre deux génomes, comme la *distance de transposition* [BP98], la *distance d'inversion signée* [WEHM82], la *distance d'inversion non signée* [BP93], ou bien encore la *distance d'intervalle conservé* [BS03]. Ces distances, dont une description complète et détaillée est donnée par Fertin et al. dans [FLR$^+$09], peuvent être utilisées afin d'établir des matrices de distances inter-génomiques, ceci dans le but d'inférer une généalogie des espèces, appelée phylogénie.

Avant les années 2000, la plupart des travaux en génomique comparative ont supposé que les génomes ne présentaient pas de gène dupliqué. Or, il est maintenant connu que de nombreux gènes sont présents plusieurs fois au sein d'un même génome. De nombreux problèmes issus de la génomique comparative se sont révélés ardus lorsqu'il a fallu prendre en compte ces gènes dupliqués. Afin de les résoudre, il est donc nécessaire de suivre une démarche informatique appropriée.

La démarche usuelle de résolution d'un problème informatique se décompose en trois parties distinctes : (i) l'étude théorique dudit problème, puis (ii) sa résolution par une méthode adaptée à sa com-

plexité et enfin (iii) une restitution des résultats à l'utilisateur. L'étude théorique vise à caractériser la difficulté d'un problème donné, en classant celui-ci dans une classe de problèmes (**P**, **NP**, **NP**-complet, **APX**-Dur, ...). Cette étude de complexité est nécessaire afin de guider les propositions de résolutions du problème considéré. Suivant l'appartenance de celui-ci à une classe donnée, nous pourrons envisager, par exemple, une résolution par un *algorithme polynomial*, par un *algorithme d'approximation*, par un *algorithme de type FPT* ou bien encore par une *heuristique* si le problème s'avère très difficile. Enfin, la dernière étape consiste à restituer à l'utilisateur les résultats obtenus par les méthodes proposées, ceci afin de répondre à sa problématique et/ou de l'aider à une prise de décision.

Notre travail suit pleinement cette démarche informatique dans le cadre du calcul de distance entre deux génomes. Plus particulièrement, nous nous intéressons dans ce mémoire aux quatre mesures suivantes avec une prise en compte des gènes dupliqués : *nombre de points de cassure*, *nombre d'adjacences*, *nombre d'intervalles communs* et *nombre d'intervalles conservés*. Après une étude théorique de la complexité des problèmes traités, nous proposons plusieurs méthodes de résolution (approche exacte, méthodes approchées et heuristiques). Enfin, nous mettons en évidence des régions fortement conservées au sein des génomes par le calcul des intervalles communs, chacun correspondant à un ensemble de gènes conservés, et offrons un outil de visualisation aidant à l'analyse des intervalles communs.

Ce mémoire est organisé de la manière suivante.

Nous introduisons, dans le Chapitre 1, la notion de génétique et précisons les principaux concepts de la biologie moderne. Le Chapitre 2 introduit ensuite les notations usuelles et les différentes définitions nécessaires à la compréhension de ce rapport. En particulier, nous définissons formellement les mesures étudiées, ainsi que les problèmes abordés dans les chapitres suivants.

Au Chapitre 3, nous présentons les différents travaux théoriques obtenus. Plus précisément, nous prouvons l'**APX**-Difficulté des problèmes du calcul de la distance entre deux génomes pour les mesures *nombre de points de cassure*, *nombre d'intervalles communs* et *nombre d'intervalles conservés* en présence de gènes dupliqués.

Nous proposons par la suite, dans le Chapitre 4, différentes méthodes de calcul de distances intergénomiques pour les quatre mesures étudiées. En premier lieu, nous présentons une méthode exacte basée sur une transformation en un problème de contraintes à variables booléennes. Nous exposons, en second lieu, deux nouvelles approches non exactes et évaluons leurs qualités respectives sur un jeu de données réel.

Dans le Chapitre 5, nous mettons en évidence l'aspect fonctionnel de plusieurs intervalles communs, chacun correspondant à des régions fortement conservées au sein des génomes. Pour cela, nous proposons un protocole complet pour le calcul des intervalles communs entre deux génomes en présence de gènes dupliqués, ainsi qu'un outil de visualisation des intervalles communs afin d'aider à leur analyse. Nous terminons par une étude expérimentale, laquelle suggère qu'il existe un réel intérêt à la recherche des intervalles communs entre deux génomes.

Enfin, nous concluons en rappelant les principaux résultats obtenus et en proposant diverses perspectives de recherche.

CHAPITRE 1

Notions de génétique

L'homme a, depuis de nombreux siècles si ce n'est depuis son apparition, étudié son environnement afin de mieux comprendre les lois qui régissent le monde. L'espèce humaine n'échappe pas à cette étude à travers la médecine, la biologie, la psychologie ou encore la génétique. La génétique désigne littéralement l'étude des *gènes*, définis comme des *séquences d'Acides DésoxyriboNucléiques* (ADN) et portant l'information héréditaire des individus. Par l'étude des gènes, nous cherchons à comprendre les mécanismes régissant l'évolution des gènes et des génomes, ainsi que leur rôle dans le fonctionnement des organismes vivants.

1.1 Historique

Gregor Mendel Ⓦ*

Les premiers travaux sur le sujet sont l'œuvre du moine autrichien Gregor Mendel (1822-1884). En observant la transmission de sept caractéristiques morphologiques du pois à travers plusieurs générations (cf. Figure 1.1), Mendel expose en 1865 les lois de l'hérédité. Ces lois définissent la manière dont les gènes se transmettent de génération en génération. Malgré le peu d'enthousiasme de ses contemporains, ses travaux feront écho plus de quarante ans plus tard, en 1907, lorsque Chappelier traduira ses écrits en français pour le Bulletin Scientifique de la France et de la Belgique.

En 1869, Friedrich Miescher (1844-1895), découvre l'ADN qu'il nomme *nucléine* et montre que cette substance se trouve dans toutes les cellules. Plus tard, Eduard Strasburger (1844-1912) et Oskar Hertwig (1849-1922) montrent en 1880 que le noyau des cellules est le siège de l'hérédité. En 1889, l'Allemand Altmann sépare, à partir de la nucléine, des protéines et une substance acide, l'acide nucléique. Au début du $XX^{\text{ième}}$ siècle, Walter Sutton (1877-1916) observe pour la première fois une méiose, et propose la théorie chromosomique de l'hérédité. Celle-ci suggère que les chromosomes seraient les supports des gènes. Il remarque également que le modèle de séparation des chromosomes supporte tout à fait la théorie de Mendel. En 1909, Wilhelm L. Johannsen (1857-1927) introduit le terme de *gène*.

Les résultats de Sutton sur la théorie chromosomique seront confirmés en 1910 par le généticien américain Thomas Morgan (1866-1945). Celui-ci étudia l'embryogenèse de la *drosophile*, nommée communément la *mouche du vinaigre*. En étudiant la descendance d'un mâle aux yeux blancs avec une femelle aux yeux rouges, il suggère que le caractère « yeux blancs »est récessif par rapport au caractère « yeux rouges »et est lié au sexe de la mouche. Cette découverte démontra qu'à la suite d'une *mutation*, une modification d'un caractère héréditaire pouvait survenir, étayant alors les lois de Mendel ainsi que la théorie de Sutton.

En 1913, Morgan construit avec Alfred Sturtevant (1891-1970) les premières cartes de localisation des gènes sur les chromosomes ; ce seront les premières *cartes génétiques*. En 1922, le biochimiste Phoebus Aaron Levene (1869-1940) identifie le sucre désoxyribose dans les acides nucléiques et en 1935, on

*Le symbole Ⓦ indique dans ce chapitre une image provenant de l'encyclopédie Wikipédia.

parle pour la première fois de l'*Acide DésoxyriboNucléique*, l'ADN. En 1944, Osward T. Avery (1877-1955) démontre, avec Colin Mc Leod (1909-1972) et Maclyn McCarthy (1911-2005), que l'ADN est une molécule associée à l'information héréditaire.

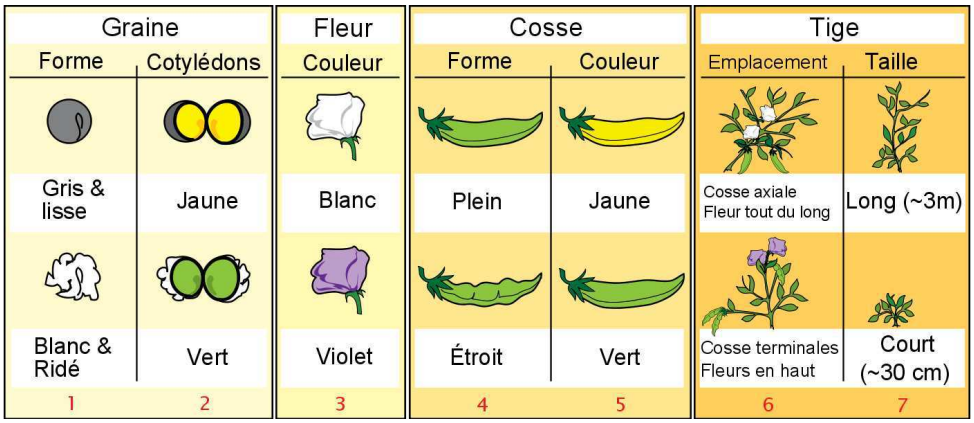


Figure 1.1 – Les sept phénotypes observés par Gregor Mendel ⓦ.

En 1953, Maurice Wilkins (1916-2004) réalise avec Rosalind Franklin (1920-1958) les premières photos d'une molécule d'ADN par diffraction des rayons X. S'appuyant sur ces travaux, James Watson (1928) et Francis Crick (1916-2004) présentent la même année le modèle en double hélice de l'ADN, et montrent que l'information génétique peut être portée par cette molécule. En 1957, le mécanisme de réplication de l'ADN est mis en évidence. Dans les années 60, François Jacob (1920) et Jacques Monod (1910-1976) élucident le mécanisme de la synthèse de protéines.

Depuis ces découvertes, les recherches en génétique ont permis de nombreuses découvertes, devenant également une science d'intérêts économique et médical. Les technologies modernes n'ont de cesse d'accroître nos connaissances dans ce domaine, ainsi que les données disponibles. Citons notamment, le premier génome séquencé, celui de la bactérie *Haemophilus influenzae* et le projet *Human Genome Project* (HGP) qui fournit en 2003 le premier séquençage d'un génome humain.

1.2 Généralités

Nous présentons dans cette partie les généralités sur le modèle de la conservation et de l'utilisation de l'information génétique. On appelle *génome* l'ensemble du matériel génétique d'un organisme. Celui-ci est constitué d'*acides nucléiques* : l'ADN pour **A**cide **D**ésoxyribo**N**ucléique.

1.2.1 L'ADN

L'ADN est l'une des plus grosses molécules de l'organisme et peut être composé de plus de 150 milliards d'atomes. Il se retrouve dans toutes les cellules vivantes, que ce soit chez les *procaryotes* (organismes à cellules sans noyau) ou chez les *eucaryotes* (organismes à cellules avec noyau) (cf. Figure 1.2). Une molécule d'ADN est formée de deux brins complémentaires enroulés et dont l'agencement décrit une double hélice. Chaque brin est une succession d'acides désoxyribonucléiques, appelés

les *nucléotides*, eux-mêmes constitués d'un *groupe phosphate*, d'un *désoxyribose* et d'une *base azotée* (cf. Figure 1.3). C'est l'enchaînement de ces bases azotées qui porte l'information génétique utile au développement des cellules.

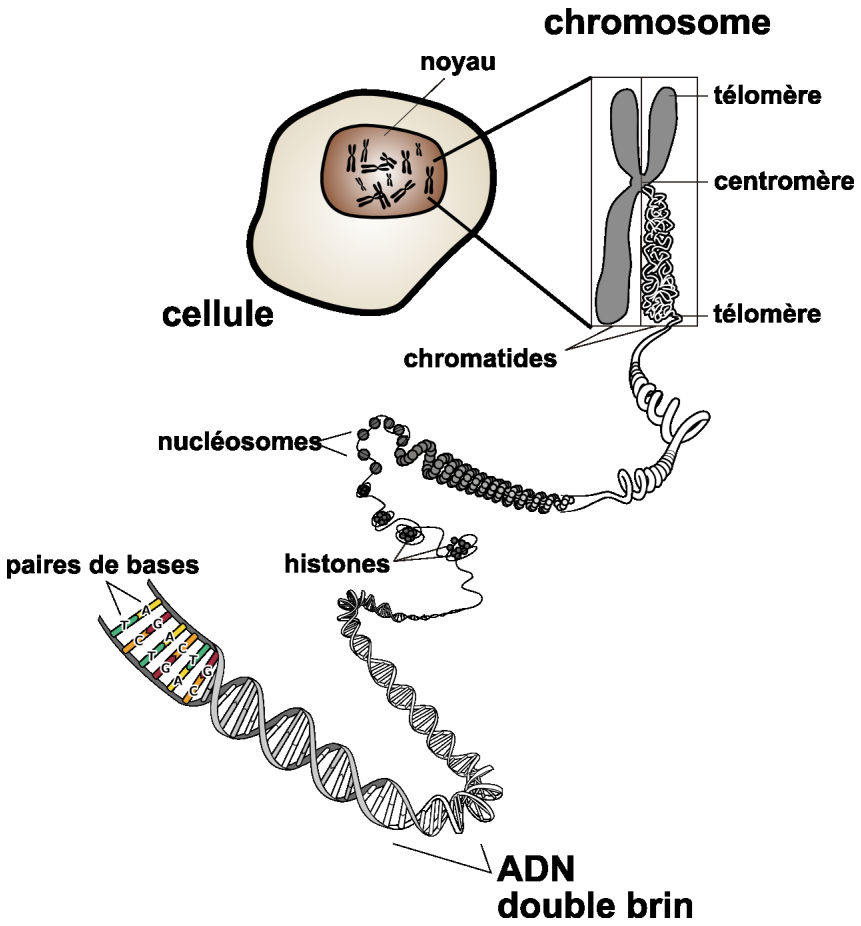


Figure 1.2 – Schéma d'un chromosome et de la molécule d'ADN (source : National Institutes of Health, National Human Genome Research Institute, Division of Intramural Research)

Il existe quatre *bases azotées* différentes : l'*adénine* (notée A), la *thymine* (notée T), la *cytosine* (notée C) et la *guanine* (notée G) (décrites en Figure 1.4). Dans chaque brin d'ADN, les nucléotides sont reliés entre eux par des liaisons appelées *liaisons 5'-3' phosphodiester*. D'autre liaisons, appelées *hydrogènes* associent les acides nucléiques en paires (un acide nucléique de chaque brin dans chaque paire) appelées *paires complémentaires* : l'adénine s'associe à la thymine (A-T) alors que la guanine s'associe à la cytosine (G-C). La structure de l'ADN ainsi obtenue est une structure en double hélice.

L'ADN a plusieurs rôles. Tout d'abord, il est le support de l'hérédité en contenant la totalité de l'information génétique par l'enchaînement de ses nucléotides. L'ADN joue un rôle essentiel lors de la *réplication* d'une cellule, c'est-à-dire son dédoublement. Par l'action de l'*ADN polymérase*, l'ADN se coupe en deux laissant l'ADN polymérase compléter les nucléotides par leur base azotée complémentaire, produisant ainsi deux molécules ADN filles, reproductions à l'identique de la molécule d'ADN

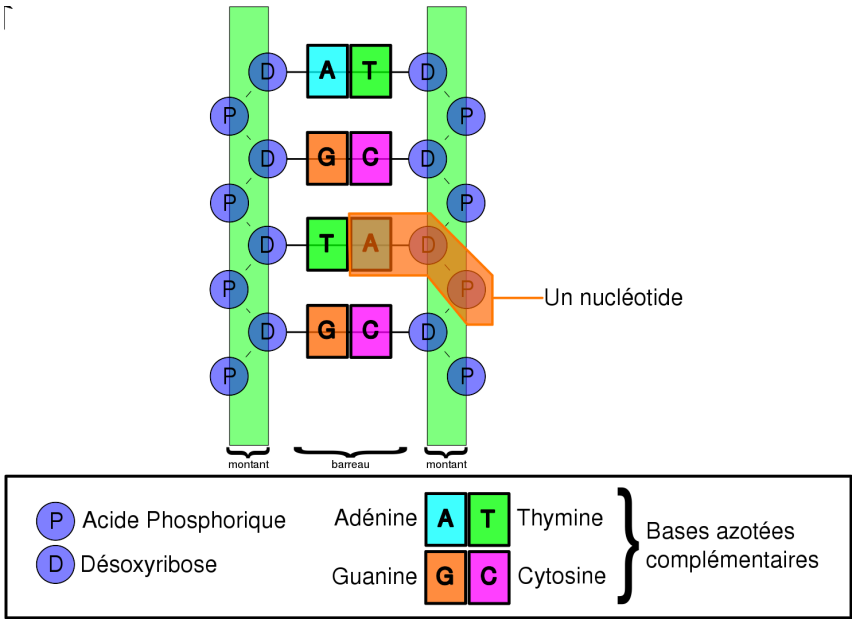


Figure 1.3 – Schéma représentant la structure des nucléotides ⓦ.

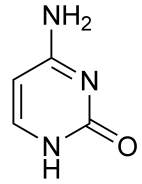
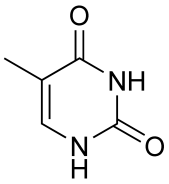
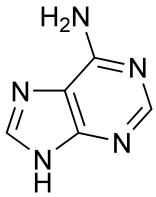
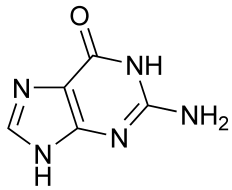
Pyrimidines		Purines	
			
cytosine (C)	thymine (T)	adénine (A)	guanine (G)

Figure 1.4 – Les quatre bases azotées de l'ADN ⓦ.

initiale. Cette phase de réplication permet la transmission de l'information génétique. Enfin, la molécule d'ADN est également le premier élément de la synthèse des *protéines*, constituants essentiels à la vie d'un organisme. La première phase de cette synthèse met en jeu l'ADN lors de la *transcription*, phase générant l'acide ribonucléique, communément appelé l'ARN.

1.2.2 L'ARN

L'ARN (pour **A**cide **R**ibo**N**ucléique) est une séquence d'*acide nucléique*, linéaire, résultant de la *transcription* de l'ADN et permettant de traiter l'information dans la cellule. L'ARN possède trois bases azotées communes à l'ADN : l'adénine (A), la cytosine (C) et la guanine (G). En revanche, la thymine (T) de l'ADN est remplacée dans l'ARN par l'uracile (notée U). La différence entre ces deux bases est le remplacement d'un groupement méthyle par un hydrogène dans l'uracile (cf. Figure 1.5).

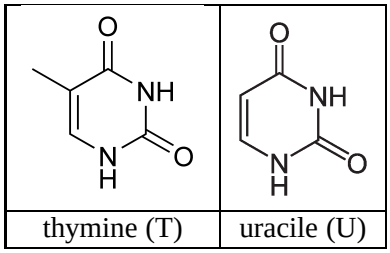
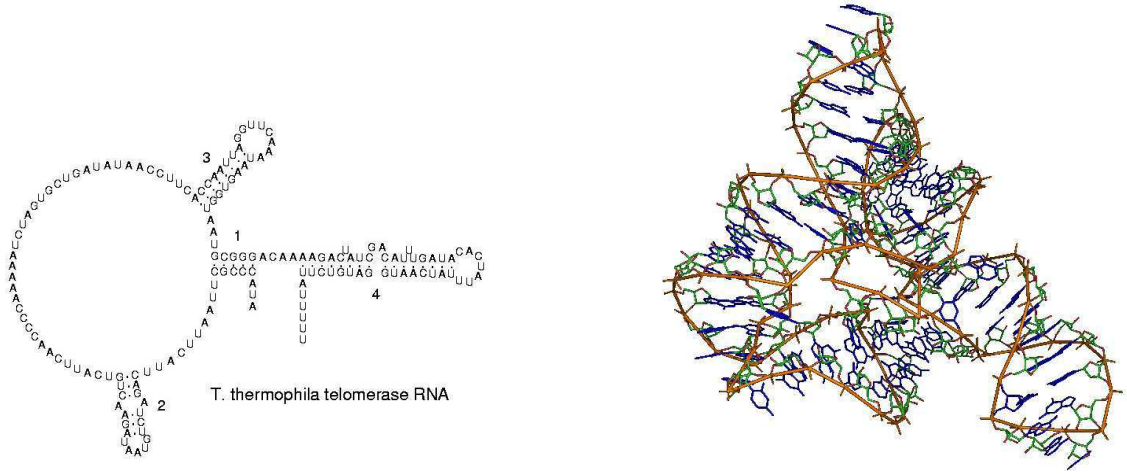


Figure 1.5 – Les bases azotées thymine et uracile ⓦ.

On distingue trois types de structures de l'ARN : les structures *primaire*, *secondaire* et *tertiaire*. La structure primaire correspond à la séquence de nucléotides du brin d'ARN. La structure secondaire (cf. Figure 1.6 a)) désigne quant à elle l'agencement planaire de la molécule qui se replie le plus souvent sur elle même, en raison de liaisons dites *covalentes* entre nucléotides. Enfin, on appelle structure tertiaire celle qui considère l'agencement de l'ARN en trois dimensions (cf. Figure 1.6 b)).



a) représentation de la structure secondaire d'un ARN b) représentation de la structure tertiaire d'un ARN

Figure 1.6 – Structures secondaire et tertiaire de l'ARN ⓦ.

1.2.3 Gènes et protéines

Le terme de *gène* désigne une unité d'information génétique transmise de génération en génération. On définit généralement un gène comme une portion d'ADN (i.e. une séquence de nucléotides) située sur un site précis, appelé *locus*. Un gène est composé d'une succession d'*exons* et d'*introns* (cf. Figure 1.7). Les exons correspondent aux séquences dites *codantes* par opposition aux portions dites *non-codantes* de l'ADN.

Les *protéines* sont des molécules essentielles à la vie d'un organisme. Elles sont produites à partir des gènes qui contiennent le code génétique permettant de les synthétiser. Cette synthèse est décomposable

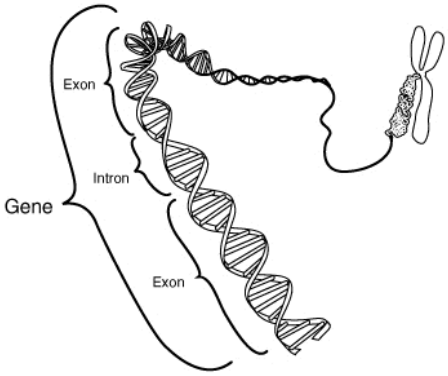


Figure 1.7 – Illustration d'un gène Ⓦ.

en deux étapes : la *transcription* et la *traduction*.

- La *transcription* de l'ADN en ARN messager est décomposée en trois phases : l'initiation, l'élongation et la terminaison (cf. Figure 1.8). Plus précisément, la transcription consiste à recopier les régions dites codantes de l'ADN en molécules d'ARN. Ces molécules d'ARN seront ensuite utilisées dans la phase suivante afin de produire des séquences protéiques. La réaction de transcription est catalysée par l'enzyme appelée *ARN polymérase*.
- La *traduction* transforme l'ARN messager en protéine. Cette opération se réalise dans le cytoplasme sous l'effet d'un *ribosome* qui, en se fixant sur l'ARN messager, va initier la production d'acides aminés. Le ribosome va ensuite « lire » la séquence d'ARN codon par codon (un codon étant un groupe de trois nucléotides) et produire une ou plusieurs protéines en chaînant un à un les acides aminés associés aux codons.

Chez les procaryotes, la transcription et la traduction s'exécutent simultanément. En revanche, chez les eucaryotes, la transcription et la traduction sont successives car elles se déroulent dans des régions distinctes de la cellule. En effet, la transcription se réalise à l'intérieur même du noyau, alors que la traduction s'effectue dans le cytoplasme.

1.3 L'évolution des génomes

On appelle *évolution* la modification des organismes par changements successifs à partir d'autres organismes. Nous pouvons considérer deux types de transformations : les *mutations* et les *réarrangements génomiques*.

- Les *mutations* sont des phénomènes spontanés et sont dues à des erreurs dans le processus de réplication de l'ADN. Malgré des mécanismes de réparation de l'ADN lors de la réplication, certains nucléotides peuvent être mal copiés, ou bien tout simplement retirés. Nous distinguons communément trois types de mutations : l'*insertion* de nucléotide, la *suppression* et la *substitution*.

  – **L'insertion**. Un nucléotide est ajouté dans la séquence d'ADN :

  $$\cdots \text{ACGTACAG} \cdots \xrightarrow{insertion} \cdots \text{ACGTTACAG} \cdots$$

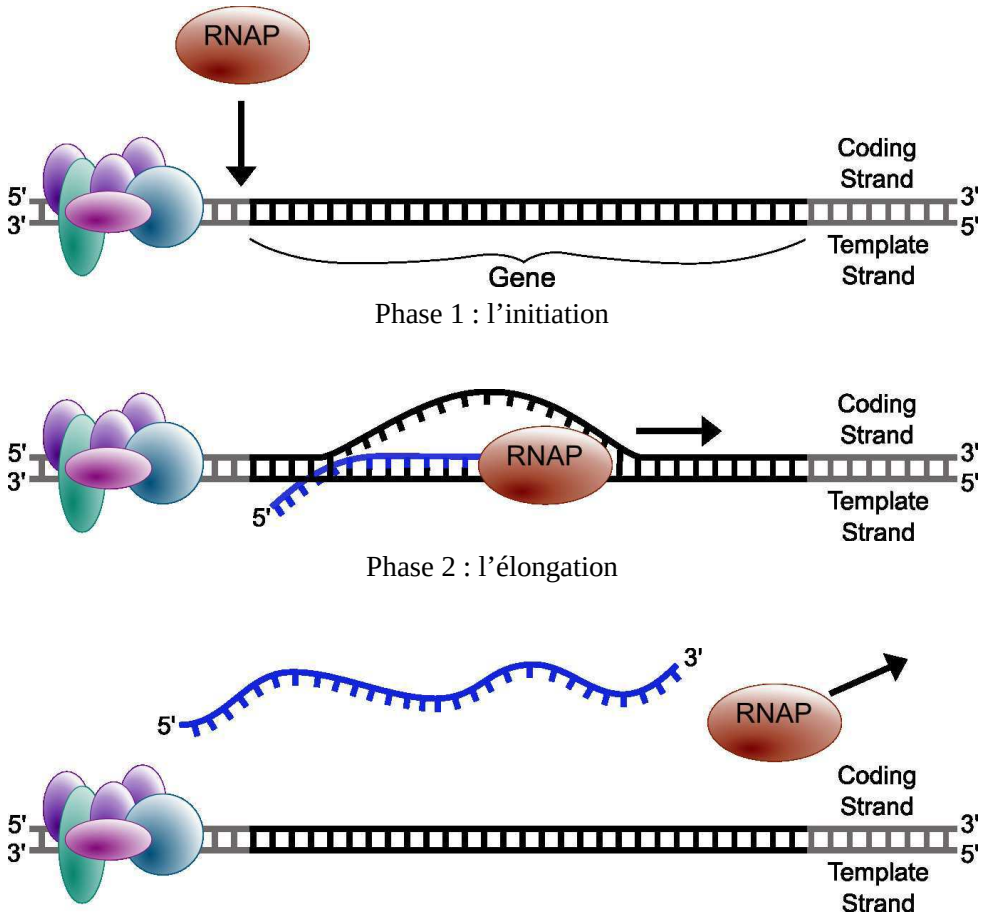


Figure 1.8 – Les différentes étapes de la transcription de l'ADN à l'ARN ⓦ.

– **La suppression**. Un nucléotide est retiré de la séquence d'ADN :

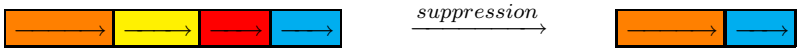
$$\cdots \text{ACGTACAG} \cdots \xrightarrow{suppression} \cdots \text{ACG}\underset{\uparrow}{\text{T}}\text{CAG} \cdots$$

– **La substitution**. Un nucléotide est modifié dans la séquence d'ADN :

$$\cdots \text{ACGTACAG} \cdots \xrightarrow{substitution} \cdots \text{ACG}\underline{\text{C}}\text{ACAG} \cdots$$

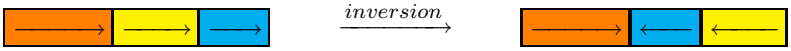
- **Les réarrangements génomiques**. Ce type de changement s'opère à une échelle beaucoup plus large. Il s'agit d'une modification de l'organisation des gènes au sein des génomes. Communément, un réarrangement génomique est considéré comme un changement de l'ordre des gènes d'un chromosome ou d'un génome. En modifiant l'organisation des gènes des chromosomes, les réarrangements génomiques sont responsables, avec les mutations ponctuelles, du polymorphisme génétique. Ces réarrangements peuvent affecter soit un seul chromosome, soit plusieurs chromosomes d'un même organisme. Parmi les modifications n'affectant qu'un seul chromosome, listons les cinq opérations les plus considérées (chaque bloc de couleur correspond à un gène) :

– **La suppression**. Il s'agit d'un retrait d'une suite de gènes d'un chromosome :

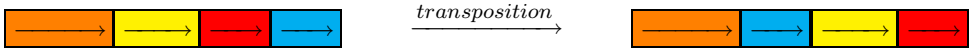


– **L'insertion**. Une suite de gènes est ajoutée dans le chromosome :

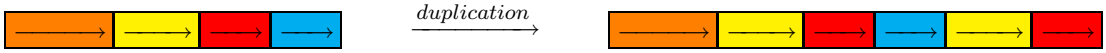
– **L'inversion**. Une suite de gènes est retournée, ce qui induit une inversion de l'ordre de ces gènes :



– **La transposition**. Une suite de gènes est déplacée sur le chromosome :

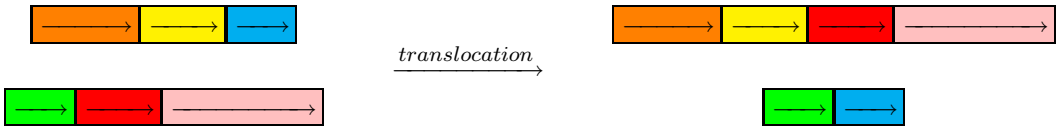


– **La duplication**. Une partie d'un chromosome est recopiée à une autre position sur le chromosome :

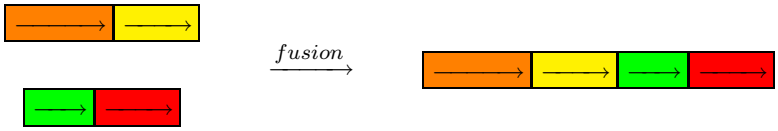


Parmi les modifications affectant plusieurs chromosomes, les trois opérations les plus considérées sont les suivantes :

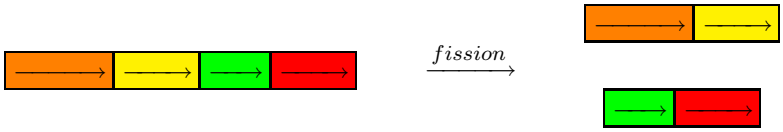
– **La translocation**. Il s'agit d'un échange de suites de gènes consécutifs entre deux chromosomes (avec une suite éventuellement vide pour l'un des chromosomes) :



– **La fusion**. Deux chromosomes fusionnent pour n'en former qu'un seul :



– **La fission**. Un chromosome se sépare en deux chromosomes distincts :



Ces modifications (mutations et réarrangements génomiques) sont en partie responsables de la diversité des organismes vivants. En effet, par changements successifs des génomes au cours des générations, de nouvelles espèces peuvent apparaître, on parle alors de *spéciation*. Lors de l'évolution, un gène peut être dupliqué (par le processus de *duplication*) ou bien être modifié (par exemple par mutations). Les gènes provenant d'un même gène ancestral sont dits des *copies d'un gène*. Ces copies ne sont pas identiques de par les changements évolutifs survenus. Étant donné un ensemble de génomes, toutes les copies d'un gène sont appelées des gènes *homologues* et forment une *famille de gènes*. Des gènes homologues de deux espèces différentes sont dits *orthologues* s'ils proviennent d'un même gène présent dans le

dernier ancêtre commun des deux espèces (cf. gènes g_1 et g_2 de la Figure 1.9 a)) et des gènes (non nécessairement d'un même génome) sont dits *paralogues* s'ils dérivent d'une duplication. On distingue deux types de gènes paralogues : les *in-paralogues* et les *out-paralogues*. Des gènes in-paralogues désignent des gènes paralogues pour lesquels la duplication s'est réalisée après la spéciation (cf. gènes g_1 et g_2 de la Figure 1.9 b)). À l'inverse, des gènes out-paralogues (non nécessairement de la même espèce) sont des gènes paralogues pour lesquels la duplication s'est réalisée avant la spéciation (cf. gènes g'_1, g'_2, g''_1 et g''_2 de la Figure 1.9 c)).

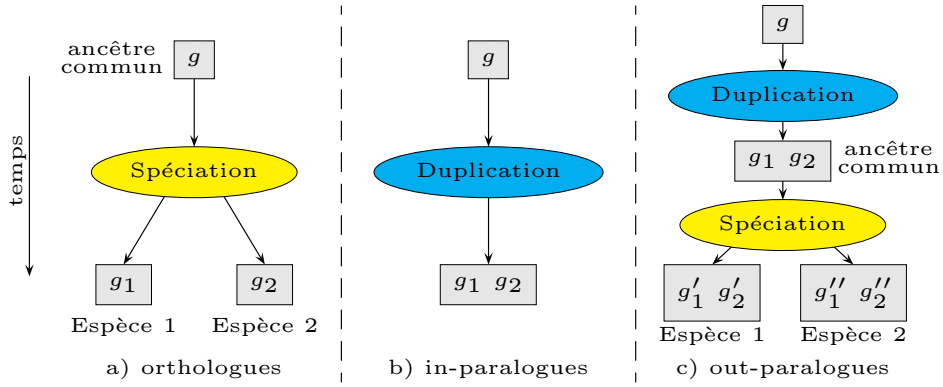


Figure 1.9 – Illustration de gènes orthologues, in-paralogues et out-paralogues. Dans le cas a), les gènes g_1 et g_2 sont orthologues. Dans le cas b), les gènes g_1 et g_2 sont dits in-paralogues. Enfin, dans le cas c), les gènes g'_1, g''_1, g'_2 et g''_2 sont appelés out-paralogues

La recherche de régions conservées lors de l'évolution est un problème majeur pour la compréhension des génomes. L'hypothèse sous-jacente est que la conservation d'une région dans plusieurs génomes d'espèces différentes est due à une fonction biologique commune. Ainsi, au niveau des gènes, la recherche d'ensemble de gènes conservés lors de l'évolution met en évidence les ensembles de gènes dits *fonctionnels* et correspondant à une fonction biologique spécifique.

Dans le chapitre suivant, nous introduisons les modèles utilisés pour représenter les génomes afin de comparer deux génomes par l'outil informatique et exposons formellement les problèmes étudiés dans ce mémoire.

CHAPITRE 2

Présentation générale

Dans ce chapitre, nous introduisons les notions et définitions utiles à ce mémoire. Nous donnons tout d'abord les diverses notations usuelles, puis exposons les différents modèles de couplage des gènes, permettant de traiter la présence de gènes dupliqués. Nous finissons ce chapitre en définissant formellement les mesures étudiées, ainsi que les problèmes traités.

2.1 Notations

Nous introduisons les notions utiles à la représentation des génomes :

Notation 1.
Une famille de gènes est représentée par un entier positif. Un ensemble de familles de gènes est représenté par un ensemble $\mathcal{F}$ d'entiers positifs, appelé alphabet.

Notation 2.
Soit un alphabet $\mathcal{F}$. On appelle un élément signé *de $\mathcal{F}$ un élément de $\mathcal{F}$ auquel est associé un signe ($+$ ou $-$). Un* gène *est représenté par un élément signé de $\mathcal{F}$, où l'étiquette (i.e. l'entier) correspond à la famille auquel il appartient et où le signe indique l'orientation du gène au sein du génome.*

Notation 3. ***(Chaîne et sous-chaîne)***
Soit un alphabet $\mathcal{F}$. On appelle chaîne *une suite d'éléments signés de $\mathcal{F}$. Étant donnée une chaîne s, on appelle* sous-chaîne *de s une chaîne présente dans s. Par exemple, soient l'alphabet $\mathcal{F} = \{1,2,3,4\}$ et la chaîne $s = +1 - 2 + 1 + 3 - 4 - 1$. La chaîne $+3 - 4 - 1$ est une sous-chaîne de s.*

Notation 4. ***(Permutation)***
Soit un alphabet $\mathcal{F}$. On appelle permutation *une chaîne définie sur $\mathcal{F}$ pour laquelle les étiquettes de ses éléments signés sont distinctes deux à deux.*

Nous précisons ci-dessous la représentation formelle des génomes :

Notation 5. ***(Représentation d'un génome)***
Nous représentons un génome G par une chaîne sur un alphabet $\mathcal{F}$. Par exemple, soit $\mathcal{F} = \{1,2,3,4\}$. Nous noterons un génome G de la manière suivante : $G = +1 - 2 + 1 + 3 - 4 - 1$. Le second élément signé du génome est le gène -2. Ce gène appartient à la famille de gènes d'étiquette 2 et est orienté négativement sur le génome.

Nous donnons maintenant plusieurs notations usuelles.

Notation 6. ***(Notations usuelles)***
Soient un alphabet $\mathcal{F}$, un génome G sur $\mathcal{F}$ et un gène g de G. Nous notons :

- $\mathcal{F}_G$ *: l'ensemble des familles de gènes présentes dans le génome* G
- $-g$ *: un gène de la même famille que* g *et de signe opposé*
- $|g|$ *: la famille de gènes auquel appartient le gène* g*. Nous avons* $|g| \in \mathcal{F}_G$
- η_G *: la taille du génome* G*, correspondant au nombre de gènes qu'il contient*
- $G[p]$, $1 \leq p \leq \eta_G$ *: le gène situé à la position* p *dans* G
- *Pour toute famille* $f \in \mathcal{F}_G$*, soit* $occ(f,G,i,j)$, $1 \leqslant i \leqslant j \leqslant \eta_G$*, le nombre de gènes de la famille* f *dans* G *entre les positions* i *et* j *(*i *et* j *comprises)*
- *Pour toute famille* $f \in \mathcal{F}_G$*, soit* $occ(f,G) = occ(f,G,1,\eta_G)$*, i.e. le nombre de gènes de la famille* f *dans* G
- $occ(G) = \max\{occ(f,G) | f \in \mathcal{F}_G\}$

Si G *ne contient pas deux gènes de la même famille, nous dirons que* G *ne contient aucun* gène dupliqué*. Supposons maintenant que* G *ne contient aucun gène dupliqué. Soient deux gènes distincts* a*,* b *tels que* a *précède* b *dans* G *(i.e. le gène* a *est situé avant le gène* b *dans le génome, celui-ci étant représenté par une suite de gènes). Nous notons :*

- $[a,b]_G$ *: la sous-chaîne de* G *commençant par* a *et finissant par* b *dans* G
- $G[a,b]$ *: l'ensemble* $S \subseteq \mathcal{F}_G$ *des familles de gènes pour lesquelles il existe au moins un gène situé entre* a *et* b *dans* G*, les familles* $|a|$ *et* $|b|$ *incluses. Nous avons l'égalité suivante :* $G[a,b] = \mathcal{F}_{[a,b]_G}$

Nous illustrons ces différentes notations par l'exemple suivant :

Exemple 2.1 (Notations usuelles).
Considérons le génome G *tel que* $G = +1\ +2\ +3\ +4\ +5\ -1\ -2\ +6\ -2$*. Ainsi, nous avons* $\mathcal{F}_G = \{1,2,3,4,5,6\}$, $\eta_G = 9$, $occ(1,G) = 2$, $occ(G) = 3$ *(trois gènes de la famille* 2*),* $G[1] = +1$, $G[7] = -2$, $-G[7] = +2$ *et* $|G[7]| = 2$*.*

Considérons maintenant le génome G *tel que* $G = +3 - 2 + 6 + 4 - 1 + 5$*. Le génome* G *ne contient aucun gène dupliqué. Nous avons* $G[+6,-1] = \{1,4,6\}$ *et* $[+6,-1]_G = +6\ +4\ -1$*.*

Par la suite, nous nous intéressons à la caractérisation des paires de génomes et des paires de gènes successifs.

Caractérisation des paires de génomes.
Afin de caractériser les paires de génomes que nous traiterons plus tard, nous introduisons ici quelques notations.

Définition 1. ***(Types des paires de génomes)***
Soient deux génomes G_0 *et* G_1*. La paire de génomes* (G_0,G_1) *est dite* de type (x,y) *si* $occ(G_0) = x$ *et* $occ(G_1) = y$*. Dans ce mémoire, nous parlerons également d'*instance (x,y) *pour désigner une paire de génomes de type* (x,y)*.*

Définition 2. ***(Génomes équilibrés)***
Soient deux génomes G_0 *et* G_1*. La paire de génomes* (G_0,G_1) *est dite* équilibrée *si, pour chaque famille* $f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}$*, nous avons* $occ(f,G_0) = occ(f,G_1)$ *; sinon,* (G_0,G_1) *sera appelée* non-équilibrée*. De même, nous dirons que les génomes* G_0 *et* G_1 *sont* équilibrés *(resp.* non-équilibrés*) si la paire* (G_0,G_1) *est équilibrée (resp. non-équilibrée).*

Remarquons ici que la paire (G_0,G_1) de type (x,x) n'est pas nécessairement équilibrée. En effet, prenons par exemple les génomes G_0 et G_1 tels que $G_0 = +1\ +2\ +3\ +4\ +5\ -1\ -2\ +6\ -2$

et $G_1 = +1\ +2\ +3\ +4\ +5\ -1\ -3\ +6\ -3$. La paire (G_0,G_1) est ici de type $(3,3)$ puisque $occ(G_0) = 3$ (famille 2) et puisque $occ(G_1) = 3$ (famille 3). En revanche, la paire (G_0,G_1) n'est pas équilibrée à cause des familles 2 et 3.

Notations sur les paires de gènes successifs.
Nous introduisons maintenant plusieurs définitions liées aux paires de gènes successifs sur un génome.

Définition 3. ***(Duo)***
Soit un génome G. Une paire de gènes consécutifs $d_i = (G[i],G[i+1])$ de G est un duo *de G, avec $1 \leqslant i < \eta_G$. Nous noterons $-d_i$ le duo $(-G[i+1], -G[i])$.*

Définition 4. ***(Duos identiques** et **Duos inversés)***
Soient deux duos $d_0 = (a_0,b_0)$ et $d_1 = (a_1,b_1)$ d'un génome G. Les duos d_0 et d_1 sont des duos identiques *si $a_0 = a_1$ et $b_0 = b_1$ et sont des* duos inversés *si d_0 et $-d_1$ sont identiques.*

Définition 5. ***(Couple de duos)***
Soit une paire de duos (d_0,d_1). La paire (d_0,d_1) est appelée un couple de duos *de (G_0,G_1) si (i) d_0 est un duo de G_0, (ii) d_1 est un duo de G_1, et (iii) si d_0 et d_1 sont identiques ou inversés.*

Exemple 2.2 (Duo, duos inversés et couple de duos).
Soient les génomes G_0 et G_1 tels que $G_0 = +1\ +2\ +3\ +4\ +5\ -1\ -2\ +6\ -2$ et $G_1 = +2\ -1\ +6\ +3\ -5\ -4\ +2\ -1\ -2$. Soient les duos $d_0 = (G_0[4],G_0[5]) = (+4, +5)$ et $d_1 = (G_1[5],G_1[6]) = (-5, -4)$. Ainsi, d_0 et d_1 sont des duos inversés et par conséquent la paire (d_0,d_1) est un couple de duos de (G_0,G_1).

2.2 Duplications de gènes

Lorsque les génomes contiennent des gènes dupliqués, les mesures définies uniquement sur des permutations ne peuvent être calculées. Une solution possible consiste fictivement à (i) déterminer une correspondance (i.e. un couplage) entre les gènes de G_0 et G_1 ; (ii) retirer des génomes les gènes non couplés ; (iii) renommer les gènes de G_0 et G_1 restants suivant le couplage choisi. Les deux nouveaux génomes (fictifs) G'_0 et G'_1 obtenus contiennent ainsi exactement les mêmes gènes et ne contiennent aucun gène dupliqué, ceci rendant possible le calcul des différentes mesures entre génomes définies sur des permutations. Nous précisons maintenant de manière plus formelle cette idée de couplage.

Définition 6. ***(Couplage de gènes)***
Soient deux génomes G_0 et G_1. Un couplage *(de gènes) $\mathcal{M}$ de (G_0,G_1) est un ensemble de paires (p_0,p_1) tel que $1 \leqslant p_0 \leqslant \eta_{G_0}$, $1 \leqslant p_1 \leqslant \eta_{G_1}$, $|G_0[p_0]| = |G_1[p_1]|$ et tel que, pour toutes paires distinctes $(p_0,p_1) \in \mathcal{M}$ et $(q_0,q_1) \in \mathcal{M}$, nous avons $p_0 \neq q_0$ et $p_1 \neq q_1$.*

Voici ci-dessous un exemple de couplage de gènes.

Exemple 2.3 (Couplage de gènes).

position	*1*	*2*	*3*	*4*	*5*	*6*	*7*
$G_0 =$	$+1$	*-2*	$+3$	*+1*	$+2$	*-1*	$+5$
$G_1 =$	*+2*	-3	*-1*	*+1*	-4	$+3$	

Nous avons dans ce cas $\mathcal{M} = \{(2,1),(4,4),(6,3)\}$.

Nous définissons maintenant les notions de $\mathcal{M}$-*élagage* et de *paire* $\mathcal{M}$-*élaguée*.

Définition 7. ***($\mathcal{M}$-élagage)***
Soient deux génomes G_0 *et* G_1 *et soit un couplage* $\mathcal{M}$ *de* (G_0,G_1). *Un* $\mathcal{M}$-élagage *de* (G_0,G_1) *est une paire* (G'_0,G'_1) *de génomes obtenue à partir de* G_0 *et* G_1 *par les deux étapes suivantes :*

1. *suppression de tous les gènes non-couplés (i.e. n'appartenant à aucune paire de* $\mathcal{M}$*)*
2. *renommage des gènes restants selon le couplage* $\mathcal{M}$. *Pour cela, nous associons tout d'abord à chaque paire* (i,j) *de* $\mathcal{M}$ *une étiquette unique (par exemple, la position de* (i,j) *en ordonnant lexicographiquement* $\mathcal{M}$*). Puis, pour chaque élément* (i,j) *de* $\mathcal{M}$, *nous renommons les gènes* $G_0[i]$ *et* $G_1[j]$ *par son étiquette associée. Les signes des gènes* $G_0[i]$ *et* $G_1[j]$ *restent quant à eux inchangés.*

Définition 8. ***(Paire $\mathcal{M}$-élaguée)***
Soient deux génomes G_0 *et* G_1 *et soit un couplage* $\mathcal{M}$ *de* (G_0,G_1). *On appelle la paire de génomes* (G'_0,G'_1) *obtenue par le* $\mathcal{M}$*-élagage de* (G_0,G_1) *la paire* $\mathcal{M}$-élaguée *de* (G_0,G_1).

Exemple 2.4 (Illustration d'un $\mathcal{M}$-élagage).
Appliquons le $\mathcal{M}$*-élagage sur l'exemple 2.3*

position	*1*	*2*	*3*	*4*	*5*	*6*	*7*
$G_0 =$	$+1$	*-2*	$+3$	*+1*	$+2$	*-1*	$+5$
$G_1 =$	*+2*	-3	*-1*	*+1*	-4	$+3$	

Étape 1. Après suppression des gènes non couplés, nous obtenons :

position	*1*	*2*	*3*	*4*	*5*	*6*	*7*
$G'_0 =$		*-2*		*+1*		*-1*	
$G'_1 =$	*+2*		*-1*	*+1*			

Étape 2. Après renommage des gènes selon le couplage $\mathcal{M}$, *nous obtenons :*

$$G'_0 = -1\ +2\ -3$$

$$G'_1 = +1\ -3\ +2$$

Les génomes G'_0 *et* G'_1 *ne contiennent aucun gène dupliqué et sont définis sur le même alphabet.*

Nous précisons ci-dessous les trois modèles classiques de couplage : le modèle *couplage exemplaire*, le modèle *couplage maximal* et enfin le modèle *couplage intermédiaire*. La Figure 2.1 illustre ces trois modèles. Nous fixons maintenant deux génomes G_0 et G_1 tels que $\mathcal{F}_{G_0} \cap \mathcal{F}_{G_1} \neq \emptyset$, i.e. il existe au moins une famille de gènes présente sur les deux génomes.

- Le *modèle couplage exemplaire* [San99] : Pour chaque famille de gènes f présente dans G_0 et G_1, nous gardons dans le couplage $\mathcal{M}$ seulement un gène de la famille f dans G_0 et dans G_1. Nous construisons ensuite la paire $\mathcal{M}$-élaguée (G_0^E,G_1^E) de (G_0,G_1). Le triplet $(G_0^E,G_1^E,\mathcal{M})$ est appelé un *couplage exemplaire* de (G_0,G_1). D'un point de vue biologique, lorsque l'on compare deux espèces E_0 et E_1, ce modèle suppose que, étant donnée une famille de gènes f, il n'existait qu'un seul gène ancestral chez l'ancêtre commun de E_0 et E_1 (qui a engendré ensuite tous les homologues). Ainsi, pour ce modèle, s'il existe deux homologues de la famille f dans G_0 (resp.

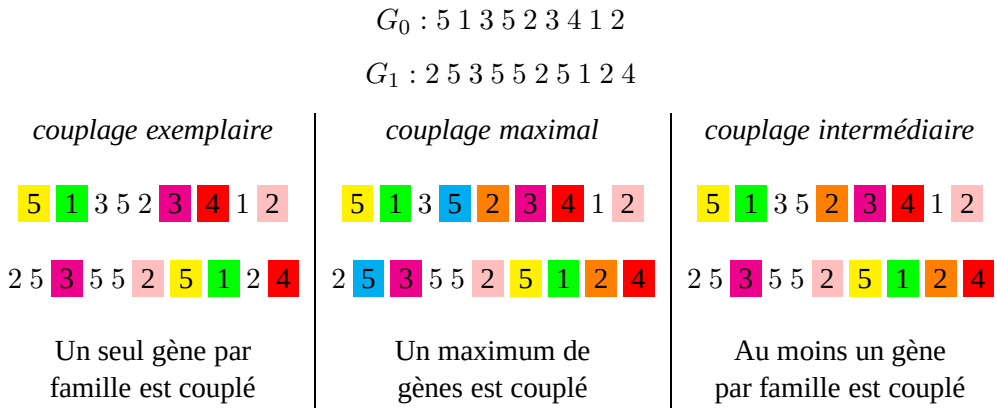


Figure 2.1 – Exemple de couplage des gènes pour chacun des modèles. Le couplage est représenté par le code couleur. Les signes des gènes ne sont pas affichés pour une meilleure lisibilité.

dans G_1), on suppose que la duplication engendrant ces deux homologues s'est réalisée après la spéciation engendrant E_0 et E_1.

- Le *modèle couplage maximal* [TM03] : dans ce modèle, nous gardons dans le couplage $\mathcal{M}$ le nombre maximum de gènes dans les deux génomes. Plus précisément, nous considérons les couplages qui associent, pour chaque famille f, exactement $min(occ(f,G_0), occ(f,G_1))$ occurrences. Nous construisons ensuite la paire $\mathcal{M}$-élaguée (G_0^M,G_1^M) de (G_0,G_1). Le triplet $(G_0^M,G_1^M,\mathcal{M})$ obtenu est appelé un *couplage maximal* de (G_0,G_1). À l'opposé du modèle *couplage exemplaire*, nous supposons ici qu'un maximum de duplications s'est réalisé avant la spéciation.

- Le *modèle couplage intermédiaire* [AFRV07] : dans ce modèle, pour chaque famille f présente à la fois dans G_0 et dans G_1, nous gardons dans $\mathcal{M}$ un nombre arbitraire k_f de gènes tel que $1 \leq k_f \leq min(occ(f,G_0),occ(f,G_1))$, afin d'obtenir la paire $\mathcal{M}$-élaguée (G_0^I,G_1^I) de (G_0,G_1). Nous appelons le triplet $(G_0^I,G_1^I,\mathcal{M})$ un *couplage intermédiaire* de (G_0,G_1).

Remarquons la propriété suivante :

Propriété 1. ***(Équivalence des modèles de couplage)***
Soient deux génomes G_0 et G_1. Si nous avons $occ(G_0) = 1$ ou $occ(G_1) = 1$ alors les modèles couplage exemplaire, couplage maximal *et* couplage intermédiaire *sont équivalents.*

Cette propriété est immédiate. En effet, si l'un des génomes ne contient aucun gène dupliqué, tout couplage des gènes ne pourra contenir deux gènes d'une même famille dans un même génome. Dans ces conditions, les modèles *couplage maximal* et *couplage intermédiaire* se ramènent au modèle *couplage exemplaire*.

Notation. Pour tout modèle, si l'un des deux génomes ne contient aucun gène dupliqué, nous nous permettrons de ne pas indiquer le type de couplage (E, I ou M) sur ce génome, celui-ci n'apportant dans ce cas aucune indication par la Propriété 1.

2.3 Intervalle commun et intervalle conservé

2.3.1 Définitions

Nous définissons ici la notion d'*intervalle commun*, initialement définie pour deux permutations par Uno et Yagiura dans [UY00] et la notion d'intervalle conservé, introduite par Bergeron et Stoye dans [BS03] pour deux permutations signées.

Définition 9. ***Intervalle commun*** *[UY00].*
Soient deux permutations G_0 et G_1. Nous appelons un intervalle commun *de (G_0,G_1) une sous-chaîne s de G_0 telle que G_1 contient une permutation de s sans prise en compte des signes.*

Exemple 2.5 (Illustration d'un intervalle commun).
Considérons les génomes suivants :

$$G_0 = +1 + 2 \text{ +3 +4 +5}$$

$$G_1 = +2 \text{ -4 +3 +5 } + 1$$

La sous-chaîne $[+3, +5]_{G_0}$ est un intervalle commun de (G_0,G_1).

Définition 10. ***Intervalle conservé*** *[BS03].*
Soient deux permutations G_0 et G_1. Considérons deux gènes a et b de G_0 tels que a précède b, où la notion de précédence accepte ici $a = b$ (ce qui n'est pas le cas dans [BS03]). La sous-chaîne $[a,b]_{G_0}$ est un intervalle conservé de (G_0,G_1) si l'une des deux propriétés suivantes est vérifiée :

- *a précède b dans G_1 et $G_0[a,b] = G_1[a,b]$*
- *$-b$ précède $-a$ et $G_0[a,b] = G_1[-b, -a]$*

Remarquons qu'un intervalle conservé est toujours un intervalle commun, mais qu'il possède des restrictions supplémentaires sur ses deux extrémités. Nous illustrons ci-dessous la notion d'intervalle conservé.

Exemple 2.6 (Illustration d'un intervalle conservé).
Considérons les génomes suivants :

$$G_0 = +1 \text{ +2 +3 +4 +5}$$

$$G_1 = \text{ -5 -4 +3 -2 } + 1$$

La sous-chaîne $[+2, +5]_{G_0}$ est un intervalle conservé de (G_0,G_1). En effet, le gène -2 précède -5 dans G_1 et $G_1[-5, -2] = G_0[+2, +5]$.

Uno et Yagiura ont proposé dans [UY00] un algorithme calculant les intervalles communs entre une permutation P de taille n et la permutation identité (i.e. la chaîne $+1\ +2 \ldots + n$). Cet algorithme a une complexité en $O(n + N)$ où N est le nombre d'intervalles communs. Puisque le nombre d'intervalles communs est dans le pire des cas de taille $O(n^2)$, la complexité de l'algorithme est en $O(n^2)$. Ce résultat a été étendu au calcul des intervalles communs de plusieurs permutations, notamment dans [HS01, LPW05, BCdMR08].

Étant donnés deux génomes, le nombre d'intervalles communs (ou conservés) entre ces génomes peut être considéré comme une mesure de similarité. Plus le nombre d'intervalles sera grand, plus les

génomes seront considérés comme proches. D'un point de vue biologique, un intervalle commun (ou conservé) met en évidence une région conservée lors de l'évolution. En effet, celui-ci correspond à un ensemble de gènes qui se retrouvent consécutivement sur les génomes, mais pas nécessairement dans le même ordre. La recherche des intervalles communs ou conservés (et non plus le calcul de leur nombre) permet d'identifier les régions conservées, qui par hypothèse correspondent à des fonctions biologiques spécifiques.

2.3.2 Les problèmes *ICOM$_X$* et *ICONS$_X$*

Nous définissons ici les problèmes *ICOM$_X$* et *ICONS$_X$*, $X \in \{E, M, I\}$, dans leurs différentes variantes.

Problème : *ICOM$_E$* (resp. *ICOM$_M$*, *ICOM$_I$*)
Entrée : Deux génomes G_0 et G_1. Un entier k.
Problème de décision : Existe-t-il un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) tel qu'il existe au moins k intervalles communs entre G'_0 et G'_1 ?
Problème d'optimisation : Trouver un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) maximisant le nombre d'intervalles communs entre G'_0 et G'_1.

Problème : *ICONS$_E$* (resp. *ICONS$_M$*, *ICONS$_I$*)
Entrée : Deux génomes G_0 et G_1. Un entier k.
Problème de décision : Existe-t-il un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) tel qu'il existe au moins k intervalles conservés entre G'_0 et G'_1 ?
Problème d'optimisation : Trouver un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) maximisant le nombre d'intervalles conservés entre G'_0 et G'_1.

Nous désignerons l'ensemble des trois problèmes *ICOM$_E$*, *ICOM$_M$* et *ICOM$_I$* (resp. *ICONS$_E$*, *ICONS$_M$* et *ICONS$_I$*) par la notation *ICOM$_X$* (resp. *ICONS$_X$*) et nous supposerons alors que $X \in \{E, M, I\}$. Les problèmes *ICOM$_E$* et *ICOM$_M$* ont été prouvés **NP**-Complets même pour les paires de génomes de type $(1,2)$ dans [CFRV06]. Nous pouvons conclure directement à partir de ces travaux et la Propriété 1 que *ICOM$_I$* est **NP**-Complet. D'autre part, dans [BR05], Blin et Rizzi ont étudié le problème de calcul de la distance nommée *distance d'intervalles conservés* introduite dans [BS03]. Cette distance diffère du nombre d'intervalles conservés que nous étudions dans ce mémoire, principalement dans le sens où (i) cette distance s'applique à deux *ensembles* de génomes (alors que notre mesure s'applique à deux génomes), et (ii) la distance entre deux génomes identiques de longueur n est égale à 0 (alors que notre mesure de similarité engendre dans ce cas $\frac{n(n+1)}{2}$ intervalles conservés). Blin et Rizzi [BR05] ont prouvé que trouver la distance d'intervalles conservés est **NP**-Complet, et ceci sous les trois modèles de couplage. Une analyse de leur preuve montre que nous pouvons aisément l'adapter pour prouver que les problèmes *ICONS$_X$* sont **NP**-Complets, même si l'un des génomes ne contient aucun gène dupliqué.

Nous illustrons par la Figure 2.2 l'élagage des génomes en deux étapes pour les modèles *couplage maximal* et *couplage exemplaire* et pour le calcul du nombre d'intervalles communs.

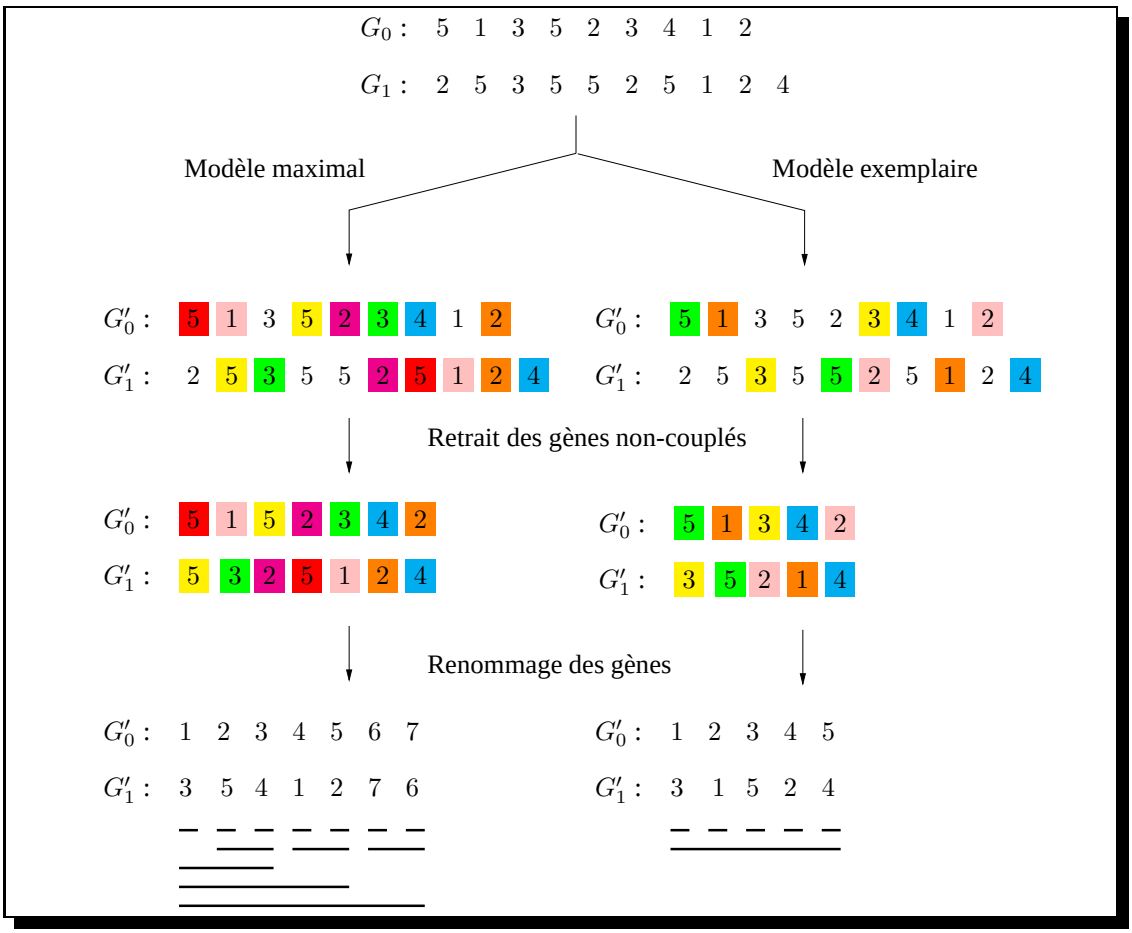


Figure 2.2 – Exemple de $\mathcal{M}$-élagage sous les modèles *couplage maximal* et *couplage exemplaire* pour le calcul des intervalles communs. Les intervalles communs sont représentés dans les deux cas par les lignes de la partie inférieure de la figure. Les signes sont retirés pour une meilleure lisibilité, ceux-ci n'intervenant pas dans le calcul des intervalles communs.

2.4 Point de cassure et Adjacence

2.4.1 Définitions

Nous définissons ici les notions de *point de cassure* [WEHM82] et d'*adjacence*. Ces deux notions sont définies uniquement pour deux permutations. Fixons deux permutations G_0 et G_1.

Définition 11. ***Point de cassure et adjacence.***
Soit un duo (a,b) *de* G_0. *Nous disons que le duo* (a,b) *induit une* adjacence *de* (G_0,G_1) *si* (a,b) *ou* $(-b,-a)$ *est un duo de* G_1. *Sinon, nous disons que le duo* (a,b) *induit un* point de cassure *de* (G_0,G_1).

Nous notons $Bkp(G_0,G_1)$ (resp. $Adj(G_0,G_1)$) le nombre de points de cassure (resp. le nombre d'adjacences) qui existent entre G_0 et G_1. Afin de comptabiliser les points de cassure aux extrémités (avant le premier gène et après le dernier), nous ajoutons usuellement deux gènes fictifs sur chacun

des deux génomes afin d'obtenir deux génomes ayant les mêmes extrémités. Dans l'ensemble de ce manuscrit, nous supposerons cet ajout comme étant déjà réalisé.

Exemple 2.7 (Illustration des points de cassure et des adjacences).
Considérons les génomes G_0 et G_1 tels que $G_0 = +0\ +1\ +2\ +3\ +4\ +5\ +6$ et $G_1 = +0\ +5\ -4\ -3\ +2\ +1\ +6$. Notons les points de cassure par le symbole $_\blacktriangle$. Nous avons :

$$G_0 = +0_\blacktriangle + 1_\blacktriangle + 2_\blacktriangle + 3 + 4_\blacktriangle + 5_\blacktriangle + 6$$

$$G_1 = +0 + 5 - 4 - 3 + 2 + 1 + 6$$

Ainsi, nous dirons, par exemple, que le duo $(+2, +3)$ de G_0 induit un point de cassure de (G_0,G_1) alors que le duo $(+3, +4)$ de G_0 induit une adjacence de (G_0,G_1).

Tout comme les intervalles communs et conservés, les points de cassures et les adjacences mettent en évidence des régions conservées lors de l'évolution. En effet, les parties du génome ne contenant que des adjacences (et donc aucun point de cassure) correspondent à des zones conservées. En revanche, à l'opposé des intervalles communs et conservés, nous mettons ici en évidence des régions sans réarrangements de gènes. Nous pouvons de plus définir des mesures entre génomes à partir des notions de point de cassure et d'adjacence. Le nombre de points de cassure peut être considéré comme une mesure de dissimilarité (plus il existe de points de cassure, plus les génomes sont éloignés) alors que le nombre d'adjacence sera une mesure de similarité entre génomes.

2.4.2 Les problèmes BD_X et ADJ_X

Nous définissons ici les problèmes BD_X et ADJ_X, $X \in \{E, M, I\}$, dans leur différentes variantes.

Problème : BD_E (resp. BD_M, BD_I)
Entrée : Deux génomes G_0 et G_1. Un entier k.
Problème de décision : Existe-t-il un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) tel qu'il existe au plus k points de cassure entre G'_0 et G'_1 ?
Problème d'optimisation : Trouver un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) minimisant le nombre de points de cassure entre G'_0 et G'_1.

Afin d'alléger les notations, nous désignerons l'ensemble de ces trois variantes par la notation BD_X et sous-entendrons alors que $X \in \{E, M, I\}$. Le problème BD_E a été introduit par Sankoff dans [San99], le problème BD_M dans [BCF04] et BD_I dans [AFRV07]. Le problème BD_E a été prouvé **NP**-Complet [Bry00] même appliqué à des paires de génomes de type $(1,2)$. Il en découle directement de [Bry00] par la Propriété 1 (cf. page 1) que BD_M et BD_I sont **NP**-Complets sous les mêmes conditions. Dans [BCF04], Blin et al. montrent également que BD_E est **NP**-Complet à partir du problème $Minimum\ Bin\ Packing$ même dans le cas où une seule famille de gènes est dupliquée.

Plusieurs résultats d'inapproximabilité existent. En particulier, il a été prouvé dans [CFZ06] que dans le cas général, BD_E ne peut être approximé sous un facteur $c \log n$, où c est une constante strictement positive, et ne peut être approximé sous un facteur 1.36 lorsque nous avons $occ(G_0) = occ(G_1) = 2$. Cependant, pour des génomes équilibrés G_0 et G_1 tels que $k = occ(G_0) = occ(G_1)$, plusieurs algorithmes d'approximation pour BD_M ont été proposés. Ces algorithmes d'approximation admettent respectivement un ratio de 1.1037 si $k = 2$ [GKZ04], 4 si $k = 3$ [GKZ04] et $4k$ dans le cas général [KW06]. Pour finir, nous montrerons dans la Section 3.5 que BD_E, BD_M et BD_I sont **APX**-Difficiles également pour les paires de génomes de type $(1,2)$.

Plusieurs algorithmes exacts ont été proposés pour résoudre ces problèmes. En particulier, Sankoff propose dans [San99] un algorithme de type *Branch and Bound* pour la résolution de BD_E ainsi que Nguyen et al. dans [NTZ05]. Pour le problème BD_M, Blin et al. proposent dans [BCF04] un algorithme de type *Branch and Cut*. Enfin, nous proposons dans la Section 4.2.6 un algorithme exact basé sur une transformation de BD_X en un problème pseudo-booléen.

Nous définissons maintenant les problèmes ADJ_E, ADJ_M et ADJ_I.

Problème : ADJ_E (resp. ADJ_M, ADJ_I)
Entrée : Deux génomes G_0 et G_1. Un entier k.
Problème de décision : Existe-t-il un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) tel qu'il existe au moins k adjacences entre G'_0 et G'_1 ?
Problème d'optimisation : Trouver un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) maximisant le nombre d'adjacences entre G'_0 et G'_1.

Nous désignerons l'ensemble des trois variantes ADJ_E, ADJ_M et ADJ_I par la notation ADJ_X. Nous montrerons en Section 3.3 que ADJ_E (resp. ADJ_M) est équivalent au problème BD_E (resp. BD_M) alors que ADJ_I et BD_I se révèlent être deux problèmes distincts. Nous prouverons également, en Section 3.5, que les problèmes BD_X et ADJ_I sont **APX**-Difficiles même pour des paires de génomes de type $(1,2)$. Nous définissons maintenant le problème *Eq-k-ADJ*$_M$ (k est un entier fixé strictement positif), variante du problème ADJ_M appliqué à deux génomes équilibrés.

Problème : *Eq-k-ADJ*$_M$
Entrée : Deux génomes équilibrés G_0 et G_1 avec $occ(G_0) = occ(G_1) = k$. Un entier p.
Problème de décision : Existe-t-il un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) tel qu'il existe au mois p adjacences entre G'_0 et G'_1 ?
Problème d'optimisation : Trouver un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) maximisant le nombre d'adjacences entre G'_0 et G'_1.

Enfin, nous définissons *Eq-ADJ_M* comme le problème *Eq-k-ADJ*$_M$ pour lequel k n'est pas borné. Nous proposons dans la Section 3.7 trois algorithmes d'approximation pour résoudre *Eq-k-ADJ*$_M$. Les ratios d'approximation que nous obtenons sont 1.1442 lorsque $k = 2$, $(3 + \epsilon)$ lorsque $k = 3$ (pour tout $\epsilon > 0$) et un ratio égal à 4 dans le cas général (problème *Eq-ADJ_M*).

2.4.3 Les problèmes ZBD_X

Nous définissons ici le problème de décision ZBD_E (resp. ZBD_I, ZBD_M) consistant à déterminer s'il existe un couplage exemplaire (resp. intermédiaire, maximal) n'engendrant aucun point de cassure.

Problème : ZBD_E (resp. ZBD_M, ZBD_I)
Entrée : Deux génomes G_0 et G_1.
Problème de décision : Existe-t-il un couplage exemplaire (resp. maximal, intermédiaire) $(G'_0,G'_1,\mathcal{M})$ de (G_0,G_1) tel qu'il n'existe aucun point de cassure entre G'_0 et G'_1 ?

Dans ce mémoire, nous désignerons l'ensemble de ces trois variantes par la notation ZBD_X. L'étude théorique de ces problèmes peut se révéler très utile pour caractériser les problèmes BD_E (resp. BD_M, BD_I). En effet, si le problème ZBD_E est prouvé **NP**-Complet, alors cela entraîne que BD_E n'est pas approximable, i.e. il n'existe *aucun* algorithme d'approximation pour BD_E. Certains résultats sont connus

sur ces problèmes. Notamment, Chen et al. ont montré dans [CFZ06] que ZBD_E est **NP**-Complet même pour les paires de génomes de type $(3,3)$. Nous affinerons ce résultat dans la Section 3.6 en montrant que ZBD_E est **NP**-Complet même pour les instances de type $(2,k)$, avec k non borné. Enfin, de récents travaux [BFSV09] montrent que ZBD_E est **NP**-Complet même pour les paires de génomes de type $(2,2)$.

<table>
<tr><th></th><th>couplage
exemplaire (E)</th><th>couplage
maximal (M)</th><th>couplage
intermédiaire (I)</th></tr>
<tr><td>$ICOM_X$
$ICONS_X$</td><td colspan="3">NP-Complet [CFRV06] (instance $(1,2)$)</td></tr>
<tr><td>BD_X</td><td colspan="3">NP-Complet [Bry00] (instance $(1,2)$)
NP-Complet [BCF04]*</td></tr>
<tr><td>ZBD_X</td><td>NP-Complet [CFZ06] (instance $(3,3)$)</td><td>?</td><td>?</td></tr>
</table>

Table 2.1 – Résultats de complexité algorithmique connus des problèmes BD_X, $ICOM_X$, $ICONS_X$ et ZBD_X. * Blin et al. ont montré que BD_M est **NP**-Complet même si une seule famille est dupliquée.

La Table 2.1 donne un aperçu des résultats théoriques connus pour les différents problèmes présentés. Dans le chapitre suivant, nous étudions ces problèmes et complétons nos connaissances théoriques sur ces problèmes.

CHAPITRE 3

Études théoriques

3.1 Introduction

Nous étudions dans ce chapitre les différents problèmes énoncés précédemment d'un point de vue théorique. Cette étude est utile afin de préciser la difficulté des problèmes rencontrés et ainsi de savoir quel type d'algorithme nous pourrons envisager pour les résoudre. Dans un premier temps, nous préciserons les éléments de théorie de la complexité utilisés dans ce chapitre. Puis, nous étudierons les relations entre les différents problèmes ADJ_X et BD_X. Ensuite, nous prouverons dans la Section 3.4 que les problèmes $ICOM_X$ et $ICONS_X$ sont **APX**-Difficiles, puis dans la Section 3.5, que les problèmes BD_X sont également **APX**-Difficiles. En Section 3.6, nous étudierons la complexité des problèmes ZBD_X. Enfin, nous proposerons dans la Section 3.7 trois algorithmes d'approximation pour le problème *Eq-k-ADJ_M* (i.e. problème ADJ_M sur des génomes équilibrées) de ratio dépendant du paramètre k.

Une partie de ces travaux a été publiée dans [AFR08a] et a été réalisée en collaboration avec Guillaume Fertin et Irena Rusu. Les relations entre les différents problèmes ADJ_X et BD_X, ainsi que l'étude de la complexité des problèmes ZBD_X, publiés dans [AFR+09], ont été le fruit d'un travail réalisé également avec Annelyse Thévenin et Stéphane Vialette.

3.2 Éléments de complexité

Nous donnons dans cette section les éléments de la théorie de la complexité nécessaires à ce document. Nous présentons tout d'abord les bases de la *théorie de la classification des problèmes*. Puis nous définissons les diverses notions d'approximations.

3.2.1 Classification des problèmes

La théorie de la complexité repose sur la définition de *classes de complexité* qui permettent de classer les problèmes en fonction de la complexité des algorithmes qui existent pour les résoudre. On distingue généralement deux types de problèmes. Les premiers sont appelés les *problèmes de décision*. Il s'agit des problèmes posant une question dont la réponse est *oui* ou *non*. Les seconds sont appelés les *problèmes d'optimisation*. Il s'agit de satisfaire un critère d'optimalité sur un ensemble de solutions potentielles. En d'autres termes, l'objectif est de trouver, étant donné un critère, une solution de meilleur *coût* (i.e. de meilleure valeur sur ce critère). Pour illustration, prenons par exemple le problème du voyageur de commerce (illustré par la Figure 3.1) défini de la manière suivante :

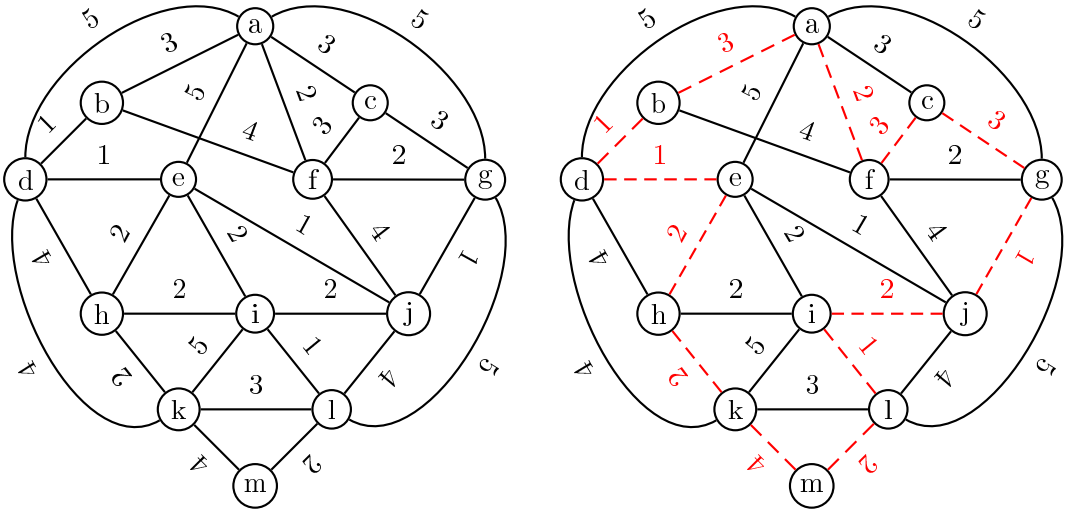


Figure 3.1 – À gauche, illustration du problème du voyageur de commerce. Chaque sommet du graphe correspond à une ville et les arêtes représentent les distances entre chaque paire de villes. Le problème consiste à trouver un circuit le plus court possible traversant exactement une fois chaque ville. À droite, le circuit proposé en pointillé passe par chaque ville une unique fois et est ainsi une *solution* du problème. De plus, ce circuit de longueur 27 est une *solution optimale*, puisqu'aucune solution n'est de meilleure qualité (i.e. aucun circuit n'est de longueur inférieure).

Problème : *VOYAGEUR DE COMMERCE* (Travelling Salesman Problem)
Entrée : Un ensemble de villes et les distances entre chaque paire de villes exprimées en kilomètres. Un entier positif K.
Problème de décision : Un voyageur peut-il prévoir un circuit passant par chaque ville exactement une fois, tel que la longueur du circuit n'excède pas K kilomètres ?
Problème d'optimisation : Trouver un circuit passant par chaque ville exactement une fois et de longueur minimum ?

Le problème du voyageur de commerce peut s'exprimer par les deux variantes présentées, c'est-à-dire soit comme un problème de décision ou soit comme un problème d'optimisation. Dans le premier cas, la réponse à ce problème est uniquement *oui* ou *non*. En revanche, pour sa seconde variante, nous devons trouver parmi toutes les *solutions* (i.e. ici, tous les circuits passant par chaque ville exactement une fois) celle ou celles qui satisfont le critère d'optimalité, ici la longueur du circuit. Dans ce manuscrit, quelque soit le problème étudié, nous distinguerons les notions de *solution* et de *solution optimale*, la première satisfaisant uniquement les conditions propres au problème considéré, la seconde satisfaisant de plus le critère d'optimalité.

Définition 12. ***(Classe* P*)***
Un problème de décision est dans la classe **P** *(Polynomial) s'il peut être décidé par un algorithme déterministe (i.e. suite d'étapes élémentaires sans choix aléatoire) en un temps polynomial par rapport à la taille de l'instance. On qualifie alors le problème de* polynomial.

Prenons par exemple le problème de la connexité dans un graphe non-orienté. Étant donné un graphe non-orienté à n sommets et m arêtes, le problème consiste à déterminer si toutes les paires de sommets

sont reliées par un chemin. Un simple parcours en profondeur du graphe permet de résoudre ce problème de décision. Si le parcours en profondeur contient tous les sommets du graphe, alors le graphe est connexe. Le temps nécessaire pour ce parcours est en $O(n + m)$, et donc que le problème est dans la classe **P**. Les problèmes de la classe **P** regroupent les problèmes les plus efficacement résolubles.

Définition 13. ***(Classe* NP*)***
Un problème de la classe **NP** *(*Non-déterministe Polynomial*) est un problème de décision, pour lequel il existe un algorithme polynomial en la taille de l'instance capable de tester la validité d'une solution proposée du problème, ceci en un temps polynomial par rapport à la taille de l'instance.*

En d'autres termes, un problème de **NP** peut être résolu en énumérant l'ensemble des solutions possibles et en testant, pour chacune d'elles, leur validité par un algorithme polynomial. Reprenons par exemple le problème du voyageur de commerce dans sa variante décisionnelle énoncée précédemment. Sa résolution peut se réaliser avec deux algorithmes :

- le premier génère l'ensemble des trajets possibles (ce qui est exponentiel) ;
- le second teste les différents trajets afin de déterminer leur validité, i.e. s'ils passent exactement une fois par chaque ville (chaque test en temps polynomial) et si leur longueur est inférieure ou égale à K.

Le problème du *voyageur de commerce* est donc dans la classe **NP**.

3.2.2 Approximations

Nous introduisons ici deux nouvelles notations avant de définir les notions liées aux algorithmes d'approximation.

Notation 7. ***(Coût d'une solution)***
Soient un problème d'optimisation P et I une instance de P. Soit s une solution de I pour le problème P. Nous notons $c_P(s)$ le coût de la solution s (i.e. la valeur selon la mesure à optimiser).

Notation 8. ***(Coût optimal)***
Soit un problème d'optimisation P. Soit I une instance de P. Nous notons $OPT_P(I)$ le coût d'une solution optimale de I pour le problème P.

Afin d'illustrer ces notations, reprenons l'exemple présenté en Figure 3.1. Notons P le problème du voyageur de commerce et I l'instance présentée. Le circuit colorié en pointillé noté s a un coût $c_P(s)$ égal à 27, correspondant ici à sa longueur. De plus, ce circuit est une solution optimale de P puisqu'il n'existe aucun autre circuit de coût inférieur. Nous avons par conséquent $OPT_P(I) = 27$. Nous introduisons maintenant plusieurs définitions concernant les algorithmes d'approximation.

Définition 14. ***(Ratio de performance)***
Étant donné un problème d'optimisation P, pour toute instance I de P, soient $Algo_P(I)$ la solution renvoyée par un algorithme A et $Opt_P(I)$ une solution optimale. Le ratio d'approximation $R_P(I)$ est défini de la manière suivante : $R_P(I) = max(\frac{Algo_P(I)}{Opt_P(I)}, \frac{Opt_P(I)}{Algo_P(I)})$.

Définition 15. ***(Algorithme d'α-approximation)***
Étant donné un problème d'optimisation P et un algorithme A pour P, A est un algorithme d'α-approximation pour P si, pour toute instance I de P, le ratio de performance $R_P(I)$ est borné supérieurement par α.

Définition 16. ***(Classe NPO)***
*La classe **NPO** contient tous les problèmes d'optimisation dont la version décisionnelle est dans **NP**.*

Définition 17. ***(Classe APX)***
*La classe **APX** est une sous-classe de **NPO**. Un problème P de **NPO** appartient à la classe **APX** s'il admet un algorithme polynomial ayant un facteur d'approximation constant, i.e. s'il existe un algorithme d'α-approximation A pour P, pour une constante $\alpha > 1$.*

Définition 18. ***(Classe PTAS)***
*La classe **PTAS** (**P**olynomial-**T**ime **A**pproximation **S**cheme) est une sous-classe de **NPO**. Un problème d'optimisation P appartient à la classe **PTAS** si, pour toute constante $\epsilon > 0$, il existe un algorithme polynomial A qui réalise une approximation de ratio de performance $(1 + \epsilon)$.*

3.2.3 Réductions

Nous introduisons maintenant les notions de réduction, de problème **C-difficile** et **C-complet**.

Définition 19. ***(Réduction)***
Il existe une transformation polynomiale, *appelée* réduction, *d'un problème de décision P_1 en un problème de décision P_2 si, étant donnée une instance I_1 de P_1, il est possible de construire une instance I_2 de P_2 en un temps polynomial (par rapport à la taille de I_1), et telle que la réponse de P_1 à I_1 est « oui »si et seulement si la réponse de P_2 à I_2 est « oui »également.*

Définition 20. ***(Problème C-difficile)***
*Soit C une classe de complexité (par exemple **NP**, **APX**). Un problème de décision P est **C-difficile** si tous les problèmes appartenant à la classe **C** se transforment polynomialement en P.*

Définition 21. ***(Problème C-complet)***
*Soit C une classe de complexité (par exemple **NP**, **APX**). On dit qu'un problème P est **C-complet** si*

- *P est dans la classe **C***
- *P est **C-difficile***

Nous introduisons pour finir la notion de *L-réduction*. La L-réduction est une transformation de problèmes d'optimisation préservant les ratios d'approximation et est utilisée pour prouver l'**APX**-Difficulté d'un problème.

Définition 22. ***(L-réduction)*** *[PY91])*
Soient deux problèmes d'optimisation P_1 et P_2. Une L-réduction *du problème P_1 vers le problème P_2 est une paire de fonctions polynomiales (R,S) satisfaisant tout d'abord les propriétés L_a) et L_b) suivantes :*

- *L_a) Si x est une instance de P_1, alors $R(x)$ est une instance de P_2*
- *L_b) Si x est une instance de P_1 et y est une solution de P_2 pour l'instance $R(x)$, alors $S(y)$ est une solution de P_1 pour x*

De plus, afin de contrôler le ratio d'approximation du problème réduit, une L-réduction doit satisfaire les deux propriétés suivantes :

- *L_c) Il existe une constante positive α telle que si x est une instance de P_1, alors $R(x)$ est une instance de P_2 telle que $OPT_{P_2}(R(x)) \leqslant \alpha.OPT_{P_1}(x)$*
- *L_d) Il existe une constante positive β telle que si s est une solution de P_2 pour l'instance $R(x)$, alors nous avons*
  $|OPT_{P_1}(x) - c_{P_1}(S(s))| \leqslant \beta|OPT_{P_2}(R(x)) - c_{P_2}(s)|$

3.3 Problèmes équivalents

Dans cette section, nous montrons les relations entre les problèmes *BD$_E$*, *BD$_M$*, *BD$_I$*, *ADJ$_E$*, *ADJ$_M$* et *ADJ$_I$* et réduisons ces six problèmes à trois problèmes distincts.

3.3.1 Équivalence des problèmes *BD$_E$* et *ADJ$_E$* (resp. *BD$_M$* et *ADJ$_M$*)

Soient deux génomes G_0 et G_1 pour lesquels les gènes fictifs aux extrémités ont été ajoutés (cf. Section 2.4). Soit un couplage $\mathcal{M}$ de (G_0,G_1) sous un modèle quelconque (*couplage exemplaire*, *couplage maximal* ou *couplage intermédiaire*). Soit la paire $\mathcal{M}$-élaguée (G'_0,G'_1) de (G_0,G_1). Il est évident que, quel que soit le couplage $\mathcal{M}$ de (G_0,G_1), nous avons la propriété suivante :

Propriété 2. ***(Équation nombre de points de cassure et d'adjacences)***

$$Adj(G'_0,G'_1) + Bkp(G'_0,G'_1) = |\mathcal{M}| + 1$$

Par suite, étant donnée une instance, si la taille du couplage $\mathcal{M}$ est fixée, alors trouver un couplage minimisant le nombre de points de cassure est équivalent à trouver un couplage maximisant le nombre d'adjacences, en raison de la Propriété 2. En conséquence, nous pouvons avancer la proposition suivante.

Proposition 1. *Minimiser le nombre de points de cassure sous le modèle* couplage exemplaire *ou* couplage maximal *est équivalent à maximiser le nombre d'adjacences.*

Nous pouvons déduire de la Proposition 1 le théorème suivant.

Théorème 1. *Toute solution optimale de BD$_E$ (resp. BD$_M$) est une solution optimale de ADJ$_E$ (resp. ADJ$_M$), et vice versa.*

En revanche, pour le modèle *couplage intermédiaire*, ce résultat ne tient plus. En effet, la taille du couplage n'est pas fixée à partir des génomes donnés en entrée. Nous illustrons la différence entre les problèmes *BD$_I$* et *ADJ$_I$* par l'exemple suivant :

Exemple 3.8 (Cas pour lequel une solution optimale de *BD$_I$* n'est pas une solution optimale de *ADJ$_I$*).
Soient les deux génomes G_0 et G_1 tels que $G_0 = +0\ +1\ -4\ +2\ -1\ +2\ -3\ +4\ +5$ et $G_1 = +0\ +3\ +1\ +2\ -1\ -3\ +4\ +5$. Soient un couplage exemplaire *$\mathcal{M}'$ de (G_0,G_1) et la paire (G'_0,G'_1) $\mathcal{M}'$-élaguée de (G_0,G_1) tels que $G'_0 = G'_1 = +0\ +1\ +2\ -3\ +4\ +5$; cette solution n'engendre aucun point de cassure et 5 adjacences et est par conséquent optimale pour BD$_I$. Soient maintenant un* couplage intermédiaire *$\mathcal{M}''$ de (G_0,G_1) et la paire (G''_0,G''_1) $\mathcal{M}''$-élaguée de (G_0,G_1) tels que $G''_0 = G''_1 = +0\ +1\ +2\ -1\ -3\ +4\ +5$. Cette solution n'engendre quant à elle aucun point de cassure, mais 6 adjacences. En conclusion, une solution optimale pour BD$_I$ n'est pas nécessairement optimale pour ADJ$_I$.*

3.3.2 Équivalence des problèmes *BD$_E$* et *BD$_I$*

Nous prouvons ici le Théorème 2 ci-dessous, montrant l'équivalence entre les problèmes *BD$_E$* et *BD$_I$*.

Théorème 2. *Les problèmes BD$_E$ et BD$_I$ sont équivalents.*

Démonstration. Soient deux génomes G_0 et G_1. Nous avons nécessairement $OPT_{BDE}(G_0,G_1) \geq OPT_{BDI}(G_0,G_1)$ puisque toute solution du modèle *couplage exemplaire* est aussi solution du modèle *couplage intermédiaire*. Nous devons maintenant prouver $OPT_{BDE}(G_0,G_1) \leq OPT_{BDI}(G_0,G_1)$.

Pour cela, considérons une solution optimale du problème ***BD$_I$***, qui est un couplage de (G_0,G_1) induisant $OPT_{BDI}(G_0,G_1)$ points de cassure. Nous construisons maintenant une solution (G'_0, G'_1) pour le problème ***BD$_E$*** qui induit au plus $OPT_{BDI}(G_0,G_1)$ points de cassure. Cette solution est construite par l'algorithme glouton suivant : tant qu'il existe deux gènes couplés dans G_0 qui appartiennent à la même famille de gènes, nous retirons arbitrairement du couplage l'une de ces deux occurrences, ainsi que le gène de G_1 couplé avec lui. L'algorithme présenté assure clairement que le couplage obtenu est un couplage exemplaire. Nous assurons également que cette solution induit au plus autant de points de cassure que la solution initiale du problème ***BD$_I$***, qui vaut $OPT_{BDI}(G_0,G_1)$. Afin de prouver cela, considérons chaque itération de l'algorithme glouton et montrons que le couplage obtenu induit au plus autant de points de cassure que le couplage initial. Quatre cas doivent être examinés. Nous rappelons que quel que soit un gène a, la notation $a_\blacktriangle$ indique la présence d'un point de cassure après le gène a.

1. $G_0 = \ldots a\ X\ b \ldots$, $G_1 = \ldots a\ X\ b \ldots$ ou $G_1 = \ldots -b\ -X\ -a \ldots$
   Suppression de X dans les deux génomes induisant les génomes $G'_0 = \ldots a\ b \ldots$, $G'_1 = \ldots a\ b \ldots$ ou $G'_1 = \ldots -b\ -a \ldots$ pour lesquels aucun point de cassure n'est ajouté.

2. $G_0 = \ldots a_\blacktriangle\ X\ b \ldots$, $G_1 = \ldots c\ X\ b \ldots$ ou $G_1 = \ldots -b\ -X\ c \ldots$
   Suppression de X dans les deux génomes induisant les génomes $G'_0 = \ldots a_\blacktriangle\ b \ldots$, $G'_1 = \ldots c\ b \ldots$ ou $G'_1 = \ldots -b\ c \ldots$ pour lesquels aucun point de cassure n'est ajouté.

3. $G_0 = \ldots a\ X_\blacktriangle\ b \ldots$, $G_1 = \ldots c\ X\ b \ldots$ ou $G_1 = \ldots b\ -X\ -c \ldots$
   Suppression de X dans les deux génomes induisant les génomes $G'_0 = \ldots a_\blacktriangle\ b \ldots$, $G'_1 = \ldots c\ b \ldots$ ou $G'_1 = \ldots b\ -c \ldots$ pour lesquels aucun point de cassure n'est ajouté.

4. $G_0 = \ldots a_\blacktriangle\ X_\blacktriangle\ b \ldots$
   Quel que soit le génome G_1, la suppression de X dans les deux génomes induit soit le génome $G'_0 = \ldots a\ b \ldots$, soit le génome $G'_0 = \ldots a_\blacktriangle\ b \ldots$
   Dans ces deux cas, le nombre de points de cassure décroît strictement, amenant à une contradiction avec l'optimalité de la solution initiale. Par conséquent, ce cas ne peut arriver.

Dans chacun de ces cas, la suppression de X induit deux génomes pour lesquels aucun point de cassure additionnel n'est ajouté. Ainsi, la solution générée ne peut admettre plus de points de cassure que $OPT_{BDI}(G_0,G_1)$, et par conséquent, nous avons $OPT_{BDE}(G_0,G_1) = OPT_{BDI}(G_0,G_1)$. Les problèmes ***BD$_E$*** et ***BD$_I$*** sont donc équivalents. □

3.4 APX-Difficulté des problèmes *ICOM$_X$* et *ICONS$_X$*

Les problèmes *ICOM$_E$* et *ICOM$_M$* ont été prouvés **NP**-Complets même pour les paires de génomes de type $(1,2)$ dans [CFRV06]. À partir de ces travaux et de la Propriété 1, nous pouvons conclure que *ICOM$_I$* est **NP**-Complet. De plus, il découle de [BR05] que les problèmes *ICONS$_X$* sont également **NP**-Complets. Nous prouvons dans cette partie l'**APX**-Difficulté des problèmes *ICOM$_X$* et *ICONS$_X$*. Ces

études théoriques sont utiles afin d'orienter nos recherches. En effet, nous pourrons écarter, sachant que $ICOM_X$ et $ICONS_X$ sont **APX**-Difficiles, la recherche d'un schéma d'approximation pour ces problèmes.

Afin de prouver l'**APX**-Difficulté des problèmes $ICOM_X$ et $ICONS_X$, nous prouvons tout d'abord que $ICOM_E$ et $ICONS_E$ sont **APX**-Difficiles. Nous généraliserons ensuite ce résultat à tous les problèmes $ICOM_X$ et $ICONS_X$.

3.4.1 APX-Difficulté des problèmes $ICOM_E$ et $ICONS_E$

Dans cette section, nous prouvons le Théorème 3 par une L-réduction [PY91] à partir du problème de couverture minimum d'un graphe (noté *MVC*, pour *Minimum Vertex Cover*) appliqué aux graphes cubiques (MVC_3).

Théorème 3. *Les problèmes $ICOM_E$ et $ICONS_E$ sont **APX**-Difficiles même appliqués à deux génomes G_0 et G_1 tels que $occ(G_0) = 1$ et $occ(G_1) = 2$.*

3.4.1.1 Les problèmes *MVC* et MVC_3

Nous définissons dans cette partie la notion de *couverture* et les problèmes *MVC* et MVC_3.

Définition 23. ***(Couverture)***
Soit un graphe $G = (V,E)$. Un ensemble de sommets $V' \subseteq V$ est appelé une couverture de G s'il existe, pour chaque arête $e \in E$, un sommet $v \in V'$ tel que e est incidente à v.

Définition 24. ***(Couverture minimum)***
Soit un graphe $G = (V,E)$. Un couverture V' de G est dite minimum *s'il n'existe pas de couverture V'' de G telle que $|V''| < |V'|$.*

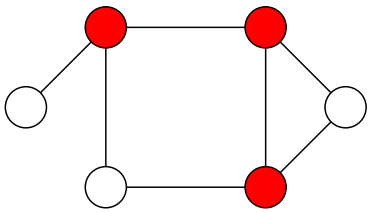


Figure 3.2 – L'ensemble des sommets en rouge, de cardinalité 3, est une couverture minimum puisque (i) chaque arête est incidente à au moins un sommet rouge et (ii) il n'existe aucune couverture de G de cardinalité strictement inférieure à 3.

La Figure 3.2 illustre une couverture minimum. Nous définissons maintenant le problème *MVC* comme suit :

Problème : *MVC* (*Minimum Vertex Cover*)
Entrée : Un graphe G, un entier positif k.
Problème de décision : Existe-t-il une couverture C de G tel que $|C| \leqslant k$?
Problème d'optimisation : Trouver une couverture minimum C de G.

Le problème *MVC* est l'un des 21 problèmes prouvés **NP**-Complets par Karp dans [Kar72]. Nous définissons maintenant les graphes cubiques et le problème MVC_3 comme la variante du problème *MVC* appliqué aux graphes cubiques. Nous commençons par définir le *degré* d'un sommet.

Définition 25. ***(Degré d'un sommet)***
Soit un graphe $G = (V,E)$. Le degré *d'un sommet $v \in V$, noté* degré(v)*, est le nombre d'arêtes de G incidentes au sommet v (i.e. reliées au sommet v).*

Définition 26. ***(Graphe cubique)***
Soit un graphe $G = (V,E)$. Le graphe G est dit cubique *si, pour tout sommet v de V, nous avons degré$(v) = 3$.*

Problème : *MVC*$_3$
Entrée : Un graphe cubique G, un entier positif k.
Problème de décision : Existe-t-il une couverture C de G tel que $\|C\| \leqslant k$?
Problème d'optimisation : Trouver une couverture minimum C de G.

Le problème *MVC*$_3$ a été prouvé **APX**-Difficile dans [AK97]. Nous présentons ci-dessous quelques propriétés des graphes cubiques et du problème *MVC*$_3$. Soient un entier positif k et un graphe cubique G possédant n sommets ($V = \{v_1,v_2,\ldots,v_n\}$) et m arêtes. Nous avons alors les trois propriétés suivantes :

$$n \geqslant 4 \tag{3.1a}$$

$$m = \frac{1}{2}\sum_{i=1}^{n} \text{degré}(v_i) = \frac{3n}{2} \tag{3.1b}$$

$$\text{Toute couverture } C \text{ de } G \text{ satisfait } |C| \geqslant \frac{m}{3} = \frac{n}{2} \tag{3.1c}$$

La Figure 3.3 montre le plus petit graphe cubique. Celui-ci possède quatre sommets (Propriété 3.1a). Afin d'expliquer la Propriété 3.1c, remarquons que dans tout graphe cubique, chaque sommet couvre exactement trois arêtes. Ainsi, un ensemble de k sommets couvre au plus $3k$ arêtes. Par conséquent, une couverture de G doit contenir au moins $\frac{m}{3}$ sommets.

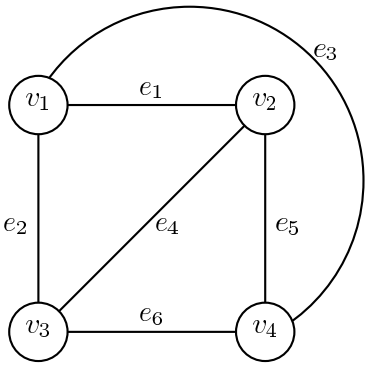


Figure 3.3 – Le graphe cubique de cardinalité minimale (quatre sommets et six arêtes).

3.4.1.2 L-réduction de *MVC*$_3$ vers *ICOM*$_E$ et *ICONS*$_E$

Nous présentons maintenant la L-réduction du problème *MVC*$_3$ vers *ICOM*$_E$ et *ICONS*$_E$. Soit (G,k) une instance du problème *MVC*$_3$, où $G = (V,E)$ est un graphe cubique tel que $V = \{v_1,\ldots,v_n\}$ et

$E = \{e_1, \ldots, e_m\}$. Considérons la fonction polynomiale R qui associe à l'instance (G,k) de *MVC*$_3$ les deux génomes G_0 et G_1 de la manière suivante, où chaque gène est de signe positif (non représenté pour plus de clarté) :

$$G_0 = b_1,b_2 \ldots b_m,x,a_1,C_1,f_1,a_2,C_2,f_2 \ldots a_n,C_n,f_n,y,b_{m+n},b_{m+n-1} \ldots b_{m+1}$$
$$G_1 = y,a_1,D_1,f_1,b_{m+1},a_2,D_2,f_2,b_{m+2} \ldots b_{m+n-1},a_n,D_n,f_n,b_{m+n},x$$

avec :

- Pour tout i, $1 \leqslant i \leqslant n$, $a_i = 6i - 5$
- Pour tout i, $1 \leqslant i \leqslant n$, $f_i = 6i$
- Pour tout i, $1 \leqslant i \leqslant n$, $C_i = (a_i + 1), (a_i + 2), (a_i + 3), (a_i + 4)$
- Pour tout i, $1 \leqslant i \leqslant n + m$, $b_i = 6n + i$
- $x = 7n + m + 1$ et $y = 7n + m + 2$
- Pour tout i, $1 \leqslant i \leqslant n$, D_i est la séquence $(a_i + 3, b_{j_i}, a_i + 1, b_{k_i}, a_i + 4, b_{l_i}, a_i + 2)$, où e_{j_i}, e_{k_i} et e_{l_i} sont les arêtes incidentes à v_i dans G, avec $j_i < k_i < l_i$.

Nous introduisons maintenant quelques définitions afin de simplifier les preuves.

Définition 27. ***(Marqueur)***
Les gènes b_i, $1 \leqslant i \leqslant m$, définis ci-dessus sont appelés des marqueurs.

Remarquons que le génome G_0 ne possède aucun gène dupliqué alors que seuls les marqueurs sont dupliqués dans G_1, ceux-ci apparaissant deux fois. Par conséquent, nous avons $occ(G_0) = 1$ et $occ(G_1) = 2$.

Définition 28. ***(Intervalle robuste)***
Nous appelons un intervalle robuste *un intervalle commun du problème ICOM$_E$ ou un intervalle conservé du problème ICONS$_E$.*

Définition 29. ***(Intervalle trivial)***
Nous appelons un intervalle trivial, un intervalle robuste de longueur 1 *(i.e. un singleton) ou le génome entier.*

Afin d'illustrer la réduction, nous donnons ci-dessous un exemple de la réduction en considérant le graphe cubique de taille minimale.

Exemple 3.9 (Réduction du problème *MVC*$_3$ vers *ICOM$_E$* et *ICONS$_E$*).
À partir du graphe cubique de la Figure 3.3, nous générons par la fonction R les génomes G_0 et G_1 suivants (quel que soit l'entier positif k) :

$$\overbrace{25}^{b_1}\ \overbrace{26}^{b_2}\ \overbrace{27}^{b_3}\ \overbrace{28}^{b_4}\ \overbrace{29}^{b_5}\ \overbrace{30}^{b_6}\ \overbrace{35}^{x}\ 1\ \overbrace{2\,3\,4\,5}^{C_1}\ 6\ 7\ \overbrace{8\,9\,10\,11}^{C_2}\ 12\ 13\ \overbrace{14\,15\,16\,17}^{C_3}\ 18\ 19\ \overbrace{20\,21\,22\,23}^{C_4}\ 24\ \overbrace{36}^{y}\ \overbrace{34}^{b_{10}}\ \overbrace{33}^{b_9}\ \overbrace{32}^{b_8}\ \overbrace{31}^{b_7}$$

$$\underbrace{36}_{y}\ 1\ \underbrace{4\,25\,2\,26\,5\,27\,3}_{D_1}\ 6\ \underbrace{31}_{b_7}\ 7\ \underbrace{10\,25\,8\,28\,11\,29\,9}_{D_2}\ 12\ \underbrace{32}_{b_8}\ 13\ \underbrace{16\,26\,14\,28\,17\,30\,15}_{D_3}\ 18\ \underbrace{33}_{b_9}\ 19\ \underbrace{22\,27\,20\,29\,23\,30\,21}_{D_4}\ 24\ \underbrace{34}_{b_{10}}\ \underbrace{35}_{x}$$

Nous fixons, pour toute cette section, un entier positif k et un graphe cubique $G = (V,E)$ avec $V = \{v_1, \ldots, v_n\}$ et $E = \{e_1, \ldots, e_m\}$. La paire (G,k) étant une instance du problème *MVC*$_3$, nous fixons également les deux génomes G_0, G_1 obtenus par la transformation $R(G,k)$.

Soit une couverture C de G, avec $C = \{v_{i_1}, v_{i_2}, \ldots, v_{i_p}\}$. Nous introduisons la fonction polynomiale F qui associe à C, G_0 et G_1 un couplage exemplaire de (G_0, G_1) de la manière suivante. Dans G_1, tous les marqueurs sont retirés des séquences D_i pour tout $i \notin \{i_1, i_2, \ldots, i_p\}$. Ensuite, pour chaque marqueur présent encore deux fois, l'une des deux occurrences est retirée arbitrairement. Puisque dans G_1, seulement les marqueurs sont dupliqués, nous concluons que $F(C, G_0, G_1)$ est un couplage exemplaire de (G_0, G_1).

Nous définissons la fonction S qui associe à un couplage exemplaire $(G_0, G_1^E, \mathcal{M})$ de (G_0, G_1) la couverture C de G de la manière suivante : $C = \{v_i | 1 \leqslant i \leqslant n \wedge \exists j \in \{1, \ldots, m\}, b_j \in G_1^E[a_i, f_i]\}$. En d'autres termes, nous gardons dans C les sommets v_i de G pour lesquels il existe un marqueur b_j dans $G_1^E[a_i, f_i]$. Nous prouvons maintenant que C est une couverture de G. Considérons une arête e_p de G, avec $1 \leqslant p \leqslant m$. Par construction de G_0 et G_1, il existe un i, $1 \leqslant i \leqslant n$, tel que le gène b_p est situé entre a_i et f_i dans G_1^E. La présence du gène b_p entre a_i et f_i implique que le sommet v_i appartient à C. Nous concluons que chaque arête est incidente à au moins un sommet de C.

Afin de prouver le théorème à la fois pour *ICONS*$_E$ et pour *ICOM*$_E$, nous introduisons les fonctions W et T de la manière suivante. Soit la fonction W définie sur $\{$*ICONS*$_E$, *ICOM*$_E\}$ par $W(pb) = 1$ si $pb =$ *ICONS*$_E$ et $W(pb) = 4$ si $pb =$ *ICOM*$_E$. Soit la fonction T qui associe à un problème $pb \in \{$*ICONS*$_E$, *ICOM*$_E\}$ et au graphe G, le nombre d'intervalles robustes triviaux pour le problème pb de tout couplage exemplaire des génomes G_0 et G_1. Nous avons $T($*ICONS*$_E, G) = 7n + m + 2$ et $T($*ICOM*$_E, G) = 7n + m + 3$. En effet, pour le problème *ICONS*$_E$, il existe $7n + m + 2$ singletons, auxquels nous ajoutons le génome entier pour le problème *ICOM*$_E$.

3.4.1.3 Résultats préliminaires

Afin de prouver le Théorème 3, nous introduisons tout d'abord quatre lemmes intermédiaires. Le premier concerne la localisation des intervalles robustes non triviaux.

Lemme 1. *Pour tout couplage exemplaire $(G_0, G_1^E, \mathcal{M})$ de (G_0, G_1), les intervalles robustes non triviaux de $(G_0, G_1^E, \mathcal{M})$ sont nécessairement contenus dans une séquence de type $a_i C_i f_i$ de G_0, avec $1 \leqslant i \leqslant n$.*

Démonstration. Nous commençons par prouver le lemme appliqué seulement aux intervalles communs, puis nous étendons la propriété aux intervalles conservés. Tout d'abord, nous prouvons que pour tout couplage exemplaire $(G_0, G_1^E, \mathcal{M})$ de (G_0, G_1), chaque intervalle commun I contenant x (resp. y) et tel que $|I| \geqslant 2$, contient également y (resp. x), impliquant de ce fait que I couvre le génome entièrement. Supposons qu'il existe un intervalle commun I_x tel que $|I_x| \geqslant 2$ et tel que I_x contient x. Soit PI_x la permutation de I_x dans G_1^E. De par sa taille (supérieure ou égale à 2), l'intervalle I_x doit nécessairement contenir b_m ou a_1 (ou les deux). Analysons en détail ces deux cas :

(a) Si I_x contient b_m, alors PI_x contient également b_m. Notons qu'il existe un i, $1 \leqslant i \leqslant n$, tel que b_m appartient à D_i dans G_1^E. La permutation PI_x contient alors tous les gènes situés entre D_i et x dans G_1^E, et notamment b_{m+n}. Par conséquent, I_x contient b_{m+n} et par suite y également.

(b) Si I_x contient a_1, alors PI_x le contient également. La permutation PI_x contient alors tous les gènes situés entre a_1 et x, notamment b_{m+n}. Ainsi, I_x contient b_{m+n} et par suite y aussi.

Supposons maintenant que I_y est un intervalle commun tel que $|I_y| \geqslant 2$ et tel que I_y contient y. Soit PI_y la permutation de I_y sur G_1^E. De par sa taille, l'intervalle I_y contient nécessairement b_{m+n} ou f_n (ou les deux). Détaillons ces deux cas :

(a) Si I_y contient b_{m+n}, alors PI_y le contient également. La permutation PI_y contient alors tous les gènes situés entre b_{m+n} et y. Ainsi, PI_y contient toutes les séquences D_i, $1 \leqslant i \leqslant n$. En particulier, PI_y contient tous les marqueurs et par conséquent I_y contient x.
(b) Si I_y contient f_n, alors PI_y contient également f_n. La permutation PI_y contient ainsi tous les gènes situés entre f_n et y, en particulier b_{m+n-1}. Par conséquent I_y contient également b_{m+n-1}, et par suite b_{m+n} aussi, retournant au cas précédent, ce qui implique que I_y contient x.

Nous concluons que chaque intervalle commun non singleton contenant soit x soit y contient nécessairement x et y. Ainsi, par construction de G_1, il existe un seul intervalle de ce type, qui est G_0 lui-même. Par conséquent, tout intervalle commun non trivial est nécessairement, dans G_0, soit strictement à gauche de x, soit entre x et y (non inclus), soit strictement à droite de y. Analysons ces trois cas :

- Soit I un intervalle commun non trivial situé strictement à gauche de x dans G_0. Ainsi, I est une séquence d'au moins deux marqueurs consécutifs. Puisque, quel que soit un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1), chaque marqueur est voisin de gènes qui ne sont pas des marqueurs dans G_1^E, ceci implique une contradiction avec le fait que I est un intervalle commun.
- Soit I un intervalle commun situé strictement à droite de y dans G_0. L'intervalle I est une sous-chaîne de $b_{m+n},\ldots,b_{m+1}$ contenant au moins deux gènes. Ainsi, dans tout couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1), pour chaque (b_{m+i},b_{m+i+1}), avec $1 \leqslant i < n$, nous avons $a_{i+1} \in G_1^E[b_{m+i},b_{m+i+1}]$. Cela contredit le fait que I est strictement à droite de y dans G_0.
- Soit I un intervalle commun situé strictement entre x et y dans G_0. Pour tout couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1), un intervalle commun ne peut contenir, dans G_0, à la fois f_i et a_{i+1} pour i, $1 \leqslant i \leqslant n-1$ (puisque b_{m+1} est situé entre f_i et a_{i+1} dans G_1^E et à droite de x dans G_0). Ainsi, un intervalle commun non trivial de (G_0,G_1^E) est inclus dans une séquence $a_iC_if_i$ dans G_0, $1 \leqslant i \leqslant n$, prouvant le lemme pour les intervalles communs. Par définition, un intervalle conservé est nécessairement un intervalle commun. Par conséquent, un intervalle conservé non trivial de (G_0,G_1^E) est également inclus dans une séquence de type $a_iC_if_i$ dans G_0, $1 \leqslant i \leqslant n$.

□

Le second lemme intermédiaire établit l'impossibilité d'avoir un intervalle robuste sous certaines conditions, et est présenté comme suit :

Lemme 2. *Soient $(G_0,G_1^E,\mathcal{M})$ un couplage exemplaire de (G_0,G_1) et $i \in [1,\ldots,n]$. Soit Δ_i une sous-chaîne de $[a_i+3,a_i+2]_{G_1^E}$ ne contenant aucun marqueur. Si $|\Delta_i| \in \{2,3\}$, alors il n'existe aucun intervalle robuste I de (G_0,G_1^E) tel que Δ_i est une permutation de I.*

Démonstration. Nous prouvons tout d'abord qu'il n'existe aucune permutation I de Δ_i telle que I soit un intervalle commun de (G_0,G_1^E). Par suite, nous montrons qu'il n'existe aucune permutation I de Δ_i telle que I soit un intervalle conservé. Par le Lemme 1, nous savons qu'un intervalle commun non trivial de (G_0,G_1^E) est une sous-chaîne d'une séquence de type $a_iC_if_i$, $1 \leqslant i \leqslant n$. De telles sous-chaînes sont composées uniquement d'entiers consécutifs. Par conséquent, s'il existe une permutation I de Δ_i telle que I est un intervalle commun de (G_0,G_1^E), alors Δ_i est nécessairement une permutation d'entiers consécutifs. Si $|\Delta_i| = 2$, nous avons $\Delta_i = (p,q)$ où p et q ne sont pas consécutifs et si $|\Delta_i| = 3$, nous avons alors $\Delta_i = (a_i+3,a_i+1,a_i+4)$ ou $\Delta_i = (a_i+1,a_i+4,a_i+2)$. Dans ces deux cas, Δ_i n'est pas une permutation d'entiers consécutifs. En conséquence, il n'existe aucune permutation I de Δ_i telle que I est un intervalle commun de (G_0,G_1^E). De plus, puisque tout intervalle conservé est un intervalle commun, il n'existe aucune permutation I de Δ_i telle que I est un intervalle conservé de (G_0,G_1^E). □

Le troisième lemme décrit le nombre d'intervalles robustes non triviaux concernant les chaînes D_i, $1 \leqslant i \leqslant n$.

Lemme 3. *Soit $pb \in \{ICOM_E, ICONS_E\}$. Soit un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) et soit i, $1 \leqslant i \leqslant n$. Alors, seulement un des deux cas suivants apparaît concernant D_i :*

1. *Soit, dans G_1^E, tous les marqueurs de D_i sont retirés, et dans ce cas, il y a exactement $W(pb)$ intervalles robustes non triviaux impliquant D_i.*
2. *Soit, dans G_1^E, au moins un marqueur est gardé dans D_i, et dans ce cas, il n'existe aucun intervalle robuste non trivial impliquant D_i.*

Démonstration. Nous prouvons tout d'abord le lemme pour le problème *ICOM$_E$*, puis nous l'étendons au problème *ICONS$_E$*. Le Lemme 1 implique que chaque intervalle commun non trivial I de (G_0,G_1^E) est contenu dans une sous-chaîne de type $a_iC_if_i$, $1 \leqslant i \leqslant n$. Ainsi, la permutation de I sur G_1^E est contenue dans une sous-chaîne de type $a_iD_if_i$, $1 \leqslant i \leqslant n$. Considérons i, $1 \leqslant i \leqslant n$, et supposons que tous les marqueurs de D_i soient retirés dans G_1^E. Par suite, les séquences $a_iC_if_i$, C_i, a_iC_i et C_if_i sont des intervalles communs de (G_0,G_1^E).

Montrons maintenant qu'il n'existe aucun autre intervalle commun non trivial impliquant D_i. Soit Δ_i une sous-chaîne de $[a_i+3,a_i+2]_{G_1^E}$ telle que $|\Delta_i| \in \{2,3\}$. Par le Lemme 2, nous savons que Δ_i n'est pas un intervalle commun. Les intervalles restants sont (a_i,a_i+3), (a_i,a_i+3,a_i+1), (a_i,a_i+3,a_i+1,a_i+4), (a_i+1,a_i+4,a_i+2,f_i), (a_i+4,a_i+2,f_i) et (a_i+2,f_i). Par construction, aucun d'eux ne peut être un intervalle commun, puisqu'aucun d'eux n'est une permutation d'entiers consécutifs. Par conséquent, il existe seulement quatre intervalles communs non triviaux impliquant D_i dans G_1^E. Parmi ces quatre intervalles communs, seulement $a_iC_if_i$ est aussi un intervalle conservé. Pour finir, si tous les marqueurs sont retirés de D_i, il existe exactement quatre intervalles communs non triviaux et un intervalle conservé non trivial impliquant D_i. Ainsi, étant donné un problème $pb \in \{ICOM_E, ICONS_E\}$, il existe $W(pb)$ intervalles robustes non triviaux impliquant D_i.

Supposons maintenant qu'au moins un marqueur de D_i soit gardé dans G_1^E. Le Lemme 1 prouve que chaque intervalle commun non trivial I de (G_0,G_1^E) est contenu dans une sous-chaîne de type $a_iC_if_i$, $1 \leqslant i \leqslant n$. Puisqu'aucun marqueur n'est présent dans une séquence $a_iC_if_i$, nous déduisons qu'il n'existe aucun intervalle commun non trivial contenant un marqueur. Par suite, un intervalle commun non trivial impliquant seulement D_i doit contenir sur G_1^E une sous-chaîne Δ_i de $[a_i+3,a_i+2]_{G_1^E}$ telle que Δ_i ne contient aucun marqueur. Puisque les marqueurs ne sont pas aux extrémités de $[a_i+3,a_i+2]_{G_1^E}$, nous avons $|\Delta_i| \leqslant 3$. Par le Lemme 2, nous savons que Δ_i n'est pas un intervalle commun. Les intervalles restant à considérer sont les intervalles $a_i\Delta_i$ et Δ_if_i. Par construction de $a_iC_if_i$, ces intervalles ne sont pas des intervalles communs (l'absence du gène a_i+2 pour $a_i\Delta_i$ et du gène a_i+3 pour Δ_if_i implique que ces intervalles ne sont pas des permutations d'entiers consécutifs). Donc, ces intervalles ne peuvent pas non plus être des intervalles conservés. □

Enfin, le dernier lemme établit la correspondance entre la cardinalité d'une couverture et le nombre d'intervalles robustes obtenus.

Lemme 4. *Soit $pb \in \{ICOM_E, ICONS_E\}$.*

1. *Soit une couverture C de G. Alors, le couplage exemplaire $F(C,G_0,G_1)$ de (G_0,G_1) admet au moins $N = W(pb) \cdot n + T(pb,G) - W(pb) \cdot |C|$ intervalles robustes.*
2. *Soit $(G_0,G_1^E,\mathcal{M})$ un couplage exemplaire de (G_0,G_1) et soit la couverture C'de G obtenue par $S(G_0,G_1^E)$. Alors, nous avons $|C'| = \frac{W(pb)\cdot n+T(pb,G)-N}{W(pb)}$, où N est le nombre d'intervalles robustes de (G_0,G_1^E).*

Démonstration.

1. Soit $pb \in \{ICOM_E, ICONS_E\}$. Supposons l'existence d'une couverture C de G. Soit le couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) obtenu par $F(C,G_0,G_1)$. Par construction, nous avons au moins $(n - |C|)$ sous-chaînes D_i dans G_1^E pour lesquelles tous les marqueurs sont retirés. Par le Lemme 3, nous savons que chacune de ces sous-chaînes implique l'existence de $W(pb)$ intervalles robustes non triviaux. Ainsi, nous avons au moins $W(pb)(n - |C|)$ intervalles robustes non triviaux. De plus, le nombre d'intervalles robustes triviaux de (G_0,G_1^E) est exactement $T(pb,G)$. Par suite, nous avons au moins $N = W(pb) \cdot n + T(pb,G) - W(pb) \cdot |C|$ intervalles robustes de (G_0,G_1^E).

2. Soit un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) et soit $n - j$ le nombre de séquences D_i, $1 \leqslant i \leqslant n$, pour lesquelles tous les marqueurs ont été retirés dans G_1^E. Par les Lemmes 1 et 3, le nombre d'intervalles robustes de (G_0,G_1^E) est égal à $N = W(pb) \cdot n + T(pb,G) - W(pb) \cdot j$. Soit la couverture C' obtenue par $S(G_0,G_1^E)$. Chaque marqueur est présent une fois dans G_1^E et ces occurrences proviennent de j séquences D_i. Ainsi, par définition de la fonction S, nous concluons que $|C'| = j = \frac{W(pb)\cdot n + T(pb,G) - N}{W(pb)}$. □

3.4.1.4 Résultat principal

Nous prouvons ici le Théorème 3 en montrant que la paire de fonctions (R,S) définie précédemment est une L-réduction du problème *MVC*$_3$ vers *ICOM*$_E$ (resp. du problème *MVC*$_3$ vers *ICONS*$_E$). Pour cela, nous devons prouver les quatre points suivants (présentés ici pour *ICOM*$_E$) :

- $\mathsf{L_a}$) Si (G,k) est une instance de *MVC*$_3$, alors $R(G,k)$ est une instance de *ICOM*$_E$
- $\mathsf{L_b}$) Si (G,k) est une instance de *MVC*$_3$ et y est une solution de *ICOM*$_E$ sur l'instance $R(G,k)$, alors $S(y)$ est une solution de *MVC*$_3$ sur (G,k)
- $\mathsf{L_c}$) Il existe une constante positive α telle que si (G,k) est une instance de *MVC*$_3$, alors $R(G,k)$ est une instance de *ICOM*$_E$ telle que $OPT_{ICOM_E}(R(G,k)) \leqslant \alpha.OPT_{MVC_3}(G,k)$
- $\mathsf{L_d}$) Il existe une constante positive β telle que si s est une solution de *ICOM*$_E$ sur $R(G,k)$, alors nous avons

$$|OPT_{MVC_3}(G,k) - |S(s)|| \leqslant \beta |OPT_{ICOM_E}(G_0,G_1) - |s||$$

Notons tout d'abord que les propriétés $\mathsf{L_a}$) et $\mathsf{L_b}$) sont immédiates par définition de R et S. De plus, les transformations R et S sont trivialement polynomiales. Considérons un problème $pb \in \{ICOM_E, ICONS_E\}$. Nous prouvons maintenant les propriétés $\mathsf{L_c}$) et $\mathsf{L_d}$). Nous devons tout d'abord prouver qu'il existe une constante positive α telle que $OPT_{pb}(G_0,G_1) \leqslant \alpha.OPT_{MVC_3}(G,k)$.

Par le Lemme 3, nous savons que les séquences de la forme $a_iC_if_i$, $1 \leqslant i \leqslant n$ contiennent soit zéro soit $W(pb)$ intervalles robustes non triviaux. Le Lemme 1 prouve qu'il n'existe pas d'autre intervalle robuste non trivial. Ainsi, nous avons l'inégalité suivante :

$$OPT_{pb}(G_0,G_1) \leqslant \underbrace{T(pb,G)}_{intervalles\ robustes\ triviaux} + W(pb) \cdot n$$

Si $pb = ICOM_E$, nous avons $OPT_{ICOM_E}(G_0,G_1) \leqslant 7n + m + 3 + 4n$ et par les Propriétés 3.1a et 3.1b, nous obtenons :

$$OPT_{ICOM_E}(G_0,G_1) \leqslant \frac{27n}{2} \tag{3.2}$$

Si $pb = \textit{ICONS}_E$, nous avons $OPT_{\textit{ICONS}_E}(G_0,G_1) \leqslant 7n + m + 2 + n$ et par les Propriétés 3.1a et 3.1b, nous avons :

$$OPT_{\textit{ICONS}_E}(G_0,G_1) \leqslant \frac{21n}{2} \tag{3.3}$$

En conséquence de 3.2 et 3.3, nous avons :

$$OPT_{pb}(G_0,G_1) \leqslant \frac{27n}{2}$$

Par la Propriété 3.1c, nous obtenons :

$$OPT_{pb}(G_0,G_1) \leqslant 27.OPT_{\textit{MVC}_3}(G,k)$$

La propriété $\mathsf{L_c}$) est ainsi satisfaite avec $\alpha = 27$.

Prouvons maintenant la propriété $\mathsf{L_d}$). Soit une couverture minimum $C = \{v_{i_1},v_{i_2},\ldots,v_{i_P}\}$ de G. Notons $P = OPT_{\textit{MVC}_3}(G,k) = |C|$. Soit un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) et soit k' le nombre d'intervalles robustes de (G_0,G_1^E). Enfin, soit la couverture C' de G tel que $C' = S(G_0,G_1^E)$. Nous devons trouver une constante β positive telle que $|P - |C'|| \leqslant \beta|OPT_{pb}(G_0,G_1) - k'|$.

Pour $pb \in \{\textit{ICOM}_E, \textit{ICONS}_E\}$, soit N_{pb} le nombre d'intervalles robustes entre les génomes obtenus par $F(C,G_0,G_1)$. Par la première propriété du Lemme 4, nous avons :

$$OPT_{pb}(G_0,G_1) \geqslant N_{pb} \geqslant W(pb) \cdot n + T(pb,G) - W(pb) \cdot P$$

Par la seconde propriété du Lemme 4, nous avons :

$$|C'| = \frac{W(pb) \cdot n + T(pb,G) - k'}{W(pb)}$$

Ainsi, il est suffisant de prouver

$$\exists \beta \geqslant 0, |P - |C'|| \leqslant \beta|W(pb) \cdot n + T(pb,G) - W(pb) \cdot P - k'|$$

Puisque $P \leqslant |C'|$, nous avons

$$|P - |C'|| = |C'| - P = \frac{W(pb) \cdot n + T(pb,G) - k'}{W(pb)} - P$$

et par suite :

$$|P - |C'|| = \frac{1}{W(pb)}(W(pb) \cdot n + T(pb,G) - W(pb) \cdot P - k')$$

Ainsi, $\beta = 1$ suffit dans les deux cas, puisque $W(\textit{ICOM}_E) = 4$ et $W(\textit{ICONS}_E) = 1$, ce qui implique $\frac{1}{W(pb)} \leqslant 1$. Nous avons alors la propriété $\mathsf{L_d}$) avec $\beta = 1$:

$$|OPT_{C_3}(G,k) - |C'|| \leqslant 1 \cdot |OPT_{pb}(G_0,G_1) - k'|$$

Nous avons prouvé que la paire de fonctions (R,S) est une L-réduction du problème *MVC*$_3$ vers *ICOM*$_E$ et *ICONS*$_E$. Cela implique que les problèmes *ICOM*$_E$ et *ICONS*$_E$ sont **APX**-Difficiles, même appliqués à des paires de génomes de type $(1,2)$. Le Théorème 3 est ainsi prouvé. □

3.4.2 Généralisation

Nous étendons par le Corollaire 1 le résultat du Théorème 3 pour les modèles *couplage maximal* et *couplage intermédiaire*.

Corollaire 1. *Les problèmes $ICOM_M$, $ICOM_I$, $ICONS_M$ et $ICONS_I$ sont **APX**-Difficiles même appliqués à des paires de génomes de type* $(1,2)$.

Démonstration. Les modèles *couplage maximal* et *couplage intermédiaire* deviennent identiques au modèle *couplage exemplaire* lorsque l'un des génomes ne contient aucun gène dupliqué (cf. Propriété 1, page 19). Par conséquent, les résultats d'**APX**-Difficulté du problème $ICOM_E$ (resp. $ICONS_E$) présentés par le Théorème 3 tiennent aussi pour les problèmes $ICOM_M$ et $ICOM_I$ (resp. $ICONS_M$ et $ICONS_I$). □

	couplage exemplaire (E)	*couplage maximal* (M)	*couplage intermédiaire* (I)
$ICOM_X$ $ICONS_X$	**NP**-Complet [CFRV06] (instance $(1,2)$) **APX**-Difficile [AFR08a] (instance $(1,2)$)		
BD_X	**NP**-Complet [Bry00] (instance $(1,2)$)	**NP**-Complet [BCF04]*	
ZBD_X	**NP**-Complet [CFZ06] (instance $(3,3)$)	?	?

Table 3.1 – Résultats de complexité algorithmique : ajout de l'**APX**-Difficulté de $ICOM_X$ et $ICONS_X$. * même si une seule famille est dupliquée.

Les travaux exposés dans cette section prouvent que les problèmes $ICOM_X$ et $ICONS_X$ sont **APX**-Difficiles (cf. Table 3.1), engendrant ainsi l'inexistence de schéma d'approximation pour ces problèmes, et ce même pour des paires de génomes de type $(1,2)$. En revanche, des algorithmes d'approximation à ratio constant ne sont pas à exclure, laissant ainsi un vaste champ d'étude. Devant ce problème difficile, nous pourrions tout d'abord nous limiter au cas de génomes équilibrés avant d'attaquer le cas général.

3.5 APX-Difficulté des problèmes BD_X

Nous étudions dans cette partie la complexité des problèmes BD_X. Le problème BD_E a été prouvé **NP**-Complet [Bry00] même appliqué à des paires de génomes de type $(1,2)$. Il en découle directement de [Bry00] par la Propriété 1 (cf. page 19) que BD_M et BD_I sont **NP**-Complets. Nous prouvons ici que les problèmes BD_X sont **APX**-Difficiles également appliqués à des paires de génomes de type $(1,2)$. Ce résultat négatif en terme de complexité implique qu'il n'existe pas de schéma d'approximation pour ces problèmes. En revanche, ce résultat n'induit pas l'inexistence d'algorithmes d'approximation, comme ceux introduits par Kolman et ses collaborateurs dans [GKZ04, KW06].

Nous prouvons maintenant que BD_E est **APX**-Difficile avant de généraliser aux problèmes BD_M, BD_I et ADJ_I.

3.5.1 APX-Difficulté du problème BD_E

Dans cette section, nous prouvons l'**APX**-Difficulté de BD_E, à savoir le théorème suivant :

Théorème 4. *Le problème BD_E est **APX**-Difficile même appliqué à des paires de génomes de type* $(1,2)$.

Afin de prouver le Théorème 4, nous utilisons une L-réduction à partir du problème *MVC*$_3$ (cf. Section 3.4.1.1) vers le problème *BD*$_E$. Soient un entier positif k et un graphe cubique $G = (V,E)$ avec $V = \{v_1, \ldots, v_n\}$ et $E = \{e_1, \ldots, e_m\}$. Pour tout i, $1 \leqslant i \leqslant n$, soient e_{f_i}, e_{g_i} et e_{h_i} les trois arêtes incidentes à v_i dans G avec $f_i < g_i < h_i$. Soit la transformation polynomiale R' qui associe à l'instance (G,k) de *MVC*$_3$ les génomes G_0 et G_1 suivants, où chaque gène est de signe positif (non représenté ici pour plus de clarté) :

$$G_0 = a_0\ a_1\ b_1\ a_2\ b_2 \ldots a_n\ b_n\ c_1\ d_1\ c_2\ d_2 \ldots c_m\ d_m\ c_{m+1}$$

$$G_1 = a_0\ a_n\ d_{f_n}\ d_{g_n}\ d_{h_n}\ b_n \ldots a_2\ d_{f_2}\ d_{g_2}\ d_{h_2}\ b_2\ a_1\ d_{f_1}\ d_{g_1}\ d_{h_1}\ b_1\ c_1\ c_2 \ldots c_m\ c_{m+1}$$

avec :

- Pour tout i, $0 \leqslant i \leqslant n$, $a_i = i$
- Pour tout i, $1 \leqslant i \leqslant n$, $b_i = n + i$
- Pour tout i, $1 \leqslant i \leqslant m + 1$, $c_i = 2n + i$
- Pour tout i, $1 \leqslant i \leqslant m$, $d_i = 2n + m + 1 + i$

Remarquons qu'il n'existe aucun gène dupliqué dans G_0, ce qui implique $occ(G_0) = 1$. Dans G_1, seuls les gènes d_i, $1 \leqslant i \leqslant m$, sont dupliqués et ceux-ci apparaissent exactement deux fois chacun. Par conséquent, nous avons $occ(G_1) = 2$.

Afin d'illustrer la réduction, nous donnons ci-dessous un exemple de la réduction en considérant le graphe cubique de taille minimale.

Exemple 3.10 (Réduction du problème *MVC*$_3$ vers *BD*$_E$).
À partir du graphe cubique de la Figure 3.3, nous générons par la fonction R' les génomes G_0 et G_1 suivants (quel que soit l'entier positif k) :

$\overbrace{a_0}^{0}\ \overbrace{a_1}^{1}\ \overbrace{b_1}^{5}\ \overbrace{a_2}^{2}\ \overbrace{b_2}^{6}\ \overbrace{a_3}^{3}\ \overbrace{b_3}^{7}\ \overbrace{a_4}^{4}\ \overbrace{b_4}^{8}\ \overbrace{c_1}^{9}\ \overbrace{d_1}^{16}\ \overbrace{c_2}^{10}\ \overbrace{d_2}^{17}\ \overbrace{c_3}^{11}\ \overbrace{d_3}^{18}\ \overbrace{c_4}^{12}\ \overbrace{d_4}^{19}\ \overbrace{c_5}^{13}\ \overbrace{d_5}^{20}\ \overbrace{c_6}^{14}\ \overbrace{d_6}^{21}\ \overbrace{c_7}^{15}$

$\underbrace{0}_{a_0}\ \underbrace{4}_{a_4}\ \underbrace{18\ 20\ 21}_{d_4\ d_5\ d_6}\ \underbrace{8}_{b_4}\ \underbrace{3}_{a_3}\ \underbrace{17\ 19\ 21}_{d_2\ d_4\ d_6}\ \underbrace{7}_{b_3}\ \underbrace{2}_{a_2}\ \underbrace{16\ 19\ 20}_{d_1\ d_4\ d_5}\ \underbrace{6}_{b_2}\ \underbrace{1}_{a_1}\ \underbrace{16\ 17\ 18}_{d_1\ d_2\ d_2}\ \underbrace{5}_{b_1}\ \underbrace{9\ 10\ 11\ 12\ 13\ 14\ 15}_{c_1\ \ldots\ c_7}$

Nous fixons, pour toute cette section, un entier positif k et un graphe cubique $G = (V,E)$ avec $V = \{v_1, \ldots, v_n\}$ et $E = \{e_1, \ldots, e_m\}$. La paire (G,k) étant une instance du problème *MVC*$_3$, nous fixons les deux génomes G_0, G_1 obtenus par la transformation $R'(G,k)$.

Afin de prouver le Théorème 4, il faut prouver les quatre points suivants :

- $\mathsf{L_a}$) Si (G,k) est une instance de *MVC*$_3$, alors $R'(G,k)$ est une instance *BD*$_E$
- $\mathsf{L_b}$) Si (G,k) est une instance de *MVC*$_3$ et y est une solution de *BD*$_E$ pour l'instance $R'(G,k)$, alors $S'(y)$ est une solution de *MVC*$_3$ pour (G,k)
- $\mathsf{L_c}$) Il existe une constante positive α telle que si (G,k) est une instance de *MVC*$_3$, alors $R'(G,k)$ est une instance de *BD*$_E$ telle que $OPT_{BD_E}(R'(G,k)) \leqslant \alpha.OPT_{MVC_3}(G,k)$
- $\mathsf{L_d}$) Il existe une constante positive β telle que si s est une solution de *BD*$_E$ pour $R'(G,k)$, alors nous avons $|OPT_{MVC_3}(G,k) - |S'(s)|| \leqslant \beta |OPT_{BD_E}(G_0,G_1) - |s||$

Soit une couverture C de G. Nous définissons la fonction polynomiale F' qui associe à C, G_0 et G_1 le couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) de la façon suivante. Pour chaque i tel que $v_i \notin C$, nous retirons de G_1 les gènes d_{f_i}, d_{g_i} et d_{h_i}. Puis, pour chaque $1 \leqslant j \leqslant m$ tel que d_j apparaît encore deux fois dans G_1, nous retirons arbitrairement l'un de ces deux occurrences afin d'obtenir le génome G_1^E. Par conséquent, $(G_0,G_1^E,\mathcal{M})$ est un couplage exemplaire de (G_0,G_1) puisque seuls les gènes de la forme d_j, $1 \leqslant j \leqslant m$, sont dupliqués.

Étant donné un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1), soit la fonction polynomiale S' qui associe à la paire (G_0,G_1^E) l'ensemble $C = \{v_i | 1 \leqslant i \leqslant n, a_i$ et b_i ne sont pas consécutifs dans $G_1^E\}$. Nous affirmons que C est une couverture de G. En effet, soit une arête e_p de G, avec $1 \leqslant p \leqslant m$. Le génome G_1^E contient une occurrence du gène d_p puisque $(G_0,G_1^E,\mathcal{M})$ est un couplage exemplaire de (G_0,G_1). Par construction, il existe i, $1 \leqslant i \leqslant n$, tel que d_p est dans $G_1^E[a_i,b_i]$ et tel que e_p est incidente à v_i. La présence de d_p dans $G_1^E[a_i,b_i]$ implique que le sommet v_i appartient à C. Nous pouvons alors conclure que chaque arête de G est incidente à au moins un sommet de C.

Notons que les propriétés $\mathsf{L_a}$) et $\mathsf{L_b}$) sont satisfaites par définition des transformations polynomiales R' et S'. Nous proposons maintenant quatre lemmes intermédiaires afin de prouver les propriétés $\mathsf{L_c}$) et $\mathsf{L_d}$), induisant ainsi que la paire de fonctions (R',S') est une L-réduction du problème *MVC*$_3$ vers *BD*$_E$. Le premier lemme définit le nombre de points de cassure obtenus depuis une couverture donnée.

Lemme 5. *Soit une couverture C de G et soit le couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ obtenu par la transformation $F'(G_0,G_1,C)$. Nous avons $Bkp(G_0,G_1^E) \leqslant n + 2m + |C| + 1$.*

Démonstration. Listons les points de cassure entre les deux génomes G_0 et G_1^E obtenus par la transformation $F'(G_0,G_1,C)$. Les duos (b_i,a_{i+1}), $1 \leqslant i \leqslant n-1$, et (b_n,c_1) induisent un point de cassure chacun. Pour tout $1 \leqslant i \leqslant m$, chaque duo de la forme (c_i,d_i) (resp. (d_i,c_{i+1})) induit également un point de cassure. Pour tout $1 \leqslant i \leqslant n$ tel que $v_i \in C$, le duo (a_i,b_i) induit au plus un point de cassure. Enfin, le duo (a_0,a_1) induit un point de cassure. En somme, nous avons au plus $n + 2m + |C| + 1$ points de cassure entre G_0 et G_1^E. □

Le second lemme définit la cardinalité d'une couverture obtenue depuis un couplage exemplaire donné.

Lemme 6. *Soit un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) et soit la couverture C' de G obtenue par $S'(G_0,G_1^E)$. Nous avons $|C'| = Bkp(G_0,G_1^E) - n - 2m - 1$.*

Démonstration. Soit un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) et soit une couverture C' obtenue par $S'(G_0,G_1^E)$. Pour tout couplage exemplaire de (G_0,G_1), les points de cassure suivants existent obligatoirement : le duo (a_0,a_1) ; les duos (c_i,d_i) et (d_i,c_{i+1}) pour tout i, $1 \leqslant i \leqslant m$; les duos (b_i,a_{i+1}) pour tout i, $1 \leqslant i \leqslant n-1$; le duo (b_n,c_1). Ainsi, il existe au moins $n + 2m + 1$ points de cassure. Les autres points de cassure possibles sont induits par les duos de la forme (a_i,b_i), qui sont au nombre de $Bkp(G_0,G_1^E) - n - 2m - 1$. Par construction de C', la cardinalité de C' est égale au nombre de points de cassure induits par les duos de la forme (a_i,b_i). En conséquence, nous avons $|C'| = Bkp(G_0,G_1^E) - n - 2m - 1$. □

Le lemme suivant prouve la propriété $\mathsf{L_c}$) de la L-réduction.

Lemme 7. *Nous avons $OPT_{BD_E}(G_0,G_1) \leqslant 12 \cdot OPT_{MVC_3}(G,k)$.*

Démonstration. Par construction des génomes G_0 et G_1, tout couplage exemplaire de (G_0,G_1) contient $2n + 2m + 1$ gènes dans chaque génome. Ainsi, nous avons $OPT_{BD_E}(G_0,G_1) \leqslant 2n + 2m + 1$. Par les Propriétés 3.1a et 3.1b, nous avons alors $OPT_{BD_E}(G_0,G_1) \leqslant 6n$. Enfin, par la Propriété 3.1c, nous concluons que $OPT_{BD_E}(G_0,G_1) \leqslant 12 \cdot OPT_{MVC_3}(G,k)$. □

Enfin, la propriété $\mathsf{L_d}$) de la L-réduction est prouvée par le lemme suivant :

Lemme 8. *Soit un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) et soit la couverture C' de G obtenue par $S'(G_0,G_1^E)$. Alors, nous avons $|OPT_{MVC_3}(G,k) - |C'|| \leqslant |OPT_{BD_E}(G_0,G_1) - Bkp(G_0,G_1^E)|$.*

Démonstration. Soit un couplage exemplaire $(G_0,G_1^E,\mathcal{M})$ de (G_0,G_1) et soit la couverture C' de G obtenu par $S'(G_0,G_1^E)$. Soit une couverture C de G tel que $|C| = OPT_{MVC_3}(G,k)$. Les problèmes *MVC*$_3$ et *BD*$_E$ étant des problèmes de minimisation, nous savons que $OPT_{MVC_3}(G,k) \leqslant |C'|$ et que $OPT_{BD_E}(G_0,G_1) \leqslant Bkp(G_0,G_1^E)$. Ainsi, il est suffisant de prouver :

$$|C'| - OPT_{MVC_3}(G,k) \leqslant Bkp(G_0,G_1^E) - OPT_{BD_E}(G_0,G_1)$$

Par le Lemme 5, nous avons $Bkp(F'(G_0,G_1C)) \leqslant n + 2m + 1 + OPT_{MVC_3}(G,k)$, ce qui implique :

$$OPT_{BD_E}(G_0,G_1) \leqslant B(F'(G_0,G_1,C)) \leqslant n + 2m + 1 + OPT_{MVC_3}(G,k)$$

Et par suite :

$$Bkp(G_0,G_1^E) - OPT_{BD_E}(G_0,G_1) \geqslant Bkp(G_0,G_1^E) - n - 2m - 1 - OPT_{MVC_3}(G,k) \tag{3.4}$$

Par le Lemme 6, nous avons $|C'| = Bkp(G_0,G_1^E) - n - 2m - 1$ ce qui implique :

$$|C'| - OPT_{MVC_3}(G,k) = Bkp(G_0,G_1^E) - n - 2m - 1 - OPT_{MVC_3}(G,k) \tag{3.5}$$

Enfin, par 3.4 et 3.5, nous obtenons $|C'| - OPT_{MVC_3} \leqslant Bkp(G_0,G_1^E) - OPT_{BD_E}(G_0,G_1)$. □

Les Lemmes 7 et 8 prouvent les propriétés $\mathsf{L_c}$) et $\mathsf{L_d}$). Ainsi, la paire (R',S') est une L-réduction du problème *MVC*$_3$ vers *BD*$_E$ avec les constantes $\alpha = 12$ et $\beta = 1$. Par conséquent, le problème *BD*$_E$ est **APX**-Difficile même appliqué à des paires de génomes de type $(1,2)$, et le Théorème 4 est ainsi prouvé. □

3.5.2 Généralisation

Nous étendons par le Corollaire 2 le résultat du Théorème 4 pour les problèmes *BD*$_M$, *BD*$_I$ et *ADJ*$_I$.

Corollaire 2. *Les problèmes BD*$_M$*, BD*$_I$ *et ADJ*$_I$ *sont* ***APX****-Difficiles même appliqués à des paires de génomes de type* $(1,2)$.

Démonstration. Les modèles *couplage maximal* et *couplage intermédiaire* deviennent identiques au modèle *couplage exemplaire* lorsque l'un des génomes ne contient aucun gène dupliqué (cf. Propriété 1, page 19). Par conséquent, le résultat d'**APX**-Difficulté du problème *BD*$_E$ du Théorème 4 tient aussi pour les problèmes *BD*$_M$ et *BD*$_I$. De plus, les problèmes *BD*$_E$ et *ADJ*$_E$ étant équivalents (cf. Théorème 1, page 31), il découle également de la Propriété 1 que *ADJ*$_I$ est **APX**-Difficile. □

Les résultats obtenus dans cette section montrent que les problèmes *BD*$_X$ sont **APX**-Difficiles (cf. Table 3.2), ce qui implique l'inexistence de schémas d'approximation pour ces problèmes, même pour des paires de génomes de type $(1,2)$. Néanmoins, nous pouvons continuer la recherche d'algorithmes d'approximation afin d'améliorer les ratios d'approximation proposés pour des génomes équilibrés dans [GKZ04, KW06], ou bien, afin de lever l'hypothèse de génomes équilibrés.

<table>
<tr><th></th><th>couplage exemplaire (E)</th><th>couplage maximal (M)</th><th>couplage intermédiaire (I)</th></tr>
<tr><td>ICOM_X
ICONS_X</td><td colspan="3">NP-Complet [CFRV06] (instance $(1,2)$)
APX-Difficile [AFR08a] (instance $(1,2)$)</td></tr>
<tr><td>BD_X</td><td colspan="3">NP-Complet [Bry00] (instance $(1,2)$)
NP-Complet [BCF04]*
APX-Difficile [AFR08a] (instance $(1,2)$)</td></tr>
<tr><td>ZBD_X</td><td>NP-Complet [CFZ06] (instance $(3,3)$)</td><td>?</td><td>?</td></tr>
</table>

Table 3.2 – Résultats de complexité algorithmique : ajout de l'**APX**-Difficulté de BD_X. * même si une seule famille est dupliquée.

3.6 Complexité des problèmes ZBD_X

Dans cette section, nous étudions la complexité des problèmes ZBD_X (définis Section 2.4.3). Chen et al. montrent dans [CFZ06] que décider s'il existe un couplage exemplaire de deux génomes n'impliquant aucun point de cassure est **NP**-Complet même pour les paires de génomes de type $(3,3)$. Ce résultat implique que le problème BD_E n'admet pas d'approximation en temps polynomial sur ces mêmes instances, sauf si $\mathbf{P} = \mathbf{NP}$. Nous complétons ici les résultats de [CFZ06] en prouvant que ZBD_E est **NP**-Complet, même si aucune famille n'apparaît plus de deux fois dans G_0 (le nombre maximum de gènes dupliqués est toutefois non borné dans G_1). Ce résultat est ensuite étendu au modèle *couplage intermédiaire*. Notons que de récents travaux [BFSV09] montrent que ZBD_E est **NP**-Complet même pour les paires de génomes de type $(2,2)$, ce qui améliore ainsi nos connaissances sur le problème ZBD_E.

Nous proposons ensuite un algorithme exponentiel, mais applicable sous certaines conditions, pour décider s'il existe un couplage exemplaire d'une paire de génomes de type $(2,2)$ n'impliquant aucun point de cassure, un problème de complexité ouverte au début de notre travail mais résolu récemment dans [BFSV09]. Enfin, nous montrons que décider s'il existe un couplage maximal de deux génomes n'impliquant aucun point de cassure est résoluble en temps polynomial et ainsi, que les résultats négatifs d'approximation (ceux obtenus pour les modèles *couplage exemplaire* et *couplage intermédiaire*) ne sont plus valables pour le modèle *couplage maximal*.

L'observation suivante sera utile pour les différentes preuves de cette section.

Observation 1. *Soient deux génomes G_0 et G_1. S'il existe un couplage exemplaire $(G_0^E, G_1^E, \mathcal{M})$ de (G_0,G_1) n'engendrant aucun point de cassure de (G_0^E, G_1^E), alors nous avons soit $G_0^E = G_1^E$, soit $-(G_0^E)^r = G_1^E$, où $-(G_0^E)^r$ correspond au génome G_0 inversé (inversion de l'ordre des gènes et des signes). La même observation peut être faite pour les modèles couplage maximal et couplage intermédiaire.*

3.6.1 NP-Complétude de ZBD_E

Nous complétons le résultat de [CFZ06] qui montre que ZBD_E est **NP**–complet même pour les paires de génomes de type $(3,3)$, par le théorème suivant.

Théorème 5. *Soient deux génomes G_0 et G_1. Le problème ZBD_E est* **NP***–complet même si nous avons $occ(G_0) = 2$.*

Démonstration. Le problème *ZBD$_E$* est évidemment dans la classe **NP**. En effet, étant donné un couplage exemplaire, le calcul du nombre de points de cassure se réalise en temps polynomial. La réduction que nous utilisons pour prouver le Théorème 5 se fait à partir du problème *MVC* défini Section 3.4.1.1. Soit une instance arbitraire de *MVC* donnée par un graphe $G = (V,E)$ et un entier positif k. Soit $V = \{v_1, v_2, \ldots, v_n\}$ et $E = \{e_1, e_2, \ldots, e_m\}$. Dans le reste de la preuve, les éléments de V (resp. E) seront interprétés soit comme des sommets (resp. arêtes) soit comme des gènes, ceci selon le contexte. L'instance correspondante (G_0, G_1) de *ZBD$_E$* est définie de la façon suivante :

$$G_0 = v_1\ X_1\ v_2\ X_2 \ldots v_n\ X_n$$
$$G_1 = Y[1]\ Y[2]\ \ldots\ Y[k]\ Y_V.$$

Pour tout i, $1 \leqslant i \leqslant n$, X_i est défini par $X_i = e_{i_1}\ e_{i_2}\ \ldots\ e_{i_j}$, où $e_{i_1}, e_{i_2}, \ldots, e_{i_j}$, $i_1 < i_2 < \ldots < i_j$, sont les arêtes incidentes au sommet v_i. Les chaînes $Y[i]$, $1 \leq i \leq k$, sont toutes identiques et sont définis par $Y[i] = Y_V\ Y_E$ avec $Y_V = v_1\ v_2\ \ldots\ v_n$ et $Y_E = e_1\ e_2\ \ldots\ e_m$.

Notons qu'aucun gène n'est présent plus de deux fois dans G_0 (les gènes v_i apparaissent une seule fois et les gènes e_i apparaissent deux fois). Cependant, le nombre d'occurrences de chaque gène dans G_1 est inférieur ou égal à $k + 1$. De plus, tous les gènes ont un signe positif, et en conséquence de l'Observation 1, nous pouvons considérer uniquement les couplages exemplaires (G_0^E, G_1^E) de (G_0, G_1) tels que $G_0^E = G_1^E$.

Il est immédiat que la construction de G_0 et G_1 peut être réalisée en un temps polynomial. Nous montrons maintenant qu'il existe une couverture de G de taille k si et seulement si il existe un couplage exemplaire de G_0 et G_1 n'engendrant aucun point de cassure.

($\Rightarrow$) Supposons qu'il existe une couverture C' de G de taille k, avec $C' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ tel que $i_1 < i_2 < \ldots < i_k$. Nous définissons i_0 égal à 0. À partir de la couverture C', nous construisons un couplage exemplaire (G_0^E,G_1^E) de (G_0,G_1) en six étapes, illustrées par la Figure 3.4, et détaillons ci-dessous la procédure.

Nous obtenons G_0^E à partir de G_0 par les deux premières étapes :

$P1$. Nous retirons dans G_0 toutes les chaînes X_i telles que $v_i \notin C'$.

$P2$. Pour tout j, $1 \leq j \leq m$, si le gène e_j apparaît encore deux fois, nous supprimons la deuxième occurrence (cette seconde étape concerne les arêtes connectées à deux sommets de C').

Nous précisons maintenant la construction de G_1^E en quatre étapes successives en traitant, pour chaque $1 \leq j \leq k$, la chaîne $Y[j] = Y_V\ Y_E$:

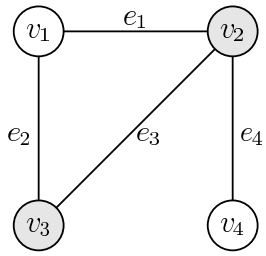
$P3$. Nous supprimons dans Y_V tous les gènes sauf v_{i_j} et les gènes $v_\ell \notin C'$ tels que $i_{j-1} < \ell < i_j$.

$P4$. Nous supprimons dans Y_E les gènes e_ℓ pour lesquels l'arête e_ℓ n'est pas incidente à v_{i_j}.

$P5$. Nous retirons dans Y_E les gènes e_ℓ pour lesquels l'arête e_ℓ est incidente à v_{i_j} et à un plus petit sommet de C' (*i.e.* $e_\ell = \{v_{i_{j'}}, v_{i_j}\}$ avec $j' < j$ et $v_{i_{j'}} \in C'$).

$P6$. Enfin, nous retirons dans la dernière chaîne $Y_V = v_1\ v_2\ \ldots\ v_n$ tous les gènes sauf les gènes v_ℓ tels que $v_\ell \notin C'$ et tels que $i_k < \ell$.

Par construction de G_0^E, chaque famille apparaît trivialement exactement une fois dans les génomes. En effet, puisque C' est une couverture de G, l'étape $P1$ ne peut en aucun cas retirer toutes les occurrences d'une même famille. La construction de G_1^E est réalisée afin d'obtenir $G_0^E = G_1^E$. Remarquons que nous avons gardé par P_1 exactement k chaînes X_i (dont certaines ont pu perdre des éléments par



Graphe G, $k = 2$, $V' = \{v_2, v_3\}$

Construction de G_0 et G_1

$$G_0 : v_1 \overbrace{e_1\ e_2\ e_3}^{X_1} v_2 \overbrace{e_1\ e_3\ e_4}^{X_2} v_3 \overbrace{e_2\ e_3}^{X_3} v_4 \overbrace{e_4}^{X_4}$$

$$G_1 : \underbrace{\underbrace{v_1\ v_2\ v_3\ v_4}_{Y_V}\underbrace{e_1\ e_2\ e_3\ e_4}_{Y_E}\underbrace{v_1\ v_2\ v_3\ v_4}_{Y_V}\underbrace{e_1\ e_2\ e_3\ e_4}_{Y_E}}_{Y[1]}\underbrace{\underbrace{v_1\ v_2\ v_3\ v_4}_{Y_V}}_{Y[2]}$$

construction de G_0^E

- étape P_1 : $G_0^E : v_1\ \not e_1\ \not e_2\ \not e_3\ v_2\ e_1\ e_3\ e_4\ v_3\ e_2\ e_3\ v_4\ \not e_4$
- étape P_2 : $G_0^E : v_1\ \not e_1\ \not e_2\ \not e_3\ v_2\ e_1\ e_3\ e_4\ v_3\ e_2\ \not e_3\ v_4\ \not e_4$

construction de G_1^E

- étape P_3 : $G_1^E : v_1\ v_2\ \not v_3\ \not v_4\ e_1\ e_2\ e_3\ e_4\ \not v_1\ \not v_2\ v_3\ \not v_4\ e_1\ e_2\ e_3\ e_4\ v_1\ v_2\ v_3\ v_4$
- étape P_4 : $G_1^E : v_1\ v_2\ \not v_3\ \not v_4\ e_1\ \not e_2\ e_3\ e_4\ \not v_1\ \not v_2\ v_3\ \not v_4\ \not e_1\ e_2\ e_3\ \not e_4\ v_1\ v_2\ v_3\ v_4$
- étape P_5 : $G_1^E : v_1\ v_2\ \not v_3\ \not v_4\ e_1\ \not e_2\ e_3\ e_4\ \not v_1\ \not v_2\ v_3\ \not v_4\ \not e_1\ e_2\ \not e_3\ \not e_4\ v_1\ v_2\ v_3\ v_4$
- étape P_6 : $G_1^E : v_1\ v_2\ \not v_3\ \not v_4\ e_1\ \not e_2\ e_3\ e_4\ \not v_1\ \not v_2\ v_3\ \not v_4\ \not e_1\ e_2\ \not e_3\ \not e_4\ \not v_1\ \not v_2\ \not v_3\ v_4$

Figure 3.4 – Réduction de *MVC* vers *ZBD$_E$*. Exemple de construction d'un couplage exemplaire n'induisant aucun point de cassure.

$P2$), celles-ci séparées par un ou plusieurs gènes de type v_x, $1 \leqslant x \leqslant n$. Les étapes P_4 et P_5 retirent les éléments des k chaînes Y_E de façon à obtenir une copie des k chaînes X_i gardées dans G_0^E. Les étapes P_3 et P_6 sont quant à elles nécessaires pour que, dans G_1^E, les gènes v_x se retrouvent exactement aux mêmes positions que dans G_0^E. Nous obtenons ainsi $G_0^E = G_1^E$ et nous concluons par conséquent qu'il existe un couplage exemplaire de (G_0,G_1) n'induisant aucun point de cassure, en l'occurrence (G_0^E, G_1^E).

($\Leftarrow$) Inversement, supposons qu'il existe un couplage exemplaire de (G_0,G_1) n'induisant aucun point de cassure. Puisque tous les gènes ont un signe positif, alors il existe un couplage exemplaire (G_0^E, G_1^E) de (G_0, G_1) tel que $G_0^E = G_1^E$. Le génome G_1^E peut être décrit par :

$$G_1^E = Y_V[1]\ Y_E[1]\ Y_V[2]\ Y_E[2]\ \ldots Y_V[k]\ Y_E[k]\ Y_V[k+1]$$

où $Y_V[i]$, $1 \leq i \leq k+1$, est une chaîne d'éléments de C et $Y_E[i]$, $1 \leq i \leq k$, est une chaîne d'éléments de E, C et E étant interprétés comme des alphabets. Nous définissons maintenant $C' \subseteq C$ de la manière suivante : $v_i \in C'$, $1 \leqslant i \leqslant n$, si et seulement si le gène v_i apparaît en dernière position dans une chaîne $Y_V[j]$, $1 \leq j \leq k$.

Prouvons maintenant que C' est une couverture de G de taille au plus k. Par construction, $|C'| \leq k$

(nous aurons même $|C'| < k$ si au moins une des chaînes $Y_V[j]$, $1 \leq j \leq k$, est une chaîne vide). Soit e_ℓ une arête de G. Remarquons que le dernier gène v_i, $1 \leqslant i \leqslant n$, précédant e_ℓ dans G_1^E appartient nécessairement à C' par construction. Puisqu'aucun gène v_p, $1 \leqslant p \leqslant n$, n'est dupliqué dans G_0 et que $G_0^E = G_1^E$, nous savons que le gène e_ℓ de G_1^E est couplé avec un gène de X_i dans G_0^E, induisant par construction de G_0 que e_ℓ est incidente à v_i. Ainsi, quelle que soit l'arête e_ℓ, e_ℓ est incidente à au moins un sommet de C'. Il s'ensuit alors que C' est une couverture de G de taille au plus k. □

Ce résultat implique directement qu'il n'existe pas d'algorithme d'approximation à ratio constant pour le problème *BD*$_E$ pour les paires de génomes de type $(2,x)$ avec x non borné, sauf si **P** = **NP**. Depuis ce travail, Blin et al. ont montré dans [BFSV09] que *ZBD*$_E$ est également **NP**-Complet même pour les paires de génomes de type $(2,2)$ (cf. Table 3.3 pour un récapitulatif des résultats de complexité algorithmique) .

	couplage exemplaire (E)	*couplage maximal* (M)	*couplage intermédiaire* (I)
ICOM$_X$ *ICONS*$_X$	**NP**-Complet [CFRV06] (instance $(1,2)$) **APX**-Difficile [AFR08a] (instance $(1,2)$)		
BD$_X$	**NP**-Complet [Bry00] (instance $(1,2)$) **NP**-Complet [BCF04]*		
ZBD$_X$	**NP**-Complet [CFZ06] (instance $(3,3)$) **NP**-Complet [AFR$^+$09] (instance $(2,k)$) **NP**-Complet [BFSV09] (instance $(2,2)$)	?	?

Table 3.3 – Résultats de complexité algorithmique : ajout de la **NP**-Complétude de *ZBD*$_E$ pour les paires de génomes de type $(2,k)$ où k est non borné. * même si une seule famille est dupliquée.

Nous proposons maintenant un algorithme exponentiel pour résoudre le problème *ZBD*$_E$ pour les paires de génomes de type $(2,2)$, et dont la complexité dépend du nombre de gènes apparaissant deux fois dans G_0 et dans G_1.

Proposition 2. *Le problème ZBD*$_E$ *pour les paires de génomes de type* $(2,2)$ *(aucun gène n'apparaît plus de deux fois dans* G_0 *et dans* G_1*) est résoluble en* $O^*(1.6182^{2k})$*, où* k *est borné supérieurement par le nombre de gènes apparaissant exactement deux fois dans* G_0 *et dans* G_1*.*

Démonstration. Selon l'Observation 1, pour toute instance (G_0, G_1), nous pouvons considérer seulement les couplages exemplaires (G_0^E, G_1^E) de (G_0,G_1) tels que $G_0^E = G_1^E$ ou $-(G_0^E)^r = G_1^E$, où $-(G_0^E)^r$ est la séquence inverse de G_0^E (inversion de l'ordre des gènes et inversion des signes). Considérons tout d'abord le cas $G_0^E = G_1^E$ (le cas $-(G_0^E)^r = G_1^E$ est identique à une inversion près et sera discuté brièvement à la fin de la preuve).

Soit une instance (G_0, G_1) de type $(2, 2)$ du problème *ZBD*$_E$. Nous supposons que toute famille est présente sur les deux génomes (sinon, nous pouvons répondre directement *non* au problème *ZBD*$_E$). Notre algorithme transforme l'instance (G_0, G_1) en une formule booléenne ϕ de type CNF (Forme Normale Conjonctive), telle que ϕ est satisfiable si et seulement si il existe un couplage exemplaire de (G_0,G_1) n'induisant aucun point de cassure. Avant de décrire la construction de ϕ, nous présentons les

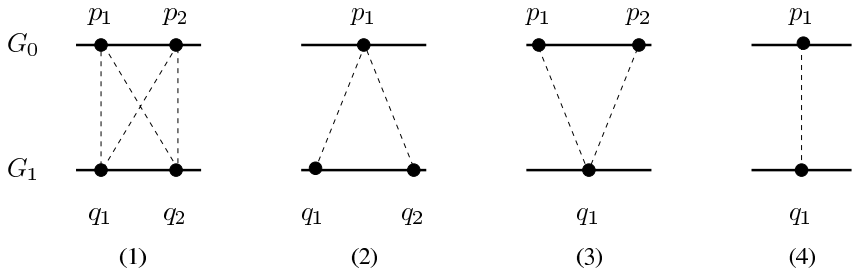


Figure 3.5 – Les quatre configurations de gènes pour les paires de génomes de type $(2,2)$: p_1 et p_2 sont les positions des occurrences de la famille f dans G_0, et q_1 et q_2 sont les positions des occurrences de la famille f dans G_1.

quatre configurations possibles des familles de gènes et précisons, pour chacune d'entre elles, les signes de leurs occurrences.

Rappelons, tout d'abord, que chaque famille apparaît au plus deux fois dans G_0 et dans G_1. Ainsi, pour toute famille f, nous avons l'une des quatre configurations décrites en Figure 3.5, où p_1, p_2, q_1 et q_2 sont les positions des gènes de la famille f dans G_0 et G_1. Rappelons également que nous nous focalisons ici sur les couplages exemplaires (G_0^E, G_1^E) de (G_0, G_1) tels que $G_0^E = G_1^E$. Ainsi, s'il existe une famille f pour laquelle une occurrence (notée g) sur G_x, $x \in \{0,1\}$, ne peut être couplée avec un gène de même signe, alors nous pouvons appliquer les règles suivantes sans perte de généralité :

- si g est la seule occurrence de f sur G_x, alors nous pouvons répondre *non* au problème *ZBD*$_E$. Cette règle s'applique par exemple dans le cas (3) de la Figure 3.5 lorsque $G_0[p_1] = G_0[p_2] = -G_1[q_1]$, engendrant alors une impossibilité de trouver un couplage exemplaire n'induisant aucun point de cassure.
- si il existe une occurrence g' de f sur G_x autre que g, alors nous pouvons supprimer g de G_x sans modifier la nature du problème. En effet, le gène g ne pourra jamais figurer dans un couplage exemplaire n'induisant aucun point de cassure. Cette règle s'applique par exemple dans le cas (3) lorsque $G_0[p_1] = -G_0[p_2] = G_1[q_1]$ et nous retirons alors le gène à la position p_2 et obtenons la configuration 4) pour cette famille de gènes.

Après application de ces règles, nous obtenons les conditions suivantes :

- Cas (1). $G_0[p_1] = G_0[p_2] = G_1[q_1] = G_1[q_2]$ ou ($G_0[p_1] = -G_0[p_2]$ et $G_1[q_1] = -G_1[q_2]$)
- Cas (2). $G_0[p_1] = G_1[q_1] = G_1[q_2]$
- Cas (3). $G_0[p_1] = G_0[p_2] = G_1[q_1]$
- Cas (4). $G_0[p_1] = G_1[q_1]$

Nous décrivons maintenant la construction de la formule booléenne ϕ de type CNF. Tout d'abord, l'ensemble des variables booléennes X est défini de la façon suivante : pour toute paire (p,q), $1 \leqslant p \leqslant \eta_{G_0}$, $1 \leqslant q \leqslant \eta_{G_1}$, telle que $|G_0[p]| = |G_1[q]|$, nous ajoutons à X la variable booléenne x_q^p. Ces variables correspondent à l'ensemble des paires de gènes pouvant être couplés. Nous définissons maintenant les clauses de ϕ permettant d'assurer un couplage exemplaire. Pour cela, nous devons, pour chaque famille de gènes $f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}$, contrôler le couplage par deux types de clauses correspondant aux deux conditions suivantes : (i) il existe au moins deux gènes de f couplés, (ii) deux paires distinctes de gènes de la famille f ne peuvent correspondre toutes les deux à des paires de gènes couplés. La première

condition s'écrira, suivant le nombre d'occurrences de même signe de la famille f, par une clause de type $(a_1 \vee a_2 \vee a_3 \vee a_4)$ ou de type $(a_1 \vee a_2)$, avec $a_i \in X$ pour $1 \leqslant i \leqslant 4$. La deuxième condition s'écrira quant à elle par un ensemble de clauses de type $(\overline{a_1} \vee \overline{a_2})$, avec $a_i \in X$ pour $1 \leqslant i \leqslant 2$.

Nous donnons ici les détails de la construction de ϕ pour chacun des quatre cas énoncés précédemment. Soit une famille $f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}$, et soient les positions des gènes de la famille f dans G_0 et dans G_1 telles que notées dans la Figure 3.5, nous notons :

- si $occ(f, G_0) = occ(f, G_1) = 2$ (cas (1)),
  - si $G_0[p_1] = G_0[p_2] = G_1[q_1] = G_1[q_2]$, 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 $G_0[p_1] = -G_0[p_2]$ et $G_1[q_1] = -G_1[q_2]$,
    - si $G_0[p_1] = G_1[q_1]$ et $G_0[p_2] = G_1[q_2]$, 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 $G_0[p_1] = G_1[q_2]$ et $G_0[p_2] = G_1[q_1]$, 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 $occ(f, G_0) = 1$ et $occ(f, G_1) = 2$ (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 $occ(f, G_0) = 2$ et $occ(f, G_1) = 1$ (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 $occ(g,G_0) = occ(g,G_1) = 1$ (cas (4)), nous ajoutons à ϕ la clause $(x_{q_1}^{p_1})$.

Il nous reste à ajouter les clauses afin d'interdire les couplages exemplaires induisant un point de cassure. Dans ce but, nous ajoutons à ϕ les clauses suivantes : pour chaque paire de variables $(x_{j_1}^{i_1}, x_{j_2}^{i_2})$ telle que $i_1 < i_2$ 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 terminée.

Par cette construction, si la formule ϕ évalue à vrai une assignation A (i.e. une affectation à vrai ou faux des variables x_q^p) et que $A(x_q^p)$ est vrai pour toute famille f apparaissant à la position p dans G_0 et q dans G_1, alors toutes les occurrences de f sauf celle à la position p (resp. à la position q) doivent être retirées dans G_0 (resp. dans G_1) afin d'obtenir un couplage exemplaire de (G_0,G_1) n'induisant aucun point de cassure. Inversement, si pour une couplage exemplaire $(G_0^E,G_1^E,\mathcal{M})$ de (G_0,G_1) n'induisant aucun point de cassure nous fixons à vrai les variables x_q^p tels que $(p_0,p_1) \in \mathcal{M}$ et à faux les autres, alors la formule ϕ est évalué à vrai.

Nous nous intéressons maintenant aux caractéristiques des clauses de ϕ, à savoir leur taille et leur nombre. Soit k le nombre de familles f qui apparaissent deux fois dans G_0 et dans G_1 avec le même signe, *i.e.* $G_0[p_1] = G_0[p_2] = G_1[q_1] = G_1[q_2]$. Remarquons que toutes les clauses dans ϕ ont une taille inférieure ou égale à 2 hormis les k clauses de taille 4 introduites dans les cas où une famille f apparaît deux fois dans G_0 et dans G_1 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 ainsi, nous assurons que ϕ est une formule de type 3-CNF (*i.e.* chaque clause est de taille au plus 3) avec exactement $2k$ clauses de taille 3.

Pour le cas $-(G_0^E)^r = G_1^E$, nous remplaçons G_0 par $-(G_0)^r$, puis nous construisons une autre formule ϕ' de type 3-CNF comme décrit précédemment. Les deux formules 3-CNF ont besoin, cependant, d'être traitées séparément.

Fernau propose dans [Fer05] un algorithme pour résoudre les formules booléennes de type 3-CNF dont la complexité est en $O^*(1.6182^{\ell})$, où ℓ est le nombre de clauses de taille 3 (précisons que $O^*(f(x)) = O(f(x)p(x))$ si $f(x)$ est une fonction exponentielle en x et $p(x)$ est un polynôme en x). Par conséquent, *ZBD$_E$* appliqué aux paires de génomes de type $(2, 2)$ est résoluble en $O^*(1.6182^{2k})$, où k est le nombre de familles f telle que $occ(f,G_0) = occ(f,G_1) = 2$. □

3.6.2 NP-Complétude de *ZBD$_I$*

Nous nous intéressons dans cette section au problème *ZBD$_I$*. Nous allons montrer ici que les problèmes *ZBD$_E$* et *ZBD$_I$* sont équivalents. Afin de prouver l'équivalence, nous introduisons le lemme suivant :

Lemme 9. *Soient deux génomes équilibrés G_0 et G_1 sans gènes dupliqués. Soient les génomes G'_0 et G'_1 obtenus en retirant tous les gènes d'une famille quelconque dans G_0 et dans G_1 . Alors nous avons $Bkp(G'_0, G'_1) \leq Bkp(G_0, G_1)$.*

Démonstration. Soient deux génomes équilibrés G_0 et G_1 sans gènes dupliqués. Soient une famille $f \in \mathcal{F}_{G_0}$ et les deux génomes G'_0 et G'_1 obtenus à partir de G_0 et G_1 en supprimant tous les gènes de la famille f. Nous examinons les quatre cas possibles (le symbole $_{\blacktriangle}$ indique un point de cassure).

1. $G_0 = \ldots\ x\ f\ y\ \ldots$ et $G_1 = \ldots\ x\ f\ y\ \ldots$ ou $G_1 = \ldots\ -y\ -f\ -x\ \ldots$
   Suppression des gènes de la famille f dans G_0 et G_1 induisant :
   $G'_0 = \ldots\ x\ y\ \ldots$ et $G'_1 = \ldots\ x\ y\ \ldots$ ou $G'_1 = \ldots\ -y\ -x\ \ldots$,
   et par conséquent $Bkp(G'_0, G'_1) = Bkp(G_0, G_1)$.
2. $G_0 = \ldots\ x_{\blacktriangle} f\ y\ \ldots$ et $G_1 = \ldots\ z\ f\ y\ \ldots$ ou $G_1 = \ldots\ -y\ -f\ -z\ \ldots$
   Suppression des gènes de la famille f dans G_0 et G_1 induisant :
   $G'_0 = \ldots\ x_{\blacktriangle} y\ \ldots$ et $G'_1 = \ldots\ z\ y\ \ldots$ ou $G'_1 = \ldots\ -y\ -z\ \ldots$,
   et ainsi $Bkp(G'_0, G'_1) = Bkp(G_0, G_1)$.
3. $G_0 = \ldots\ x\ f_{\blacktriangle} y\ \ldots$ et $G_1 = \ldots\ x\ f\ z\ \ldots$ ou $G_1 = \ldots\ -z\ -f\ -x\ \ldots$
   Suppression des gènes de la famille f dans G_0 et G_1 induisant :
   $G'_0 = \ldots\ x_{\blacktriangle} y\ \ldots$ et $G'_1 = \ldots\ x\ z\ \ldots$ ou $G'_1 = \ldots\ -z\ -x\ \ldots$,
   et par conséquent $Bkp(G'_0, G'_1) = Bkp(G_0, G_1)$.
4. $G_0 = \ldots\ x_{\blacktriangle} f_{\blacktriangle} y\ \ldots$
   suppression des gènes de la famille f dans G_0 et G_1 induisant :
   $G'_0 = \ldots\ x_{\blacktriangle} y\ \ldots$ ou $G'_0 = \ldots\ x\ y\ \ldots$,
   et ainsi $Bkp(G'_0, G'_1) < Bkp(G_0, G_1)$.

Dans chacun de ces quatre cas, nous avons $Bkp(G'_0, G'_1) \leq Bkp(G_0, G_1)$. □

Théorème 6. *Les problèmes ZBD$_E$ et ZBD$_I$ sont équivalents.*

Démonstration. L'implication de *ZBD$_E$* vers *ZBD$_I$* est triviale. En effet, tout couplage exemplaire est un couplage intermédiaire. Ainsi, s'il existe un couplage exemplaire n'induisant aucun point de cassure, alors il existe également un couplage intermédiaire (le même couplage) n'induisant aucun point de cassure. L'implication de *ZBD$_I$* vers *ZBD$_E$* découle du Lemme 9, celui-ci montrant que l'on peut, à partir d'un couplage intermédiaire (i.e. une correspondance un à un des gènes), construire un couplage exemplaire n'induisant pas plus de points de cassure. Ainsi, s'il existe un couplage intermédiaire n'induisant aucun point de cassure, alors il existe également un couplage exemplaire (en appliquant itérativement le Lemme 9 jusqu'à obtenir un couplage exemplaire) n'induisant aucun point de cassure. □

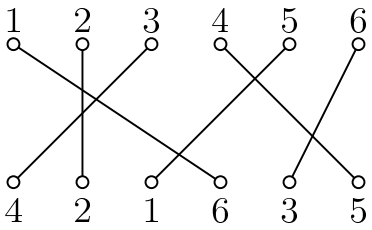


Figure 3.6 – Exemple d'instance de *diagramme de couplage*. Celle-ci est composée de deux lignes parallèles de points étiquetés et de segments joignant des paires de points distincts.

Étant donnés deux génomes G_0 et G_1, il découle des Théorèmes 5 et 6 que le problème BD_I n'est pas approximable même si $occ(G_0) = 2$. Par les récents travaux de Blin et al. [BFSV09], nous pouvons conclure que BD_I n'est pas approximable même pour les paires de génomes de type $(2,2)$.

	couplage exemplaire (E)	*couplage maximal* (M)	*couplage intermédiaire* (I)
$ICOM_X$ $ICONS_X$	**NP**-Complet [CFRV06] (instance $(1,2)$) **APX**-Difficile [AFR08a] (instance $(1,2)$)		
BD_X	**NP**-Complet [Bry00] (instance $(1,2)$) **NP**-Complet [BCF04]*		
	APX-Difficile [AFR08a] (instance $(1,2)$)		
ZBD_X	**NP**-Complet [CFZ06] (instance $(3,3)$) **NP**-Complet [AFR+09] (instance $(2,k)$) **NP**-Complet [BFSV09] (instance $(2,2)$)	?	$\equiv ZBD_E$ [AFR+09]

Table 3.4 – Résultats de complexité algorithmique : ajout de l'équivalence des problèmes ZBD_E et ZBD_I.
* même si une seule famille est dupliquée.

3.6.3 Polynomialité de ZBD_M

Nous montrons dans cette section que, à l'inverse des modèles *couplage exemplaire* et *couplage intermédiaire*, décider s'il existe un couplage maximal de deux génomes G_0 et G_1 n'induisant aucun point de cassure est résoluble en temps polynomial, et qu'ainsi nous ne pouvons pas exclure l'existence d'algorithmes d'approximation pour le modèle *couplage maximal* (problème ZBD_M).

L'idée principale de notre approche est de transformer toute instance de ZBD_M en une instance de *diagramme de couplage* et d'utiliser ensuite un algorithme efficace pour trouver dans ce diagramme un plus grand ensemble de segments ne se croisant pas. Notons que ce dernier problème est équivalent à trouver la plus grande sous-séquence croissante dans des permutations. Un *diagramme de couplage* [Gol80] est composé de lignes parallèles de n points étiquetés et de n segments joignant des paires de points distincts (cf. Figure 3.6).

Le graphe d'intersection des segments est appelé un *graphe de permutation*. En effet, si les points de la ligne supérieure sont étiquetés par $1, 2, \ldots, n$, alors les points sur la seconde ligne sont étiquetés par

une permutation de $1, 2, \ldots, n$.

Selon l'Observation 1, il est suffisant de considérer les deux cas suivants : $G_0^M = G_1^M$ ou $-(G_0^M)^r = G_1^M$, où (G_0^M, G_1^M,$\mathcal{M}$) est un couplage maximal de (G_0, G_1). Nous nous focalisons ici sur le cas $G_0^M = G_1^M$ (le cas $-(G_0^M)^r = G_1^M$ est identique à une inversion près et sera traité à la fin). Pour la clarté de la présentation, nous introduisons les notations suivantes. Pour toute famille de gènes f, nous notons $occ_{\text{pos}}(G, f)$ (resp. $occ_{\text{neg}}(G, f)$) le nombre d'occurrences positives (resp. négatives) de f dans le génome G.

Afin de transformer une instance de *ZBD$_M$* en une instance de *diagramme de couplage*, il est nécessaire de traiter les génomes initiaux, et ceci sans altérer la réponse au problème *ZBD$_M$*. Soit la transformation polynomiale qui associe à (G_0,G_1) les deux génomes G'_0 et G'_1 en supprimant les gènes présents uniquement sur un seul génome, en prenant en compte les signes (cf. Figure 3.7 étape 1). Puisque les gènes présents uniquement sur un seul génome n'interviennent pas dans les couplages maximaux qui nous intéressent (i.e. ceux qui associent uniquement des gènes de même signe), leur suppression ne retire pas de couplage maximaux intéressants.

Nous décrivons maintenant comment construire un *diagramme de couplage* $D(G'_0, G'_1)$ à partir de la paire de génomes (G'_0,G'_1) (cf. Figure 3.7). Nous définissons la construction des points étiquetés de la ligne supérieure de la façon suivante. En lisant le génome G'_0 de gauche à droite, nous remplaçons chaque gène g par la séquence de points étiquetés

$$+\mathsf{g}_0^{|g|}(i, occ_{\text{pos}}(G'_1, |g|)) \quad + \mathsf{g}_0^{|g|}(i, occ_{\text{pos}}(G'_1, |g|) - 1) \quad \ldots \quad + \mathsf{g}_0^{|g|}(i, 1)$$

si le gène g est la i-ième occurrence positive de la famille $|g|$ dans le génome G'_0 ou par la séquence de points étiquetés

$$-\mathsf{g}_0^{|g|}(i, occ_{\text{neg}}(G'_1, |g|)) \quad - \mathsf{g}_0^{|g|}(i, occ_{\text{neg}}(G'_1, |g|) - 1) \quad \ldots \quad - \mathsf{g}_0^{|g|}(i, 1)$$

si g est la i-ième occurrence négative de la famille $|g|$ dans le génome G'_0. Une construction symétrique est réalisée pour la création de la ligne inférieure, i.e. en lisant le génome G'_1 de gauche à droite, nous remplaçons chaque gène g par la séquence de points étiquetés

$$+\mathsf{g}_1^{|g|}(i, occ_{\text{pos}}(G'_0, |g|)) \quad + \mathsf{g}_1^{|g|}(i, occ_{\text{pos}}(G'_0, |g|) - 1) \quad \ldots \quad + \mathsf{g}_1^{|g|}(i, 1)$$

si g est la i-ième occurrence positive de la famille $|g|$ dans le génome G'_1 ou par la séquence de points étiquetés

$$-\mathsf{g}_1^{|g|}(i, occ_{\text{neg}}(G'_0, |g|)) \quad - \mathsf{g}_1^{|g|}(i, occ_{\text{neg}}(G'_0, |g|) - 1) \quad \ldots \quad - \mathsf{g}_1^{|g|}(i, 1)$$

si g est la i-ième occurrence négative de la famille f dans le génome G'_1. Nous obtenons maintenant le *diagramme de couplage* $D(G'_0, G'_1)$ de la façon suivante : chaque point étiqueté $+\mathsf{g}_1^{|g|}(i, j)$ (resp. $-\mathsf{g}_1^{|g|}(i, j)$) de la ligne supérieure est connecté au point étiqueté $+\mathsf{g}_2^{|g|}(j, i)$ (resp. $-\mathsf{g}_2^{|g|}(j, i)$) de la ligne inférieure par un segment. Clairement, chaque point étiqueté est incident à exactement un segment, et par conséquent $D(G'_0, G'_1)$ est un *diagramme de couplage*. Remarquons que le nombre de points sur chaque ligne (et par conséquent le nombre de segments) est borné supérieurement par $\eta_{G_0}.\eta_{G_1}$ (cas extrême où $\mathcal{F}_{G_0} = \mathcal{F}_{G_1} = \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1} = 1$).

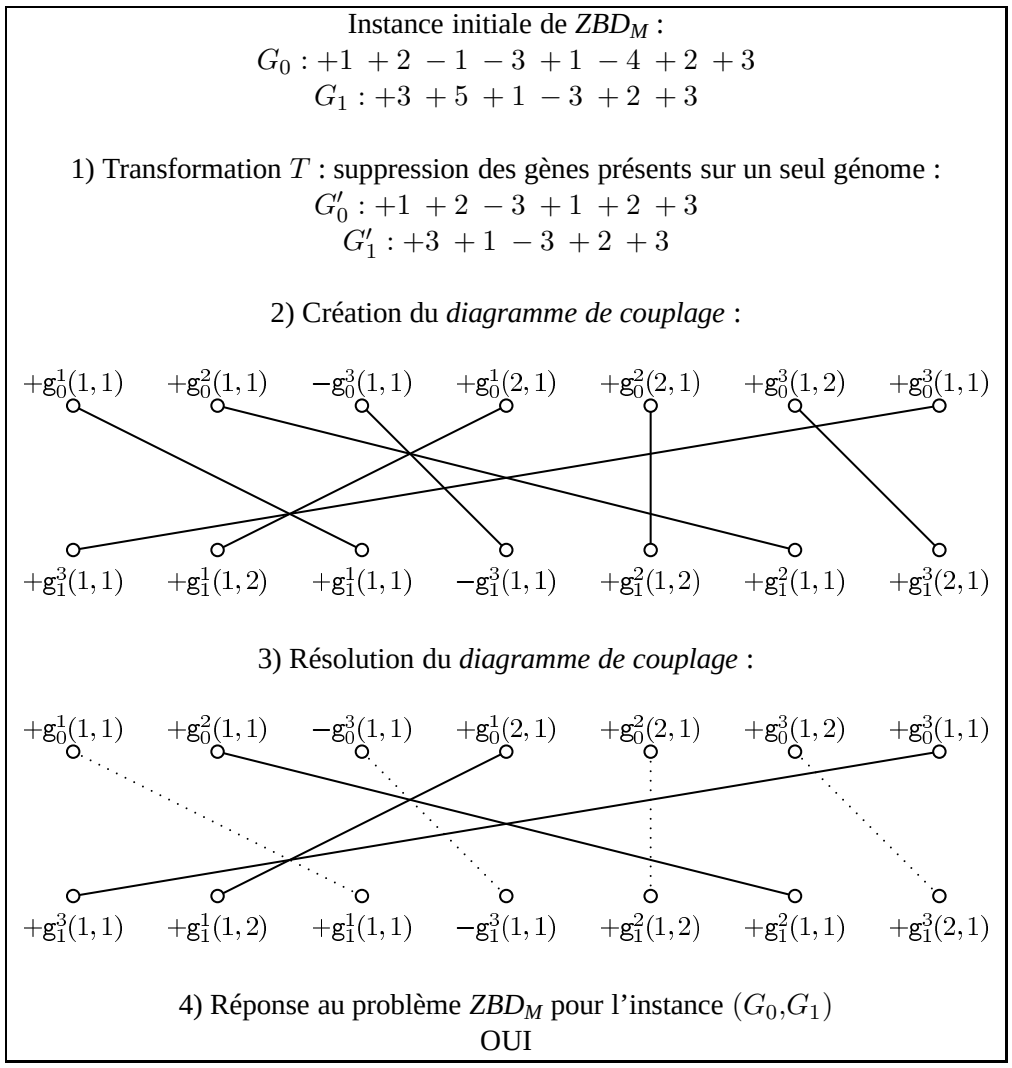


Figure 3.7 – Exemple de construction et de résolution de *ZBD$_M$*. Les segments en pointillé correspondent aux segments d'une solution pour ce *diagramme de couplage* (i.e. aucune paire de segments croisés). Nous obtenons dans cet exemple quatre segments. La taille d'un couplage maximal de (G_0,G_1) étant également égale à 4, nous pouvons affirmer qu'il existe un couplage maximal de (G_0,G_1) n'induisant aucun point de cassure.

Observons que par construction du *diagramme de couplage*, pour tout $x \in \{0,1\}$ et pour toute paire de points étiquetés $(+\mathsf{g}_x^f(i,j), +\mathsf{g}_x^f(i,k))$, $j \neq k$ et $f \in \mathcal{F}_{G'_0} \cup \mathcal{F}_{G'_1}$, les deux segments incidents à l'un de ces deux points s'intersectent ; la même observation peut être faite pour toute paire de points étiquetés $(-\mathsf{g}_x^f(i,j), -\mathsf{g}_x^f(i,k))$, $j \neq k$ et $f \in \mathcal{F}_{G'_0} \cup \mathcal{F}_{G'_1}$. Il en résulte directement le lemme suivant.

Lemme 10. *Soit une famille de gènes $f \in \mathcal{F}_{G'_0} \cup \mathcal{F}_{G'_1}$.*
Si $[+g_0^f(i,j), +g_1^f(j,i)]$ et $[+g_0^f(k,\ell), +g_1^f(\ell,k)]$ (resp. $[-g_0^f(i,j), -g_1^f(j,i)]$ et $[-g_0^f(k,\ell), -g_1^f(\ell,k)]$) sont deux segments qui ne se croisent pas dans le diagramme de couplage $D(G'_0, G'_1)$*, alors nous avons $i \neq k$ et $j \neq \ell$.*

Théorème 7. *Le problème ZBD_M est résoluble en temps polynomial.*

Démonstration. Soient deux génomes G_0 et G_1. Soient les génomes G'_0 et G'_1 obtenus par la transformation T. Soit m la taille d'un couplage maximal de (G_0,G_1). Selon le Lemme 10, il existe un couplage maximal $(G_0^M, G_1^M, \mathcal{M})$ de (G'_0, G'_1) tel que $G_0^M = G_1^M$ si il existe m segments ne se croisant pas dans $D(G'_0, G'_1)$. Le nombre maximum de segments ne se croisant pas dans un *diagramme de couplage* avec n points sur chaque ligne peut être calculé en $O(n \log\log n)$ [CW92].

Dans le cas $-(G_0^M)^r = G_1^M$, nous remplaçons G_0 par $-(G_0)^r$ et appliquons la transformation T, puis le même algorithme sur le nouveau *diagramme de couplage* obtenu. □

	couplage exemplaire (E)	*couplage maximal* (M)	*couplage intermédiaire* (I)
$ICOM_X$ $ICONS_X$	**NP**-Complet [CFRV06] (instance $(1,2)$) **APX**-Difficile [AFR08a] (instance $(1,2)$)		
BD_X	**NP**-Complet [Bry00] (instance $(1,2)$) **NP**-Complet [BCF04]*		
	APX-Difficile [AFR08a] (instance $(1,2)$)		
ZBD_X	**NP**-Complet [CFZ06] (instance $(3,3)$) **NP**-Complet [AFR+09] (instance $(2,k)$) **NP**-Complet [BFSV09] (instance $(2,2)$)	polynomial [AFR+09]	$\equiv ZBD_E$ [AFR+09]

Table 3.5 – Résultats de complexité algorithmique : ajout de la polynomialité de ZBD_M. * même si une seule famille est dupliquée.

3.7 Méthodes approchées : algorithmes d'approximation

Plusieurs algorithmes d'approximation ont été proposés en ce qui concerne la minimisation du nombre de *points de cassure* entre deux génomes équilibrés pour le modèle *couplage maximal* [GKZ04, KW06] (problème BD_M). Nous proposons dans cette section trois algorithmes d'approximation concernant la maximisation du nombre d'adjacences (à l'opposé de minimisation du nombre de points de cassure), également pour le modèle *couplage maximal* :

- De ratio 1.1442 lorsque $occ(G_0) = 2$
- De ratio $(3+\epsilon)$ lorsque $occ(G_0) = 3$, pour tout $\epsilon > 0$
- De ratio 4 dans le cas général

Ces travaux ont été publiés dans [AFR+09]. Remarquons que dans le cas général, à l'opposé des ratios proposés dans [GKZ04, KW06], notre ratio d'approximation est indépendant du nombre d'occurrences des gènes dans les génomes. Notons aussi que dans [CFX+07], Chen et al. ont montré que BD_E n'admet pas d'algorithme d'approximation de ratio $n^{1-\epsilon}$ pour deux génomes non-équilibrés G_0 et G_1.

3.7.1 1.1442-approximation pour *Eq*-2-ADJ_M

Nous nous focalisons ici sur deux génomes équilibrés G_0 et G_1 tels que $occ(G_0) = occ(G_1) = 2$, et donnons un algorithme d'approximation pour *Eq*-2-ADJ_M basé sur une transformation en une instance du problème *Max*-2-*CSP*, pour lequel un algorithme d'approximation de ratio 1.1442 est donné dans

[CMM07]. L'idée principale est de construire une formule booléenne φ pour chaque adjacence possible, et ensuite de maximiser le nombre de formules booléennes ϕ qui peuvent être simultanément satisfaites par une assignation (i.e. une fixation de chaque variable à vrai ou faux). Le nombre de formules simultanément satisfaites sera exactement le nombre d'adjacences, et par conséquent tout ratio d'approximation pour *Max*-2-*CSP* sera un ratio d'approximation pour *Eq*-2-*ADJ*$_M$. Nous définissons maintenant le problème *Max*-k-*CSP*.

Problème : *Max*-k-*CSP*
Entrée : Une paire (χ,Φ), où χ est un ensemble de variables booléennes et Φ un ensemble de formules booléennes sur χ telles que chaque formule contient au plus k littéraux de χ. Un entier p.
Problème de décision : Existe-t-il une assignation de χ tel que au moins p formules de Φ sont satisfaites ?
Problème d'optimisation : Trouver une assignation de χ maximisant le nombre de formules de Φ satisfaites.

Nous définissons la transformation polynomiale ConstruireCSP qui associe à toute instance de *Eq*-2-*ADJ*$_M$ une instance de *Max*-2-*CSP*. Étant donnée une instance (G_0,G_1) de *Eq*-2-*ADJ*$_M$ (i.e. G_0 et G_1 sont équilibrés et $occ(G_0) = occ(G_1) = 2$), nous créons une variable X_f pour toute famille $f \in \mathcal{F}_{G_0}$ et définissons χ comme l'ensemble des variables X_f.

Afin de définir les formules, nous introduisons la notation suivante. Pour tout génome G, nous notons $N_G[p]$, $1 \leq p \leq \eta_G$, le nombre d'occurrences de la famille $|G[p]|$ dans les $(p-1)$ premières positions de G. Nous construisons maintenant l'ensemble Φ de formules. Pour tout duo (cf. Définition 3, page 17) $d_i = (G_0[i],G_0[i+1])$, $1 \leq i \leq \eta_{G_0} - 1$, tel que d_i ou $-d_i$ apparaît dans G_1, nous distinguons quatre cas pour créer une formule φ_i de Φ :

1. Il existe un unique duo $d_j = (G_1[j],G_1[j+1])$ dans G_1 tel que $d_j = d_i$ ou $d_j = -d_i$. Pour une meilleure lisibilité, nous définissons le littéral Y_p^q, $1 \leq p \leq \eta_{G_0}$, $1 \leq q \leq \eta_{G_1}$, avec $|G_0[p]| = |G_1[q]|$, de la façon suivante : $Y_p^q = X_{|G_0[p]|}$ si $N_{G_0}[p] = N_{G_1}[q]$ et $Y_p^q = \overline{X_{|G_0[p]|}}$ sinon. Nous considérons maintenant les deux cas suivants :
   (a) $d_i = d_j$: dans ce cas, $\varphi_i = (Y_i^j \wedge Y_{i+1}^{j+1})$.
   (b) $d_i = -d_j$: dans ce cas, $\varphi_i = (Y_i^{j+1} \wedge Y_{i+1}^j)$.
2. Le duo d_i apparaît deux fois dans G_1 (comme étant d_i ou $-d_i$). Nous considérons les deux cas suivants :
   (c) $N_{G_0}[i] = N_{G_0}[i+1]$: dans ce cas, $\varphi_i = (\overline{X_{|G_0[i]|} \oplus X_{|G_0[i+1]|}})$ où $\oplus$ est la fonction booléenne XOR.
   (d) $N_{G_0}[i] \neq N_{G_0}[i+1]$: dans ce cas, $\varphi_i = (X_{|G_0[i]|} \oplus X_{|G_0[i+1]|})$.

Remarquons que chaque formule φ_i contient deux littéraux. Par conséquent, (χ,Φ) est une instance de *Max*-2-*CSP*. Nous nous donnons maintenant, et jusqu'à la fin de cette section, deux génomes équilibrés G_0 et G_1 avec $occ(G_0) = occ(G_1) = 2$. Nous posons également l'instance (χ,Φ) de *Max*-2-*CSP* obtenue par ConstruireCSP(G_0,G_1).

Lemme 11. *Pour tout entier k, s'il existe un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) induisant au moins k adjacences, alors il existe une assignation des variables de χ tel que au moins k formules de Φ sont satisfaites.*

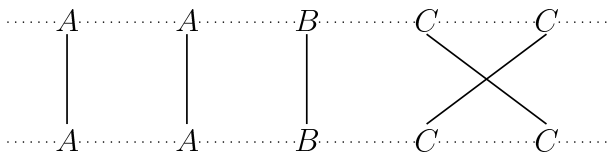


Figure 3.8 – Illustration de toutes les possibilités d'assignation : $X_A = 1$ (la famille A apparaît deux fois et les gènes de la famille A sont couplés dans l'ordre de lecture), $X_B = 1$ (la famille B apparaît une seule fois) et $X_C = 0$ (la famille C apparaît deux fois et les gènes de la famille C ne sont pas couplés dans l'ordre de lecture). Notons que cette construction est indépendante des signes des gènes.

Démonstration. Soit un entier positif k. Supposons qu'il existe un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) induisant au moins k adjacences. Nous construisons l'assignation des variables de χ de la manière suivante. Pour toute famille $f \in \mathcal{F}_{G_0}$, nous fixons la variable booléenne X_f à 1 (i.e. à vrai) si un seul gène de la famille f apparaît dans G_0, sinon nous fixons $X_f = 1$ si et seulement si les deux occurrences de la famille f sont couplées dans l'ordre de lecture (cf. Figure 3.8). Nous montrons maintenant que pour chaque duo induisant une adjacence entre G_0^M et G_1^M, il existe une formule de Φ distincte et satisfaite. Soit un duo $d_i = (G_0^M[i],G_0^M[i+1])$, $1 \leq i \leq \eta_{G_0} - 1$, induisant une adjacence, et soit $d_j = (G_1^M[j],G_1^M[j+1])$ le duo correspondant sur G_1^M. Par construction de Φ, il existe une formule $\varphi_i \in \Phi$ qui a été précédemment définie par l'un des cas (a), (b), (c) ou (d) de la définition de ConstruireCSP. Nous affirmons que, pour chacun de ces cas, φ_i est satisfaite :

(a) $\varphi_i = (Y_i^j \wedge Y_{i+1}^{j+1})$ et $d_i = d_j$. Nous prouvons tout d'abord que le littéral Y_i^j est vrai. Trois cas sont possibles :
   (i) $occ(|G_0[i]|,G_0) = 1$; alors nous avons fixé dans notre assignation $X_{|G_0[i]|} = 1$. De plus, nous avons $Y_i^j = X_{|G_0[i]|}$ (puisque $N_{G_0}[i] = N_{G_1}[j] = 0$), et par conséquent Y_i^j est vrai.
   (ii) $occ(|G_0[i]|,G_0) = 2$ et $N_{G_0}[i] = N_{G_1}[j]$; alors, par définition de notre assignation et puisque $(i,j) \in \mathcal{M}$ (i.e. les gènes $G_0[i]$ et $G_1[j]$ sont mis en correspondance par le couplage maximal $(G_0^M,G_1^M,\mathcal{M})$), nous avons $X_{|G_0[i]|} = 1$ (les gènes sont couplés ici dans l'ordre de lecture). De plus, nous avons $Y_i^j = X_{|G_0[i]|}$ ce qui implique que Y_i^j est vrai.
   (iii) $occ(|G_0[i]|,G_0) = 2$ et $N_{G_0}[i] \neq N_{G_1}[j]$; alors, par définition de notre assignation et puisque $G_0[i]$ et $G_1[j]$ sont couplés dans le couplage maximal $(G_0^M,G_1^M,\mathcal{M})$, nous avons $X_{|G_0[i]|} = 0$ (les gènes ne sont pas couplés ici dans l'ordre de lecture). De plus, nous avons dans ce cas $Y_i^j = \overline{X_{|G_0[i]|}}$ ce qui implique que Y_i^j est vrai.

   Dans chacun de ces cas, nous avons prouvé que Y_i^j est vrai. Nous pouvons également prouver que Y_{i+1}^{j+1} est vrai, en utilisant les mêmes arguments. Par conséquent, nous concluons que φ_i est vraie.

(b) $\varphi_i = Y_i^{j+1} \wedge Y_{i+1}^j$ et $d_i = -d_j$. Par des arguments similaires au cas (a), nous pouvons prouver que Y_i^{j+1} et Y_{i+1}^j sont vrais.

(c) Nous avons $N_{G_0}[i] = N_{G_0}[i+1]$ et le duo d_i apparaît deux fois dans G_1 (noté par d_j et $d_{j'}$). Puisque d_i induit une adjacence, le duo d_i est couplé soit avec d_j soit avec $d_{j'}$. Dans ces deux cas, nous avons $X_{|G_0[i]|} = X_{|G_0[i+1]|}$ (sinon $G_0[i]$ et $G_0[i+1]$ ne seraient pas couplés avec des gènes consécutifs). De plus, $\varphi_i = (\overline{X_{|G_0[i]|} \oplus X_{|G_0[i+1]|}})$ et ainsi, φ_i est vraie.

(d) Nous avons $N_{G_0}[i] \neq N_{G_0}[i+1]$ et le duo d_i apparaît deux fois dans G_1 (noté par d_j et $d_{j'}$).

Puisque d_i induit une adjacence, le duo d_i est couplé avec soit d_j soit $d_{j'}$. Dans ces deux cas, nous avons $X_{|G_0[i]|} \neq X_{|G_0[i+1]|}$ (sinon $G_0[i]$ et $G_0[i+1]$ ne seraient pas couplés avec des gènes consécutifs). De plus, $\varphi_i = (X_{|G_0[i]|} \oplus X_{|G_0[i+1]|})$ et ainsi, φ_i est vraie.

Nous avons construit une assignation des variables de χ tel que, pour chaque duo d_i de G_0^M impliquant une adjacence, il existe une formule distincte et satisfaite $\varphi_i \in \Phi$. Ainsi, s'il existe un couplage maximal de (G_0,G_1) induisant au moins k adjacences, alors l'assignation correspondant implique au moins k formules satisfaites. □

Lemme 12. *Soit un entier positif k. S'il existe une assignation de χ tel que au moins k formules de Φ sont satisfaites, alors il existe un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) induisant au moins k adjacences.*

Démonstration. Soit un entier positif k. Supposons qu'il existe une assignation A de χ tel que au moins k formules $\varphi_i \in \Phi$ soient satisfaites. Nous créons le couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) par la construction CSP_vers_Adj prenant en entrée le triplet (A,G_0,G_1) et définie de la manière suivante.
Construction CSP_vers_Adj **en deux étapes :**

1. Pour toute variable X_f, $f \in \mathcal{F}_{G_0}$, telle que $occ(f,G_0) = 2$, nous couplons les occurrences des gènes de la famille f dans l'ordre de lecture si $X_f = 1$ (telle la famille A dans la Figure 3.8). Si nous avons $X_f = 0$, nous couplons la première occurrence de f sur G_0 avec la seconde occurrence sur G_1 et la seconde occurrence de f sur G_0 avec la première sur G_1 (telle la famille C de la Figure 3.8).
2. Nous couplons les gènes qui ne sont pas dupliqués (telle la famille B de la Figure 3.8).

Nous prouvons maintenant que chaque formule satisfaite $\varphi_i \in \Phi$ induit une adjacence distincte de $(G_0^M,G_1^M,\mathcal{M})$. Soit $\varphi_i \in \Phi$ une formule satisfaite qui est définie dans l'un des cas (a), (b), (c) ou (d) de la définition de ConstruireCSP :

(a) Nous avons $\varphi_i = (Y_i^j \wedge Y_{i+1}^{j+1})$ et les duos $d_i = (G_0[i],G_0[i+1])$ et $d_j = (G_1[j],G_1[j+1])$ sont identiques.
Nous devons prouver ici que les duos d_i et d_j sont mis en correspondance par le couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ et ainsi que d_i induit une adjacence. Tout d'abord, nous montrons que les gènes (i,j) appartiennent à $\mathcal{M}$. Puisque φ_i est satisfaite, nous avons $Y_i^j = 1$. Nous devons étudier séparément les trois cas possibles :
(i) $occ(|G_0[i]|,G_0) = 1$. Dans ce cas, le gène $G_0[i]$ peut être couplé uniquement avec $G_1[j]$.
(ii) $occ(|G_0[i]|,G_0) = 2$ et $N_{G_0}[i] = N_{G_1}[j]$. Dans ce cas, nous avons défini $Y_i^j = X_{|G_0[i]|}$ impliquant $X_{|G_0[i]|} = 1$. Ainsi, puisque $N_{G_0}[i] = N_{G_1}[j]$, nous avons $(i,j) \in \mathcal{M}$.
(iii) $occ(|G_0[i]|,G_0) = 2$ et $N_{G_0}[i] \neq N_{G_1}[j]$. Dans ce cas, nous avons défini $Y_i^j = \overline{X_{|G_0[i]|}}$ impliquant $X_{|G_0[i]|} = 0$. Ainsi, puisque $N_{G_0}[i] \neq N_{G_1}[j]$, nous avons $(i,j) \in \mathcal{M}$.
Dans chacun de ces cas, la paire de positions (i,j) appartient à $\mathcal{M}$. Nous pouvons conclure de la même manière que les gènes $G_0[i+1]$ et $G_1[j+1]$ sont aussi mis en correspondance par le couplage maximal $(G_0^M,G_1^M,\mathcal{M})$, impliquant par suite que d_i induit une adjacence.

(b) Nous avons $\varphi_i = (Y_i^{j+1} \wedge Y_{i+1}^j) = 1$ et les duos $d_i = (G_0[i],G_0[i+1])$ et $d_j = (G_1[j],G_1[j+1])$ sont inversés.
Nous pouvons utiliser le même raisonnement que celui utilisé dans le cas (a) pour prouver que d_i induit une adjacence.

(c) Le duo d_i apparaît deux fois dans G_1 (noté par d_j et $d_{j'}$). Nous avons $\varphi_i = (\overline{X_{|G_0[i]|} \oplus X_{|G_0[i+1]|}})$ et $N_{G_0}[i] = N_{G_0}[i+1]$.
Puisque φ_i est vraie, nous avons $X_{|G_0[i]|} = X_{|G_0[i+1]|}$ impliquant par construction du couplage maximal que le duo d_i est couplé avec d_j ou avec $d_{j'}$.

(d) Le duo d_i apparaît deux fois dans G_1 (noté par d_j et $d_{j'}$). Nous avons $\varphi_i = (X_{|G_0[i]|} \oplus X_{|G_0[i+1]|})$ et $N_{G_0}[i] \neq N_{G_0}[i+1]$. Puisque φ_i est vraie, nous avons $X_{|G_0[i]|} \neq X_{|G_0[i+1]|}$ impliquant par construction du couplage maximal que d_i est couplé avec d_j ou avec $d_{j'}$.

En conséquence, pour chaque formule satisfaite, il existe une adjacence distincte de (G_0^M,G_1^M). Ainsi, s'il existe une assignation de χ induisant au moins k formules satisfaites de Φ, alors il existe un couplage maximal de (G_0,G_1) induisant au moins k adjacences. □

Une conséquence des Lemmes 11 et 12 est que toute α-approximation pour *Max*-2-*CSP* implique une α-approximation pour *Eq*-2-*ADJ*$_M$. Dans [CMM07], un algorithme d'approximation est donné pour *Max*-2-*CSP*, pour lequel le ratio d'approximation est égal à $\frac{1}{0.874} \sim 1.1442$. Ainsi, nous proposons l'Algorithme 3.1 pour la résolution de *Eq*-2-*ADJ*$_M$ et obtenons le théorème suivant.

Théorème 8. *Le problème Eq-2-ADJ$_M$ est* 1.1442*-approximable.*

ALG. 3.1 – Algorithme Approx-Eq-2-Adj$_M$

{**Entrée :** Deux génomes équilibrés G_0 et G_1 avec $occ(G_0) = occ(G_1) = 2$}
{**Sortie :** Un *couplage maximal* $(G_0^M, G_1^M, \mathcal{M})$ de (G_0,G_1)}

1 $(\chi,\Phi) \leftarrow$ ConstruireCSP(G_0,G_1).
2 Invocation de l'algorithme d'approximation de ratio $\frac{1}{0.874}$ (Charikar et al. [CMM07]) afin d'obtenir une assignation A de χ.
3 Construction du *couplage maximal* $(G_0^M,G_1^M,\mathcal{M}) \leftarrow$ CSP_vers_Adj(A,G_0,G_1).

3.7.2 $(3+\epsilon)$-approximation pour *Eq*-3-*ADJ*$_M$

Nous proposons maintenant une $(3+\epsilon)$-approximation pour le problème *Eq*-3-*ADJ*$_M$ en s'appuyant sur le problème du calcul d'un *ensemble indépendant maximum* d'un graphe (problème noté *MIS* pour *Maximum Independent Set*). Nous précisons tout d'abord les notions d'ensemble indépendant et d'ensemble indépendant maximum de la manière suivante :

Définition 30. ***(Ensemble indépendant)***
Soit un graphe $G = (V,E)$. Un sous-ensemble V' de V est appelé un ensemble indépendant *de G s'il n'existe pas deux sommets de V' adjacents dans G.*

Définition 31. ***(Ensemble indépendant maximum)***
Soit un graphe $G = (V,E)$. Un ensemble indépendant V' de G est dit maximum *s'il n'existe pas d'ensemble indépendant V'' de G tel que $|V'| < |V''|$.*

Le problème *MIS* est défini comme suit.

Problème : *MIS* (*Maximum Independent Set*)
Entrée : Un graphe $G = (V,E)$. Un entier k.
Problème de décision : Existe-t-il un ensemble indépendant $|V'|$ de G tel que $|V'| \geqslant k$?
Problème d'optimisation : Trouver un ensemble indépendant maximum de G.

Dans [GKZ04], Goldstein *et al*. utilisent le problème *MIS* afin de produire un algorithme d'approximation pour le problème $Minimum\ Common\ String\ Partition$ en créant un *graphe des conflits*. Nous construisons de la même manière une instance de *MIS* où chaque sommet représente une adjacence possible et où une arête représente un conflit entre deux adjacences. Nous définissons la transformation polynomiale ConstruireMIS qui associe à deux génomes équilibrés G_0 et G_1 une instance de *MIS* de la manière suivante. Nous construisons un sommet pour chaque couple de duos de (G_0,G_1), puis nous créons une arête entre deux sommets lorsqu'ils sont en conflit, i.e. lorsque les deux couples de duos correspondants sont incompatibles. La Figure 3.9 illustre le graphe obtenu par la construction ConstruireMIS(G_0,G_1) en considérant les génomes $G_0 = +3+1+2+3+4+2+5$ et $G_1 = +3+4+2+3+1+2+5$.

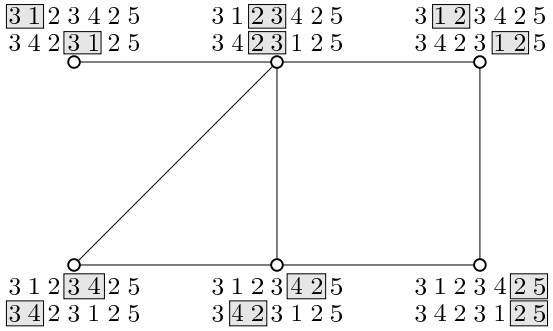


Figure 3.9 – Le graphe des conflits obtenu par ConstruireMIS(G_0,G_1) avec $G_0 = +3+1+2+3+4+2+5$ et $G_1 = +3+4+2+3+1+2+5$. Les signes des gènes, tous positifs, ne sont pas reproduits pour une meilleure lisibilité. Une arête est créée entre chaque paire de sommets incompatibles.

Afin de prouver qu'il existe une $(3+\epsilon)$-approximation pour le problème *Eq-3-ADJ$_M$*, nous introduisons deux lemmes intermédiaires. Nous nous donnons pour toute cette section deux génomes équilibrés G_0 et G_1 et construisons le graphe G par ConstruireMIS(G_0,G_1).

Lemme 13. *Pour tout entier positif k, il existe un ensemble indépendant V' de G tel que $|V'| \geq k$ si et seulement si il existe un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) induisant au moins k adjacences.*

Démonstration. Soit un entier positif k.

($\Rightarrow$) Supposons qu'il existe un ensemble indépendant V' de G tel que $|V'| \geq k$. Nous construisons un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) par la transformation MIS_vers_Adj, laquelle associe au triplet (V',G_0,G_1) un couplage maximal.
Construction MIS_vers_Adj **en deux étapes :**

1. Tout d'abord, pour chaque sommet de V', les deux duos assimilés sont mis en correspondance par le couplage, induisant ainsi une adjacence que nous appellerons une *adjacence définitive*. Par construction de G, cette opération est possible. En effet, deux sommets non adjacents de G induisent deux adjacences compatibles.
2. Puis, nous couplons arbitrairement les gènes non couplés restants. Cette opération ne retire aucune adjacence définitive.

Nous obtenons au final par MIS_vers_Adj un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ induisant au moins $|V'|$ adjacences, et par conséquent au moins k adjacences.

($\Leftarrow$) Supposons qu'il existe un couplage maximal $(G_0^M,G_1^M,\mathcal{M})$ de (G_0,G_1) induisant au moins k adjacences. Nous construisons un ensemble V' en prenant chaque sommet correspondant à un couple de duos de (G_0^M,G_1^M). Par construction de G, V' est un ensemble indépendant (aucune paire d'adjacences ne crée de conflit), et nous avons par suite $|V'| \geq k$. □

Lemme 14. *Soient G_0 et G_1 les deux génomes équilibrés tels que $occ(G_0) = occ(G_1) = k$. Soit $\Delta(G)$ le degré maximum du graphe des conflits G obtenu par* ConstruireMIS(G_0,G_1). *Alors nous avons $\Delta \leq 6(k-1)$.*

Démonstration. Soit $occ(G_0) = k$. Considérons un couple de duos $m = (d_0,d_1)$ tel que $d_0 = (G_0[i], G_0[i+1])$, $1 \leq i \leq \eta_{G_0} - 1$, et $d_1 = (G_1[j],G_1[j+1])$, $1 \leq j \leq \eta_{G_1} - 1$. Nous affirmons que le sommet v_m de G, correspondant au couple de duos m, est adjacent à au plus $6(k-1)$ sommets. Pour cela, nous listons les couples de duos $m' = (d'_0,d'_1)$ tels que le sommet $v_{m'}$ de G, correspondant au couple de duos m', est adjacent à v_m. Remarquons que si $v_{m'}$ est adjacent à v_m (i.e. m et m' sont en conflit), alors au moins l'un des duos d'_0 et d'_1 chevauche, respectivement, d_0 ou d_1. Soit d'_0 un duo de G_0 chevauchant d_0. Nous listons tout d'abord les différents duos d'_1 pour lesquels les couples de duos $m = (d_0,d_1)$ et $m' = (d'_0,d'_1)$ sont en conflit. Remarquons que d'_0 (ou $-d'_0$) apparaît au plus k fois dans G_1 puisqu'une famille peut apparaître au plus k fois dans G_0 ou G_1. Nous distinguons les trois cas suivants :

(a) $d'_0 = (G_0[i-1],G_0[i])$: si d'_0 (ou $-d'_0$) apparaît k fois dans G_1, l'une de ces occurrences est nécessairement $d'_1 = (G_1[j-1],G_1[j])$ si $d_0 = d_1$, ou $d'_1 = (G_1[j+1],G_1[j+2])$ si $d_0 = -d_1$. Pour chacun de ces deux cas, les couples de duos m et (d'_0,d'_1) ne sont pas en conflit.

(b) $d'_0 = d_0$: si d'_0 (ou $-d'_0$) apparaît k fois dans G_1, l'une de ces occurrences est nécessairement d_1, n'induisant aucun conflit avec m.

(c) $d'_0 = (G_0[i+1],G_0[i+2])$: si d'_0 (ou $-d'_0$) apparaît k fois dans G_1, l'une de ces occurrences est nécessairement $d'_1 = (G_1[j+1],G_1[j+2])$ si $d_0 = d_1$, ou $d'_1 = (G_1[j-1],G_1[j])$ si $d_0 = -d_1$. Pour chacun de ces deux cas, les couples de duos m et m' ne sont pas en conflit.

Ainsi, pour chacun des cas, l'un des k duos d'_1 possibles n'implique pas de conflit entre m et m'. Par conséquent, pour tout duo d'_0 chevauchant d_0, il existe au plus $k-1$ duos d'_1 dans G_1 tels que m et m' sont en conflit. En utilisant les même arguments, nous pouvons prouver que pour tout duo d'_1 chevauchant d_1, il existe au plus $k-1$ duos d'_0 de G_0 tels que m et m' sont en conflit. Par suite, chacun des six duos chevauchant d_0 ou d_1 induisent au plus $k-1$ conflits. L'Exemple 3.10 illustre le cas où le graphe construit est de degré $6(k-1)$ avec $k = 2$. Ainsi, nous obtenons au plus $6(k-1)$ sommets adjacents au sommet v_m dans le graphe G. □

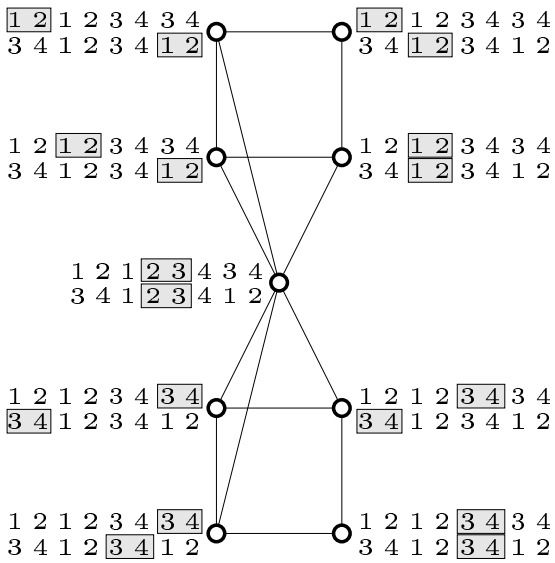


Figure 3.10 – Le graphe des conflits obtenu par ConstruireMIS(G_0,G_1) avec $G_0 = +1+2+1+2+3+4+3+4$ et $G_1 = +3+4+1+2+3+4+1+2$. Les signes des gènes, tous positifs, ne sont pas représentés pour une meilleure lisibilité. Nous avons $k = occ(G_0) = 2$. Le degré maximum du graphe obtenu est égal à $6(k-1) = 6$.

Une conséquence du Lemme 13 est que toute α-approximation pour le problème *MIS* induit une α-approximation pour *Eq-ADJ$_M$*. Dans [BF94], Berman et Fürer proposent un algorithme d'approximation pour *MIS* avec un ratio dépendant du degré Δ du graphe. Pour tous $\Delta > 2$ et $\epsilon > 0$, le ratio d'approximation est $\frac{\Delta+3}{5} + \epsilon$ si Δ est pair, et $\frac{\Delta+3.25}{5} + \epsilon$ si Δ est impair. Par le Lemme 14, nous obtenons le résultat suivant.

Théorème 9. *Pour tout $\epsilon > 0$, Eq-k-ADJ$_M$ est $(\frac{6k-3}{5} + \epsilon)$-approximable.*

Notons que dans le cas où $k = 2$, nous obtenons un ratio égal à $1.8 + \epsilon$, ce qui est au-delà de celui obtenu par le Théorème 8. De plus, nous introduisons dans la prochaine section une 4-approximation dans le cas général. Par conséquent, le seul résultat intéressant du Théorème 9 ci-dessus est le cas où $k = 3$, induisant une $(3+\epsilon)$-approximation (Algorithme 3.2) pour le problème *Eq-3-ADJ$_M$* :

Corollaire 3. *Pour tout $\epsilon > 0$, le problème Eq-3-ADJ$_M$ est $(3+\epsilon)$-approximable.*

3.7.3 4-approximation pour *Eq-ADJ$_M$*

Nous présentons dans cette partie une 4-approximation pour le problème *Eq-ADJ$_M$* en utilisant le problème Maximum Weighted 2-Interval Pattern (noté *MW-2-IP*) pour lequel une 4-approximation est donnée dans [CHLV05].

Le problème Maximum Weighted 2-Interval Pattern (*MW-2-IP*). Avant de définir le problème en lui-même, nous introduisons d'abord la notion de 2-intervalle.

ALG. 3.2 – Algorithme Approx-Eq-3-Adj$_M$

{**Entrée :** Deux génomes équilibrés G_0 et G_1 avec $occ(G_0) = occ(G_1) = 3$}
{**Sortie :** Un *couplage maximal* $(G_0^M, G_1^M, \mathcal{M})$ de (G_0,G_1)}

1 $G = (V,E) \leftarrow$ ConstruireMIS(G_0,G_1).
2 Invocation de l'algorithme d'approximation de ratio $(3+\epsilon)$ (Berman et Fürer [BF94]) afin d'obtenir un ensemble indépendant V' de G.
3 Construction du *couplage maximal* $(G_0^M,G_1^M,\mathcal{M}) \leftarrow$ MIS_vers_Adj(V',G_0,G_1).

Définition 32. ***(2-intervalle** [TH79, GW79])*
Un 2-intervalle est une paire d'intervalles disjoints définis sur une même ligne.

Par la suite, pour tout 2-intervalle $D = (I,J)$, nous supposerons toujours que $I < J$, i.e. I est strictement à gauche de J. Nous appelons deux 2-intervalles $D_0 = (I_0,J_1)$ et $D_1 = (I_1,J_2)$ des *intervalles disjoints* si D_0 et D_1 ne se chevauchent pas, i.e. $(I_0 \cup J_1) \cap (I_1 \cup J_2) = \emptyset$. Nous notons les trois relations possibles entre deux 2-intervalles disjoints de la manière suivante :

1. $D_0 \prec D_1$ (D_0 précède D_1), si $I_0 < J_1 < I_1 < J_2$,
2. $D_0 \sqsubset D_1$ (D_0 est inclus dans D_1), si $I_1 < I_0 < J_1 < J_2$ et
3. $D_0 \between D_1$ (D_0 chevauche D_1), si $I_0 < I_1 < J_1 < J_2$.

Une paire de 2-intervalles (D_0,D_1) est dite R-$comparable$ pour $R \in \{\prec, \sqsubset, \between\}$, si soit $(D_0,D_1) \in R$ soit $(D_1,D_0) \in R$. Un ensemble de 2-intervalles $\mathcal{D}$ est dit $\mathcal{R}$-comparable pour un $\mathcal{R} \subseteq \{\prec, \sqsubset, \between\}$, $\mathcal{R} \neq \emptyset$, si toute paire de 2-intervalles distincts de $\mathcal{D}$ est R-comparable pour un $R \in \mathcal{R}$. L'ensemble non vide $\mathcal{R}$ est appelé un $\mathcal{R}$-*modèle*. Le problème Max-Weighted 2-Interval Pattern est alors défini de la façon suivante.

Problème : Max-Weighted 2-Interval Pattern (*MW-2-IP*)
Entrée : Un ensemble $\mathcal{D}$ de 2-intervalles, un $\mathcal{R}$-modèle $\mathcal{R} \subseteq \{\prec, \sqsubset, \between\}$ avec $\mathcal{R} \neq \emptyset$, et une fonction de poids $\omega : \mathcal{D} \rightarrow \mathbb{N}^+$. Un entier p.
Problème de décision : Existe-t-il un sous-ensemble $\mathcal{R}$-comparable D' de $\mathcal{D}$ tel que le poids de D' (i.e. la somme des poids de ses éléments) soit supérieur ou égal à p ?
Problème d'optimisation : Trouver un sous-ensemble $\mathcal{R}$-comparable D' de $\mathcal{D}$ maximisant le poids de D'.

De nombreux travaux ont été réalisés sur la complexité du problème *MW-2-IP* selon le $\mathcal{R}$-modèle choisi [MV80, BYHN$^+$02, Via04], et plusieurs algorithmes d'approximation existent [Via04, CHLV05]. Nous proposons une transformation d'une instance de *Eq-ADJ$_M$* en instance de *MW-2-IP*, ceci afin de profiter pleinement des algorithmes d'approximation existants pour résoudre *Eq-ADJ$_M$*.

Transformation. Nous décrivons tout d'abord comment transformer toute instance (G_0, G_1) de *Eq-ADJ$_M$* en une instance de *MW-2-IP*, notée par la suite par Construire2I$(G_0, G_1) = (\mathcal{D}, \mathcal{R}, \omega)$.

Nous introduisons tout d'abord une nouvelle définition. Soient deux génomes équilibrés G_0 et G_1. Un intervalle I_0 de G_0 et un intervalle I_1 de G_1, tous deux de taille supérieure ou égale à 2, sont dits *identiques* s'ils correspondent à la même chaîne à une inversion près, où l'inversion change aussi tous les signes de la chaîne. Clairement, deux intervalles identiques ont la même longueur.

Nous construisons l'instance du problème *MW-2-IP* de la manière suivante. L'ensemble $\mathcal{D}$ est obtenu en concaténant G_0 et G_1, et pour chaque paire (I_0, I_1) d'intervalles identiques (où I_0 est un intervalle de G_0 et I_1 est un intervalle de G_1), nous construisons le 2-intervalle $D = (I_0, I_1)$ de poids $\omega(D) = |I_0| - 1$ $(= |I_1| - 1)$ puis l'ajoutons à $\mathcal{D}$. Notons que, puisque deux intervalles identiques ont une taille supérieure ou égale à 2, chaque 2-intervalle de $\mathcal{D}$ a un poids d'au moins 1. La Figure 3.11 donne un exemple d'une telle construction. Observons que, par construction, aucune paire de 2-intervalles de $\mathcal{D}$ n'est $\{\prec\}$-comparable. Pour finir de construire l'instance de *MW-2-IP* voulue, nous fixons $\mathcal{R} = \{\prec, \sqsubset, \between\}$, i.e. nous regardons tous les 2-intervalles disjoints, quelle que soit la relation entre ces 2-intervalles. Par conséquent, afin d'alléger les notations, nous noterons simplement Construire2I$(G_0, G_1) = (\mathcal{D}, \omega)$ et omettrons de mentionner le modèle.

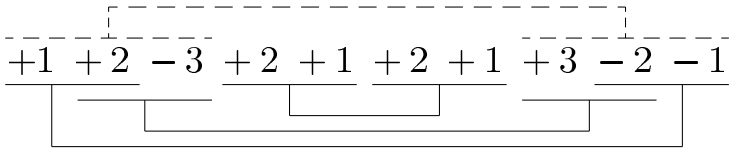


Figure 3.11 – Résolution de *Eq-ADJ$_M$* : illustration de la transformation Construire2I. L'instance de *MW-2-IP* présentée est induite par les génomes $G_0 = +1\ +2\ -3\ +2\ +1$ et $G_1 = +2\ +1\ +3\ -2\ -1$. Les pointillés correspondent aux 2-intervalles de poids égal à 2, alors que les autres ont un poids de 1.

Nous décrivons maintenant la transformation de toute solution de *MW-2-IP* en une solution de *Eq-ADJ$_M$*. Soient deux génomes équilibrés G_0 et G_1 et soit $(\mathcal{D}, \omega)$ une instance du problème *MW-2-IP* obtenue par Construire2I(G_0, G_1). De plus, soit un ensemble $\mathcal{S} \subseteq \mathcal{D}$ de 2-intervalles disjoints (i.e. une solution de *MW-2-IP* pour le modèle $\{\prec\ ,\ \sqsubset\ ,\ \between\}$ et pour l'instance $(\mathcal{D}, \omega)$).

Nous définissons la transformation Max-W2IP_vers_Adj$(\mathcal{S}$,G_0,$G_1)$ qui associe à $(\mathcal{S}$,G_0,$G_1)$ le *couplage maximal* $(G_0^M, G_1^M, \mathcal{M})$ de $(G_0$,$G_1)$ construit de la manière suivante. Tout d'abord, pour chaque 2-intervalle $D = (I_0$,$I_1)$ de $\mathcal{S}$, nous mettons en correspondance les gènes de I_0 et I_1 de manière naturelle ; puis, afin d'achever le *couplage maximal* (puisqu'un gène n'est pas nécessairement contenu dans un 2-intervalle de $\mathcal{S}$), nous appliquons l'algorithme glouton suivant : nous couplons itérativement deux gènes non couplés g_0 et g_1 tels que $|g_0| = |g_1|$ et tels que g_i est un gène de G_i ($i = 0$,1), jusqu'à ce qu'il n'existe plus de telle paire. Après un renommage des gènes selon le couplage obtenu (noté $\mathcal{M}$), nous obtenons un *couplage maximal* $(G_0^M, G_1^M, \mathcal{M})$ de (G_0, G_1).

La définition de cette construction nous permet d'obtenir les deux lemmes suivants.

Lemme 15. *Soient deux génomes équilibrés G_0 et G_1. Soit $(\mathcal{D}, \omega)$ une instance du problème MW-2-IP obtenue par* Construire2I$(G_0$,$G_1)$. *Soit un ensemble $\mathcal{S}$ de 2-intervalles disjoints de $\mathcal{D}$, et soit $W_\mathcal{S}$ le poids total de $\mathcal{S}$. Alors le* couplage maximal $(G_0^M, G_1^M, \mathcal{M})$ *de (G_0, G_1) obtenu par la transformation* Max-W2IP_vers_Adj$(\mathcal{S}$,G_0,$G_1)$ *induit au moins $W_\mathcal{S}$ adjacences.*

Démonstration. Pour tout 2-intervalle $D = (I_0, I_1)$ de $\mathcal{S}$, les gènes de I_0 et I_1 ont été mis en correspondance dans le couplage de façon naturelle. Ainsi, pour chaque 2-intervalle $D = (I_0, I_1)$ de $\mathcal{S}$, nous

obtenons $|I_0|-1$ adjacences dans $(G_0^M, G_1^M, \mathcal{M})$ puisque I_0 et I_1 sont des intervalles identiques. Puisque l'algorithme glouton final de Max-W2IP_vers_Adj($\mathcal{S}$,G_0,G_1) ne retire aucune de ces adjacences, nous avons au moins $W_{\mathcal{S}}$ adjacences dans $(G_0^M, G_1^M, \mathcal{M})$. □

Lemme 16. *Soient deux génomes équilibrés G_0 et G_1, et soit un* couplage maximal $(G_0^M, G_1^M, \mathcal{M})$ *de* $(G_0$,$G_1)$. *Soit* $(\mathcal{D}, \omega)$ *une instance du problème MW-2-IP obtenue par* Construire2I$(G_0$,$G_1)$. *Soit A le nombre d'adjacences induites par $(G_0^M, G_1^M, \mathcal{M})$. Alors, il existe un sous-ensemble $\mathcal{S} \subseteq \mathcal{D}$ de 2-intervalles disjoints de poids total supérieur ou égal à A.*

Démonstration. Considérons l'ensemble des positions $\{p_1, p_2, \ldots, p_b\}$ tel que les seuls points de cassure de $(G_0^M$,$G_1^M)$ sont ceux induits par les duos $(G_0^M[p_i]$,$G_0^M[p_i+1])$, $1 \leqslant i \leqslant b$. Ainsi, il existe b points de cassure entre G_0^M et G_1^M, et par conséquent $a = \eta_{G_0^M} - b - 1$ adjacences. Soit $\{c_1, c_2, \ldots, c_{b+1}\}$ l'ensemble des sous-chaînes de G_0^M tel que leur concaténation (dans ce même ordre) est égale à G_0^M et tel que $c_1 = [1$,$p_1]_{G_0^M}$, $c_{b+1} = [p_b$,$\eta_{G_0^M}]_{G_0^M}$ et $c_i = [p_{i-1}+1$,$p_i]_{G_0^M}$ pour tout $2 \leqslant i \leqslant b$. En d'autres termes, l'ensemble des chaînes c_i, $1 \leqslant i \leqslant b+1$ correspond aux sous-chaînes de G_0^M séparées par un point de cassure et contenant uniquement des adjacences. Ainsi, chaque sous-chaîne c_i de taille l_i, $1 \leqslant i \leqslant b+1$, contient $l_i - 1$ adjacences. De plus, chacune des sous-chaînes c_i de G_0^M correspond à une sous-chaîne t_i de G_1^M telle que c_i et t_i sont identiques. Nous construisons l'ensemble $\mathcal{S}$ de 2-intervalles comme l'union de $D_i = (\hat{c_i}, \hat{t_i})$, $1 \leq i \leq p$, où $\hat{c_i}$ (resp. $\hat{t_i}$) est l'intervalle obtenu à partir de c_i (resp. t_i). Cette construction implique que les 2-intervalles construits sont disjoints, et par conséquent que le poids total de $\mathcal{S}$ est $\sum_{i=1}^{b+1}(l_i - 1) = \sum_{i=1}^{b+1} l_i - \sum_{i=1}^{b+1} 1 = \eta_{G_0^M} - b - 1 = a$. □

ALG. 3.3 – Algorithme Approx-Eq-Adj$_M$

{**Entrée :** Deux génomes équilibrés G_0 et G_1}
{**Sortie :** Un *couplage maximal* $(G_0^M, G_1^M, \mathcal{M})$ de $(G_0$,$G_1)$}

1 $(\mathcal{D}, \omega) \leftarrow$ Construire2I$(G_0$,$G_1)$.
2 Invocation de l'algorithme d'approximation de ratio 4 (Crochemore et al. [CHLV05]) afin d'obtenir un ensemble $\mathcal{S} \subseteq \mathcal{D}$ de 2-intervalles disjoints.
3 Construction du *couplage maximal* $(G_0^M$,G_1^M,$\mathcal{M}) \leftarrow$ Max-W2IP_vers_Adj$(\mathcal{S}$,G_0,$G_1)$.

Nous proposons maintenant l'Algorithme 3.3 nommé Approx-Eq-Adj$_M$ pour la résolution du problème *Eq-ADJ$_M$* et prouvons ci-dessous le théorème suivant.

Théorème 10. *L'Algorithme* Approx-Eq-Adj$_M$ *est un algorithme d'approximation de ratio* 4 *pour Eq-ADJ$_M$.*

Démonstration. Selon les lemmes 15 et 16, il existe un *couplage maximal* $(G_0^M$,G_1^M,$\mathcal{M})$ de (G_0, G_1) induisant A adjacences si et seulement si il existe un sous-ensemble $\mathcal{S} \subseteq \mathcal{D}$ de 2-intervalles disjoints d'un poids total A. Ainsi, tout ratio d'approximation pour *MW-2-IP* implique le même ratio d'approximation pour *Eq-ADJ$_M$*. Dans [CHLV05], un algorithme d'approximation de ratio 4 a été proposé pour *MW-2-IP*. Par conséquent, l'Algorithme Approx-Eq-Adj$_M$ est une 4-approximation pour *Eq-ADJ$_M$*. □

3.8 Conclusion

Dans ce chapitre, nous avons étudié la complexité des différents problèmes abordés dans ce mémoire. En particulier, nous avons montré que les problèmes ADJ_X et BD_X sont équivalents pour les modèles *couplage exemplaire* et *couplage maximal*. En revanche, les problèmes ADJ_I et BD_I se révèlent être deux problèmes distincts. Nous avons également dans cette partie étudié les problèmes ZBD_X. Plus précisément, nous avons prouvé que les problèmes ZBD_E et ZBD_I sont **NP**-Complets même appliqués à des génomes G_0 et G_1 tels que $occ(G_0) = 2$ et où $occ(G_1)$ n'est pas borné. Notons que ce résultat a été depuis amélioré par les travaux récents de Blin et al. [BFSV09], montrant que ZBD_E est **NP**-Complet même appliqué à des paires de génomes de type $(2,2)$. Le problème ZBD_M se révèle quant à lui polynomial.

Nous avons également proposé trois algorithmes d'approximation pour le problème *Eq-k-*ADJ_M : de ratio 1.1442 pour k égal à 2, de ratio $(3 + \epsilon)$ lorsque k est égal à 3 (pour tout $\epsilon > 0$), et enfin de ratio 4 dans le cas général.

Enfin, nous avons montré que les problèmes $ICOM_X$, $ICONS_X$ et BD_X sont **APX**-Difficiles même appliqués à des paires de génomes de type $(1,2)$. Par conséquent, ces problèmes n'admettent pas de schéma d'approximation. Néanmoins, des algorithmes d'approximation peuvent être envisagés en se limitant tout d'abord au cas de paires de génomes équilibrées. Kolman et ses collaborateurs ont déjà étudié cette piste pour BD_M et les ratios d'approximation proposés peuvent, peut-être, être améliorés ([GKZ04, KW06]). Une première perspective pour de futurs travaux serait de rechercher des algorithmes d'approximation pour les modèles *couplage exemplaire* et *couplage intermédiaire* (problèmes BD_E et BD_I). Pour les problèmes $ICOM_X$ et $ICONS_X$, aucun algorithme d'approximation n'a également vu le jour, et ce même pour deux génomes équilibrés, laissant un vaste champ d'étude à investir.

CHAPITRE 4

Algorithmes d'optimisation de mesures inter-génomiques

4.1 Introduction

Nous avons étudié dans le chapitre précédent la complexité théorique des problèmes BD_X, ADJ_X, $ICOM_X$ et $ICONS_X$. En particulier, nous avons prouvé que ces problèmes sont **APX**-Difficiles même appliqués à des paires de génomes de type $(1,2)$. Dans ce chapitre, nous proposons plusieurs algorithmes de résolution des problèmes BD_X, ADJ_X, $ICOM_X$ et $ICONS_X$. Tout d'abord, nous présentons un algorithme exact basé sur une transformation du problème à résoudre en un problème pseudo-booléen. Puis, nous proposons deux approches non exactes résolvant également ces mêmes problèmes : une heuristique et une méthode hybride qui s'appuie à la fois sur la méthode exacte et sur l'heuristique. Enfin, nous présentons une étude de ces différentes approches proposées en nous appuyant sur un jeu de données composé de douze génomes de gamma-protéobactéries.

Les travaux portant sur les problèmes $ICOM_X$ ont été publiés dans [AFRV06, AFRV07]. L'étude des problèmes BD_X et ADJ_X a quant à elle été publiée dans [AFR$^+$07, AFR$^+$08b]. Ces études ont été le fruit d'un travail réalisé avec Guillaume Fertin, Irena Rusu, Annelyse Thévenin et Stéphane Vialette.

4.2 Méthodes exactes

Nous présentons dans cette section une approche exacte pour la résolution des problèmes BD_X, ADJ_X, $ICOM_X$ et $ICONS_X$. Cette approche se base sur une transformation de l'instance à résoudre en un problème pseudo-booléen. Dans un premier temps, nous définissons de manière générale ce qu'est un problème pseudo-booléen. Nous présentons ensuite les transformations que nous proposons, tout d'abord pour $ICOM_X$ et $ICONS_X$, puis pour les problèmes BD_X et ADJ_X.

4.2.1 Programme pseudo-booléen

Un programme *Pseudo-booléen linéaire* (appelé LPB) est un programme linéaire [Sch98] (i.e. un problème d'optimisation où la fonction objectif et les contraintes sont toutes des fonctions linéaires des variables) pour lequel toutes les variables sont booléennes, c'est-à-dire de valeur 0 ou 1. Les contraintes

d'un programme LPB sont ainsi des inégalités (ou des égalités) entre sommes pondérées des variables et la fonction objectif consiste à maximiser ou à minimiser une somme pondérée des variables. Nous donnons en illustration un exemple de programme pseudo-booléen linéaire en Figure 4.1.

Exemple de programme LPB

Fonction objectif :

$\texttt{Maximiser} \sum_{1 \leqslant i \leqslant 10} (x_i - y_i)$

Variables :

$X = \{x_i : 1 \leqslant i \leqslant 10\}$
$Y = \{y_i : 1 \leqslant i \leqslant 10\}$

Contraintes :

(C1) $x_4 + y_4 = 2$
(C2) $\forall i, 1 \leqslant i \leqslant 9, x_{i+1} - x_i \leqslant 0$
(C3) $\forall i, 1 \leqslant i \leqslant 9, y_i + y_{i+1} < 2$

Domaines des variables :

$\forall x_i \in X, x_i \in \{0,1\}$ $\quad$ $\forall y_i \in Y, y_i \in \{0,1\}$

Figure 4.1 – Exemple de programme pseudo-booléen (LPB). D'un point de vue du problème de satisfaction de formules booléennes, les contraintes décrites correspondent aux formules booléennes suivantes : (C1) $x_4 \wedge y_4$, (C2) $\forall i, 1 \leqslant i \leqslant 9$, $x_{i+1} \Rightarrow x_i$ et (C3) $\forall i, 1 \leqslant i \leqslant 9$, $\overline{y_i} \vee \overline{y_{i+1}}$.

Les programmes LPB sont vus par la communauté de la programmation linéaire comme une restriction de la programmation linéaire générale. Pour d'autres, qui voient ces programmes linéaires sous la forme de problèmes *SAT*, les contraintes pseudo-booléennes peuvent être vues comme une *généralisation* des clauses fournissant une extension des contraintes propositionnelles [CK03, ES06].

Les problèmes LPB peuvent se résoudre par des solveurs dédiés aux problèmes linéaires en nombres entiers (ILP). En effet, les problèmes LPB peuvent être vus comme des problèmes ILP particuliers en raison du domaine des variables restreint aux valeurs 0 et 1. L'inconvénient de cette approche est que de tels solveurs génériques ignorent la nature des variables qui sont ici booléennes, impliquant souvent des temps de calcul longs. D'un autre côté, les problèmes de décision LPB peuvent être transformés aisément en instances de problèmes *SAT* sous une forme normale conjonctive appelée CNF (i.e. conjonctions de disjonctions de littéraux booléens), lesquelles peuvent être résolues par une approche hautement spécialisée aux problèmes *SAT*. Cependant, le nombre de clauses obtenues lors de la transformation depuis un problème LPB peut être très grand.

Les solveurs de satisfiabilité booléenne actuels sont le résultat de plusieurs décennies de recherches et sont réputés être parmi les solveurs les plus efficaces dédiés à un problème **NP**-Difficile. La dernière génération de solveurs *SAT* a généralement les trois caractéristiques principales suivantes : une sélection aléatoire des variables, une recherche en backtracking, un apprentissage des clauses. De plus, ces solveurs ont habituellement des temps d'exécution raisonnables, même pour de très grandes instances.

Plusieurs généralisations de solveurs *SAT* dédiés à la résolution des problèmes LPB ont été proposés dans un passé récent (`Pueblo` [SS06], `Galena` [CK03], `OPBDP` [Bar95], `minisat+` [ES06]). Parmi cet ensemble de solveurs, `minisat+` s'est révélé être, sur un jeu de données réel (cf. Section 4.4), le plus efficace en terme de rapidité de calcul, confirmant ainsi ses bons résultats obtenus lors d'évaluations dédiées*.

4.2.2 Approche pseudo-booléenne pour le calcul des intervalles communs ou conservés

Nous proposons dans cette section de résoudre les problèmes *ICOM$_E$*, *ICOM$_M$*, *ICONS$_E$* et *ICONS$_M$* par une transformation en problème pseudo-booléen. Nous présentons tout d'abord la transformation pour le problème *ICOM$_M$*. Puis, nous présenterons les modifications à apporter afin de résoudre les problèmes *ICOM$_E$*, *ICONS$_E$* , *ICONS$_M$*, *ICOM$_I$* et *ICONS$_I$*.

Transformation de *ICOM$_M$* en problème LPB. Nous présentons ici la transformation du problème *ICOM$_M$* en un problème LPB, laquelle est présentée dans la Figure 4.2. Nous supposons ici que toutes les familles $f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}$ apparaissent à la fois dans G_0 et dans G_1 ; si tel n'est pas le cas, un pré-traitement supprimera de G_0 et de G_1 les gènes des familles apparaissant uniquement sur un seul génome.

Nous pouvons voir que le programme $\mathsf{LPB}-\mathsf{ICOM}_M$ donné en Figure 4.2 est un programme pseudo-booléen, c'est-à-dire un programme linéaire à variables booléennes. Nous divisons les variables booléennes en deux ensembles C et X. L'ensemble C contient les variables correspondant aux intervalles communs possibles, alors que l'ensemble X contient les variables correspondant au couplage des gènes que l'on construit.

Nous décrivons maintenant les contraintes présentées à la Figure 4.2. Les contraintes (`C.01`) et (`C.02`) sont nécessaires afin de tester la validité du couplage obtenu par l'assignation des variables. Chaque gène de G_0 est couplé avec au plus un gène de G_1, et inversement, chaque gène de G_1 est couplé avec au plus un gène de G_0. Certains gènes seront en conséquence non couplés, dans le cas de génomes non-équilibrés.

Les contraintes (`C.03`) assurent que chaque intervalle commun est compté exactement une fois. L'idée principale est d'imposer que la propriété dite des *bords actifs* soit vérifiée, i.e. si une variable $c^{i,j}_{k,\ell}$ vaut 1 alors, les gènes $G_0[i]$ et $G_0[j]$ doivent être couplés avec des gènes distincts situés entre les positions k et ℓ dans G_1, et que les gènes $G_1[k]$ et $G_1[\ell]$ doivent être couplés avec des gènes distincts situés entre les positions i et j dans G_0. Intuitivement, cette contrainte assure qu'un intervalle commun ne sera pas compté plus d'une fois, ceci en forçant les deux gènes aux extrémités d'un intervalle commun à être couplés.

Les contraintes (`C.04`), (`C.05`), (`C.06`) et (`C.07`) assurent que si $c^{i,j}_{k,\ell} = 1$ alors la sous-chaîne $[G_0[i],G_0[j]]_{G_0}$ de G_0 est un intervalle commun selon le couplage induit par l'assignation des variables de X tel que la sous-chaîne correspondante sur G_1 est $[G_1[k],G_1[\ell]]_{G_1}$. Par exemple, la contrainte (`C.04`) assure que chaque gène de $[G_0[i],G_0[j]]_{G_0}$ est soit non couplé, soit couplé avec un gène de $[G_1[k],G_1[\ell]]_{G_1}$ (grâce aux contraintes (`C.01`), (`C.02`) et (`C.03`), les gènes aux positions i et j dans G_0 sont toujours couplés à des gènes distincts de G_1 si $i < j$ et si $c^{i,j}_{k,\ell} = 1$).

Enfin, les contraintes (`C.08`) imposent un couplage maximal entre G_0 et G_1.

Nous en déduisons donc :

*SAT COMPETITION, http://www.satcompetition.org/

Programme LPB−ICOM$_M$

Fonction objectif :

Maximiser $\sum_{c^{i,j}_{k,\ell} \in C} c^{i,j}_{k,\ell}$

Variables :

$C = \{c^{i,j}_{k,\ell} : 1 \leqslant i \leqslant j \leqslant \eta_{G_0} \wedge 1 \leqslant k \leqslant \ell \leqslant \eta_{G_1}\}$

$X = \{x^i_k : 1 \leqslant i \leqslant \eta_{G_0} \wedge 1 \leqslant k \leqslant \eta_{G_1} \wedge G_0[i] = G_1[k]\}$

Contraintes :

(C.01) $\forall i = 1,2,\ldots,\eta_{G_0}, \quad \sum_{\substack{1 \leqslant k \leqslant \eta_{G_1} \\ G_0[i]=G_1[k]}} x^i_k \leqslant 1$

(C.02) $\forall k = 1,2,\ldots,\eta_{G_1}, \quad \sum_{\substack{1 \leqslant i \leqslant \eta_{G_0} \\ G_0[i]=G_1[k]}} x^i_k \leqslant 1$

(C.03) $\forall c^{i,j}_{k,\ell} \in C, \quad 4\,c^{i,j}_{k,\ell} - \sum_{\substack{k \leqslant r \leqslant \ell \\ G_0[i]=G_1[r]}} x^i_r - \sum_{\substack{k \leqslant s \leqslant \ell \\ G_0[j]=G_1[s]}} x^j_s - \sum_{\substack{i \leqslant p \leqslant j \\ G_0[p]=G_1[k]}} x^p_k - \sum_{\substack{i \leqslant q \leqslant j \\ G_0[q]=G_1[\ell]}} x^q_\ell \leqslant 0$

(C.04) $\forall c^{i,j}_{k,\ell} \in C,\ \forall i < p < j, \forall 1 \leqslant r < k, \quad G_0[p] = G_1[r], \quad c^{i,j}_{k,\ell} + x^p_r \leqslant 1$

(C.05) $\forall c^{i,j}_{k,\ell} \in C,\ \forall i < p < j, \forall \ell < r \leqslant \eta_{G_1}, \ G_0[p] = G_1[r], \quad c^{i,j}_{k,\ell} + x^p_r \leqslant 1$

(C.06) $\forall c^{i,j}_{k,\ell} \in C,\ \forall k < r < \ell, \forall 1 \leqslant p < i, \quad G_0[p] = G_1[r], \quad c^{i,j}_{k,\ell} + x^p_r \leqslant 1$

(C.07) $\forall c^{i,j}_{k,\ell} \in C,\ \forall k < r < \ell, \forall j < p \leqslant \eta_{G_0}, \ G_0[p] = G_1[r], \quad c^{i,j}_{k,\ell} + x^p_r \leqslant 1$

(C.08) $\forall f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}, \quad \sum_{\substack{1 \leqslant i \leqslant \eta_{G_0} \\ G_0[i]=f}} \sum_{\substack{1 \leqslant k \leqslant \eta_{G_1} \\ G_1[k]=f}} x^i_k = \min\{occ(f,G_0), occ(f,G_1)\}$

Domaines des variables :

$\forall x^i_k \in X, \quad x^i_k \in \{0,1\}$

$\forall c^{i,j}_{k,\ell} \in C, \quad c^{i,j}_{k,\ell} \in \{0,1\}$

Figure 4.2 – Programme LPB−ICOM$_M$ pour la recherche du nombre maximum d'intervalles communs entre deux génomes sous le modèle *couplage maximal* (*ICOM$_M$*)

Proposition 3. *Le programme* LPB−ICOM$_M$ *calcule correctement le nombre maximum d'intervalles communs entre G_0 et G_1 sous le modèle* couplage maximal.

Nous précisons ici la taille du programme généré LPB−ICOM$_M$. Tout d'abord, il est aisé de voir que $|C| = \Theta(\eta^2_{G_0}\,\eta^2_{G_1})$ et ainsi que (C.03) est composée de $\Theta(\eta^2_{G_0}\,\eta^2_{G_1})$ contraintes. Le nombre de contraintes de (C.04) à (C.07) dépend du nombre de gènes dupliqués dans les deux génomes. Dans le pire des cas, c'est-à-dire lorsqu'il n'existe qu'une seule famille de gènes, le nombre de contraintes de (C.04) à (C.07) est en $\Theta(\eta^3_{G_0}\,\eta^3_{G_1})$. Nous avons également $|X| = \mathcal{O}(\eta_{G_0}\,\eta_{G_1})$. Clairement, la taille de l'ensemble X détermine exactement le nombre de contraintes (C.01) et (C.02). Enfin, le nombre de contraintes (C.08) est linéaire en la taille des deux génomes. Nous décrirons plus tard dans la Section 4.2.4 comment accélérer la résolution du programme en réduisant le nombre de variables et le nombre de contraintes. Avant cela, nous montrons dans la section suivante comment modifier le programme LPB−ICOM$_M$ afin de l'adapter, respectivement, aux modèles *couplage exemplaire* et *couplage intermédiaire* ainsi qu'à la mesure *nombre d'intervalles conservés*.

4.2.3 Adaptation du programme pour les autres modèles et aux problèmes $ICONS_X$

Dans cette section, nous adaptons le programme LPB−ICOM$_M$ afin de calculer un couplage maximisant le nombre d'intervalles communs sous le modèle *couplage exemplaire* ou sous le modèle *couplage intermédiaire*. Puis, nous discuterons des modifications à apporter pour adapter ce programme à la maximisation de la mesure *nombre d'intervalles conservés*.

Nous observons que le remplacement des contraintes (`C.08`) par les contraintes (`C.08'`) – définies ci-après – dans le Programme LPB−ICOM$_M$ induit le programme pseudo-booléen LPB−ICOM$_E$ qui calcule le nombre maximum d'intervalles communs entre les génomes G_0 et G_1 sous le modèle *couplage exemplaire*.

(`C.08'`) $\forall f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}, \quad \sum_{\substack{1 \leq i \leq \eta_{G_0} \\ G_0[i]=f}} \; \sum_{\substack{1 \leq k \leq \eta_{G_1} \\ G_1[k]=f}} x_k^i = 1$

De la même manière, la substitution des contraintes (`C.08`) par les contraintes (`C.08''`) – définies ci-dessous – dans le Programme LPB−ICOM$_M$ induit le programme pseudo-booléen LPB−ICOM$_I$ calculant le nombre maximum d'intervalles communs entre G_0 et G_1 sous le modèle *couplage intermédiaire*.

(`C.08''`) $\forall f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}, \quad \sum_{\substack{1 \leq i \leq \eta_{G_0} \\ G_0[i]=f}} \; \sum_{\substack{1 \leq k \leq \eta_{G_1} \\ G_1[k]=f}} x_k^i \geq 1$

Afin d'étudier la mesure *nombre d'intervalles conservés*, nous pouvons aisément modifier les programmes proposés ci-dessus pour les adapter à cette mesure. Rappelons qu'un intervalle conservé (cf. Section 2.3) est un intervalle commun avec des restrictions supplémentaires sur ses extrémités. Nous devons pour cela sélectionner, dans l'ensemble C, uniquement les variables correspondant à un possible intervalle conservé. Dans notre cas, l'ensemble des variables C devient alors :

$C = \{c_{k,\ell}^{i,j} : 1 \leq i \leq j \leq \eta_{G_0} \wedge 1 \leq k \leq \ell \leq \eta_{G_1} \wedge$
$(\; (\; G_0[i] = G_1[k] \wedge G_0[j] = G_1[\ell] \;) \vee (\; G_0[i] = -G_1[k] \wedge G_0[j] = -G_1[\ell] \;) \;)\}$

4.2.4 Amélioration du programme pseudo-booléen

Nous donnons dans cette section plusieurs règles de simplification valables uniquement pour le Programme LPB−ICOM$_M$. En théorie, une instance de très grande taille d'un problème pseudo-booléen peut être facile à résoudre alors qu'une petite instance peut s'avérer très difficile. Cependant, les petites instances difficiles sont des problèmes très spécifiques, (e.g. le problème `pigeonhole` [Ajt88]), et par conséquent, dans les cas pratiques, les temps d'exécution dépendent majoritairement de la taille des instances. L'idée principale ici est de réduire le nombre de variables et de contraintes du programme pseudo-booléen généré. Plus précisément, nous appliquons des règles permettant d'éviter l'ajout de variables inutiles $c_{k,\ell}^{i,j}$ dans C tout en maintenant le Programme LPB−ICOM$_M$ correct ; deux de ces règles peuvent toutefois modifier le nombre d'intervalles communs obtenus par la résolution du programme pseudo-booléen. De ce fait, un calcul post-traitement sera nécessaire afin d'obtenir une solution valide.

Nous présentons maintenant les différentes règles réduisant le nombre de variables et le nombre de contraintes.

[**Règle 1**] Retirer de C toutes les variables $c_{k,k}^{i,i}$, $1 \leq i \leq \eta_{G_0}$ et $1 \leq k \leq \eta_{G_1}$.

L'application de la **Règle 1** modifie le nombre d'intervalles communs obtenus entre G_0 et G_1. En effet, les intervalles communs singletons (i.e. de taille égale à 1) n'apparaissent plus dans la fonction objectif du programme pseudo-booléen. Par conséquent, un post-traitement doit être appliqué afin d'ajouter ces intervalles communs manquant. Nous pouvons aisément calculer les nombres d_1 et d_2 correspondant aux nombres de gènes qui doivent être retirés dans G_0 et G_1 afin d'obtenir un *couplage maximal* entre les deux génomes. Ainsi, nous savons que les génomes résultants (i.e. après suppression des gènes non couplés) contiennent chacun $L = \eta_{G_0} - d_1 = \eta_{G_1} - d_2$ gènes, où

$$L = \sum_{f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_0}} \min\{occ(f,G_0), occ(f,G_1)\}.$$

Chacun de ces gènes correspond à un intervalle commun de taille 1, ceci quel que soit le couplage maximal choisi. Nous pouvons donc retirer toutes les variables correspondantes et ajouter L au nombre d'intervalles communs entre G_0 et G_1 trouvé par le Programme $\mathsf{LPB{-}ICOM}_M$.

[**Règle 2**] Retirer de C toutes les variables $c_{k,\ell}^{i,j}$ si au moins l'une des conditions suivantes est vraie (le symbole # indique la cardinalité de l'ensemble) :

1. $(card\{r : k \leq r \leq \ell \wedge G_0[i] = G_1[r]\} = 0) \vee (card\{s : k \leq s \leq \ell \wedge G_0[j] = G_1[s]\} = 0)$,
2. $(card\{r : k \leq r \leq \ell \wedge G_0[i] = G_1[r]\} < 2) \wedge (G_0[i] = G_0[j])$,
3. $(card\{p : i \leq p \leq j \wedge G_1[k] = G_0[p]\} = 0) \vee (card\{q : i \leq q \leq j \wedge G_1[\ell] = G_0[q]\} = 0)$,
4. $(card\{p : i \leq p \leq j \wedge G_1[k] = G_0[p]\} < 2) \wedge (G_1[k] = G_1[\ell])$.

La **Règle 2** est une amélioration des contraintes (`C.03`). En effet, ces contraintes assurent que chaque intervalle commun ne soit compté qu'une seule fois par le programme en forçant les extrémités de chaque intervalle à être couplées dans la solution obtenue, (i.e. les gènes $G_0[i]$ et $G_0[j]$ sont couplés à deux gènes distincts situés entre k et ℓ (inclus) dans G_1, et les gènes $G_1[k]$ et $G_1[\ell]$ sont couplés à des gènes distincts situés entre i et j (inclus) dans G_0). La validité de la **Règle 2** découle du fait que le Programme $\mathsf{LPB{-}ICOM}_M$ fixera toujours une variable $c_{k,\ell}^{i,j}$ à 0 si la propriété des bords ne peut être satisfaite.

[**Règle 3**] Retirer de C toutes les variables $c_{k,\ell}^{i,j}$ pour lesquelles il existe au moins une famille de gènes $f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}$ telle que $|occ(f,G_0,i,j) - occ(f,G_1,k,l)| > |occ(f,G_0) - occ(f,G_1)|$.

La **Règle 3** nous évite de supprimer trop de gènes dans un intervalle commun de (G_0,G_1). En effet, pour toute famille $f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}$, le nombre $|occ(f,G_0,i,j) - occ(f,G_1,k,l)|$ correspond au nombre minimum d'occurrences de f qui doivent être retirées si $c_{k,\ell}^{i,j} = 1$; i.e. $[G_0[i],G_0[j]]_{G_0}$ est un intervalle commun de (G_0,G_1) et sa permutation sur G_1 est la sous-chaîne $[G_1[k],G_1[\ell]]_{G_1}$. De plus, $|occ(f,G_0) - occ(f,G_1)|$ correspond au nombre d'occurrences de f à retirer dans G_0 ou dans G_1 afin d'obtenir un *couplage maximal*. La validité de la **Règle 3** découle du fait que nous ne devons pas retirer plus de $|occ(f,G_0) - occ(f,G_1)|$ occurrences de la famille f.

4.2.5 Adaptation du programme pour les génomes circulaires

La notion d'intervalle commun a été définie dans ce mémoire pour deux génomes, ceux-ci étant représentés par des séquences signées de gènes. Dans l'optique d'une application pratique pour des

génomes bactériens, nous devons prendre en compte la circularité des génomes. Pour cela, uniquement dans cette section, nous interprétons les séquences signées représentant les génomes comme des séquences circulaires. En d'autres termes, nous supposons que le dernier élément de la séquence précède immédiatement le premier. Nous étendons également la définition d'un intervalle commun afin de permettre à un intervalle commun de chevaucher les extrémités de la séquence. Par exemple, considérons les génomes G_0 et G_1 suivants : $G_0 = +1 + 2 + 3 + 4 + 5$ et $G_1 = +2 - 4 + 3 + 5 + 1$. Alors, la sous-chaîne $[+5, +1]_{G_1}$ sera également un intervalle commun dans cette version étendue de la définition.

Nous présentons maintenant les trois programmes $\mathsf{LPB{-}ICOM{-}CIRC}_M$, $\mathsf{LPB{-}ICOM{-}CIRC}_E$ et $\mathsf{LPB{-}ICOM{-}CIRC}_I$ qui résolvent respectivement les problèmes *ICOM$_M$*, *ICOM$_E$* et *ICOM$_I$* avec la notion d'intervalle commun étendue présentée ci-dessus. Nous supposons ici que toutes les familles $f \in \mathcal{F}_{G_0} \cup \mathcal{F}_{G_1}$ apparaissent à la fois dans G_0 et dans G_1 ; si tel n'est pas le cas, un pré-traitement supprimera de G_0 et de G_1 les gènes des familles apparaissant uniquement sur un seul génome. Nous définissons tout d'abord les variables, puis la fonction objectif et enfin les différentes contraintes nécessaires, suivant les modèles *couplage exemplaire* et *couplage maximal*.

Variables et fonction objectif. Nous définissons deux types de variables : les *variables de couplage* et les *variables d'intervalle*. Une *variable de couplage* est définie pour chaque paire de gènes pouvant être couplée. Une telle variable sera égale à 1 si et seulement si les gènes correspondants sont couplés. Ainsi, nous définissons l'ensemble des *variables de couplage* de la manière suivante :

$$X = \{x_j^i : 0 \leqslant i < \eta_{G_0} \wedge 0 \leqslant j < \eta_{G_1} \wedge |G_0[i]| = |G_1[j]|\}$$
$$\forall x_j^i \in X, x_j^i \in \{0,1\}$$

Les *variables d'intervalle* correspondent aux intervalles communs possibles. Notons que le génome entier G_0 est nécessairement un intervalle commun. Par conséquent, nous ne considérons pas les intervalles couvrant tout le génome (c'est-à-dire les intervalles $G_0[G_0[i],G_0[i-1]]$, pour $2 \leqslant i \leqslant \eta_{G_0}$, et l'intervalle $G_0[1,G_0[\eta_{G_0}]]$), ceci afin de réduire la taille du programme pseudo-booléen généré. Les *variables d'intervalle* sont définies de la manière suivante :

$$C = \{c_{j,m}^{i,n} : 0 \leqslant i < \eta_{G_0} \wedge 0 \leqslant j < \eta_{G_1} \wedge 0 \leqslant n < \eta_{G_0} - 1 \wedge 0 \leqslant m < \eta_{G_1} - 1\}$$
$$\forall c_{j,m}^{i,n} \in C, c_{j,m}^{i,n} \in \{0,1\}$$

Une variable $c_{j,m}^{i,n} \in C$ correspond à l'intervalle commun commençant à la position i sur G_0 et contenant $n+1$ gènes. Cet intervalle commun correspond à l'intervalle sur G_1 commençant à la position j et contenant $m+1$ gènes. Enfin, nous définissons la fonction objectif comme la somme des variables de type c, et ce afin de maximiser le nombre d'intervalle communs.

Programme LPB−ICOM−CIRC$_M$

Fonction objectif :

$$\texttt{Maximiser} \sum_{c^{i,n}_{j,m} \in C} c^{i,n}_{j,m}$$

Variables :

$X = \{x^i_j : 0 \leqslant i < \eta_{G_0} \wedge 0 \leqslant j < \eta_{G_1} \wedge |G_0[i]| = |G_1[j]|\}$

$C = \{c^{i,n}_{j,m} : 0 \leqslant i < \eta_{G_0} \wedge 0 \leqslant j < \eta_{G_1} \wedge 0 \leqslant n < \eta_{G_0} - 1 \wedge 0 \leqslant m < \eta_{G_1} - 1\}$

Contraintes :

(C1.a) $\forall i, 0 \leqslant i < \eta_{G_0}, \sum_{\substack{0 \leqslant j < \eta_{G_1} \\ |G_0[i]| = |G_1[j]|}} x^i_j \leqslant 1$

(C1.b) $\forall i, 0 \leqslant j < \eta_{G_1}, \sum_{\substack{0 \leqslant i < \eta_{G_0} \\ |G_0[i]| = |G_1[j]|}} x^i_j \leqslant 1$

(C2) $\forall f \in \mathcal{F}, \sum_{\substack{0 \leqslant i < \eta_{G_0} \\ |G_0[i]| = f}} \sum_{\substack{0 \leqslant j < \eta_{G_1} \\ |G_1[j]| = f}} x^i_j = \min\{occ(f,G_0), occ(f,G_1)\}$

(C3.a) $\forall c^{i,n}_{j,m} \in C, 2c^{i,n}_{j,m} - \sum_{\substack{0 \leqslant p \leqslant m \\ |G_0[i]| = |G_1[f_1(j,p)]|}} x^i_{f_1(j,p)} - \sum_{\substack{0 \leqslant p \leqslant m \\ |G_0[f_0(i,n)]| = |G_1[f_1(j,p)]|}} x^{f_0(i,n)}_{f_1(j,p)} \leqslant 0$

(C3.b) $\forall c^{i,n}_{j,m} \in C, 2c^{i,n}_{j,m} - \sum_{\substack{0 \leqslant p \leqslant n \\ |G_0[f_0(i,p)]| = |G_1[j]|}} x^{f_0(i,p)}_j - \sum_{\substack{0 \leqslant p \leqslant n \\ |G_0[f_0(i,p)]| = |G_1[f_1(j,m)]|}} x^{f_0(i,p)}_{f_1(j,m)} \leqslant 0$

(C4.a) $\forall c^{i,n}_{j,m} \in C, j + m < \eta_{G_1}, \forall 1 \leqslant p < n, \forall 0 \leqslant r < j, |G_0[f_0(i,p)]| = |G_1[r]|, c^{i,n}_{j,m} + x^{f_0(i,p)}_r \leqslant 1$

(C4.b) $\forall c^{i,n}_{j,m} \in C, j + m < \eta_{G_1}, \forall 1 \leqslant p < n, \forall j + m < r \leqslant \eta_{G_1}, |G_0[f_0(i,p)]| = |G_1[r]|, c^{i,n}_{j,m} + x^{f_0(i,p)}_r \leqslant 1$

(C4.c) $\forall c^{i,n}_{j,m} \in C, j + m \geqslant \eta_{G_1}, \forall 1 \leqslant p < n, \forall f_1(j,m) < r < j, |G_0[f_0(i,p)]| = |G_1[r]|, c^{i,n}_{j,m} + x^{f_0(i,p)}_r \leqslant 1$

(C5.a) $\forall c^{i,n}_{j,m} \in C, i + n < \eta_{G_0}, \forall 1 \leqslant p < m, \forall 0 \leqslant r < i, |G_0[r]| = |G_1[f_1(j,p)]|, c^{i,n}_{j,m} + x^r_{f_1(j,p)} \leqslant 1$

(C5.b) $\forall c^{i,n}_{j,m} \in C, i + n < \eta_{G_0}, \forall 1 \leqslant p < m, \forall i + n < r \leqslant \eta_{G_0}, |G_0[r]| = |G_1[f_1(j,p)]|, c^{i,n}_{j,m} + x^r_{f_1(j,p)} \leqslant 1$

(C5.c) $\forall c^{i,n}_{j,m} \in C, i + n \geqslant \eta_{G_0}, \forall 1 \leqslant p < m, \forall f_0(i,n) < r < i, |G_0[r]| = |G_1[f_1(j,p)]|, c^{i,n}_{j,m} + x^r_{f_1(j,p)} \leqslant 1$

Domaines des variables :

$\forall x^i_j \in X, x^i_j \in \{0,1\} \qquad \forall c^{i,n}_{j,m} \in C, c^{i,n}_{j,m} \in \{0,1\}$

Notations :

Nous notons $f_q(i,p)$, $q \in \{0,1\}$, le $p^{\text{ième}}$ gène après la position i dans G_q.
Cette notation est nécessaire afin de prendre en compte la circularité des génomes.
Mathématiquement, nous avons $f_q(i,p) \equiv (i+p) \mod \eta_{G_q}$.

Figure 4.3 – Programme LPB−ICOM−CIRC$_M$ pour la recherche du nombre maximum d'intervalles communs (version étendue aux génomes circulaires) entre deux génomes sous le modèle *couplage maximal*

Contraintes. Nous définissons les différentes contraintes permettant d'assurer la bonne assignation des variables. Pour cela, nous définissons deux contraintes pour vérifier que chaque gène ne peut être couplé plus d'une fois :

$$(C1.a)\ \forall i, 0 \leqslant i < \eta_{G_0}, \sum_{\substack{0 \leqslant j < \eta_{G_1} \\ |G_0[i]|=|G_1[j]|}} x_j^i \leqslant 1$$

$$(C1.b)\ \forall i, 0 \leqslant j < \eta_{G_1}, \sum_{\substack{0 \leqslant i < \eta_{G_0} \\ |G_0[i]|=|G_1[j]|}} x_j^i \leqslant 1$$

Ensuite, pour chaque famille de gènes, nous comptons le nombre de gènes couplés afin d'obtenir un couplage valide suivant le modèle de couplage choisi. Ainsi, pour le modèle *couplage maximal* (programme $\mathsf{LPB{-}ICOM{-}CIRC}_M$ seulement), nous définissons la contrainte suivante :

$$(C2)\ \forall f \in \mathcal{F}, \sum_{\substack{0 \leqslant i < \eta_{G_0} \\ |G_O[i]|=f}} \sum_{\substack{0 \leqslant j < \eta_{G_1} \\ |G_1[j]|=f}} x_j^i = \min\{occ(f,G_0), occ(f,G_1)\}$$

De même, pour le modèle *couplage exemplaire* (programme $\mathsf{LPB{-}ICOM{-}CIRC}_E$ seulement), nous définissons :

$$(C2')\ \forall f \in \mathcal{F}, \sum_{\substack{0 \leqslant i < \eta_{G_0} \\ |G_1[i]|=f}} \sum_{\substack{0 \leqslant j < \eta_{G_1} \\ |G_1[j]|=f}} x_j^i = 1$$

Pour le modèle *couplage intermédiaire* (programme $\mathsf{LPB{-}ICOM{-}CIRC}_I$ seulement), nous définissons :

$$(C2'')\ \forall f \in \mathcal{F}, \sum_{\substack{0 \leqslant i < \eta_{G_0} \\ |G_1[i]|=f}} \sum_{\substack{0 \leqslant j < \eta_{G_1} \\ |G_1[j]|=f}} x_j^i \geqslant 1$$

Afin de contrôler l'assignation des *variables d'intervalle*, nous introduisons ici une nouvelle notation. Pour tout entier $q \in \{0,1\}$, soit $f_q(i,p) \equiv (i+p) \mod \eta_{G_q}$, avec $0 \leqslant i \leqslant \eta_{G_q}$ et $0 \leqslant p \leqslant \eta_{G_q}$. Cette notation est nécessaire afin de prendre en compte les intervalles communs contenant les gènes $G_q[0]$ et $G_q[\eta_{G_q} - 1]$.

Tout d'abord, nous devons nous assurer que chaque extrémité d'un intervalle commun est couplé, ceci grâce aux contraintes suivantes :

$$(C3.a)\ \forall c^{i,n}_{j,m} \in C, 2c^{i,n}_{j,m} - \sum_{\substack{0\leqslant p\leqslant m \\ |G_0[i]|=|G_1[f_1(j,p)]|}} x^{i}_{f_1(j,p)} - \sum_{\substack{0\leqslant p\leqslant m \\ |G_0[f_0(i,n)]|=|G_1[f_1(j,p)]|}} x^{f_0(i,n)}_{f_1(j,p)} \leqslant 0$$

$$(C3.b)\ \forall c^{i,n}_{j,m} \in C, 2c^{i,n}_{j,m} - \sum_{\substack{0\leqslant p\leqslant n \\ |G_0[f_1(i,p)]|=|G_1[j]|}} x^{f_1(i,p)}_{j} - \sum_{\substack{0\leqslant p\leqslant n \\ |G_0[f_0(i,p)]|=|G_1[f_1(j,m)]|}} x^{f_0(i,p)}_{f_1(j,m)} \leqslant 0$$

Ensuite, nous définissons les contraintes permettant d'assurer que chaque gène de G_0 contenu dans un intervalle commun I est correctement couplé. Pour cela, nous devons considérer deux cas. Si l'intervalle correspondant à I sur G_1 ne contient pas les deux gènes $G_1[0]$ et $G_1[\eta_{G_1} - 1]$, nous écrivons :

$$(C4.a)\ \forall c^{i,n}_{j,m} \in C, j+m < \eta_{G_1}, \forall 1 \leqslant p < n, \forall 0 \leqslant r < j,$$
$$|G_0[f_0(i,p)]| = |G_1[r]|, c^{i,n}_{j,m} + x^{f_0(i,p)}_{r} \leqslant 1$$

$$(C4.b)\ \forall c^{i,n}_{j,m} \in C, j+m < \eta_{G_1}, \forall 1 \leqslant p < n, \forall j+m < r \leqslant \eta_{G_1},$$
$$|G_0[f_0(i,p)]| = |G_1[r]|, c^{i,n}_{j,m} + x^{f_0(i,p)}_{r} \leqslant 1$$

Sinon (l'intervalle correspondant à I sur G_1 contient les deux gènes $G_1[0]$ et $G_1[\eta_{G_1} - 1]$), nous écrivons :

$$(C4.c)\ \forall c^{i,n}_{j,m} \in C, j+m \geqslant \eta_{G_1}, \forall 1 \leqslant p < n, \forall f_1(j,m) < r < j,$$
$$|G_0[f_0(i,p)]| = |G_1[r]|, c^{i,n}_{j,m} + x^{f_0(i,p)}_{r} \leqslant 1$$

Enfin, nous définissons également les trois contraintes $(C5.a)$, $(C5.b)$ et $(C5.c)$, symétriques de $(C4.a)$, $(C4.b)$ et $(C4.c)$, afin de considérer chaque gène de G_1. La Figure 4.3 reprend dans son ensemble le programme pseudo-booléen $\mathsf{LPB}-\mathsf{ICOM}-\mathsf{CIRC}_M$.

4.2.6 Approche pseudo-booléenne pour le calcul du nombre d'adjacences ou du nombre de points de cassure

Nous proposons dans cette section une approche pseudo-booléenne afin de résoudre les problèmes *BD*$_X$ et *ADJ*$_X$. Cette approche est la même que celle présentée dans la section précédente s'intéressant aux intervalles communs et conservés. Nous présentons tout d'abord le programme pseudo-booléen pour le problème *ADJ*$_M$, que nous noterons $\mathsf{LPB}-\mathsf{ADJ}_M$. Puis nous décrivons dans la Section 4.2.8 les modifications à apporter pour traiter les autres problèmes étudiés.

Le programme $\mathsf{LPB}-\mathsf{ADJ}_M$ prend en entrée deux génomes G_0 et G_1 de tailles respectives η_{G_0} et η_{G_1}. Ce programme est présenté dans la Figure 4.4. Nous présentons maintenant en détail les variables, les contraintes et la fonction objectif du programme linéaire généré.

Programme LPB−ADJ$_M$

Fonction objectif :

Maximiser $\sum_{0\leqslant i<\eta_{G_0}} \sum_{i<j\leqslant\eta_{G_0}} \sum_{0\leqslant k<\eta_{G_1}} \sum_{k<\ell\leqslant\eta_{G_1}} d(i,j,k,\ell)$

Variables :

$A = \{a(i,k) : 1 \leqslant i \leqslant \eta_{G_0} \wedge 1 \leqslant k \leqslant \eta_{G_1}\}$
$B = \{b_x(i) : x \in \{0,1\} \wedge 1 \leqslant i \leqslant \eta_{G_x}\}$
$C = \{c_x(i,j) : x \in \{0,1\} \wedge 1 \leqslant i \leqslant j \leqslant \eta_{G_x}\}$
$D = \{d(i,j,k,l) : 1 \leqslant i \leqslant j \leqslant \eta_{G_0} \wedge 1 \leqslant k \leqslant l \leqslant \eta_{G_1}\}$

Contraintes :

(D.01) $\forall\, 1 \leqslant i \leqslant \eta_{G_0}, \sum_{\substack{1\leqslant k\leqslant\eta_{G_1} \\ |G_0[i]|=|G_1[k]|}} a(i,k) = b_0(i)$

$\forall\, 1 \leqslant k \leqslant \eta_{G_1}, \sum_{\substack{1\leqslant i\leqslant\eta_{G_0} \\ |G_0[i]|=|G_1[k]|}} a(i,k) = b_1(k)$

(D.02) $\forall\, x \in \{0,1\}, \forall\, f \in \mathcal{F}_{G_x}, \sum_{\substack{1\leqslant i\leqslant n_x \\ |G_x[i]|=f}} b_x(i) = \min(occ(f,G_0), occ(f,G_1))$

(D.03) $\forall\, x \in \{0,1\}, \forall\, 1 \leqslant i \leqslant j-1 < n_x, c_x(i,j) + \sum_{i<p<j} b_x(p) \geq 1$

(D.04) $\forall\, x \in \{0,1\}, \forall\, 1 \leqslant i < p < j \leqslant n_x, c_x(i,j) + b_x(p) \leqslant 1$

(D.05) $\forall\, 1 \leqslant i < j \leqslant \eta_{G_0}, \forall\, 1 \leqslant k < \ell \leqslant \eta_{G_1}$, tel que $G_0[i] = G_1[k]$ et $G_0[j] = G_1[\ell]$,
$a(i,k) + a(j,\ell) + c_0(i,j) + c_1(k,\ell) - d(i,j,k,\ell) \leqslant 3$

(D.06) $\forall\, 1 \leqslant i < j \leqslant \eta_{G_0}, \forall\, 1 \leqslant k < \ell \leqslant \eta_{G_1}$, tel que $G_0[i] = G_1[k]$ et $G_0[j] = G_1[\ell]$,
$a(i,k) - d(i,j,k,\ell) \geq 0$
$a(j,\ell) - d(i,j,k,\ell) \geq 0$
$c_0(i,j) - d(i,j,k,\ell) \geq 0$
$c_1(k,\ell) - d(i,j,k,\ell) \geq 0$

(D.07) $\forall\, 1 \leqslant i < j \leqslant \eta_{G_0}, \forall\, 1 \leqslant k < \ell \leqslant \eta_{G_1}$, tel que $G_0[i] = -G_1[\ell]$ et $G_0[j] = -G_1[k]$,
$a(i,\ell) + a(j,k) + c_0(i,j) + c_1(k,\ell) - d(i,j,k,\ell) \leqslant 3$

(D.08) $\forall\, 1 \leqslant i < j \leqslant \eta_{G_0},\ \forall\, 1 \leqslant k < \ell \leqslant \eta_{G_1}$, tel que $G_0[i] = -G_1[\ell]$ et $G_0[j] = -G_1[k]$,
$a(i,\ell) - d(i,j,k,\ell) \geq 0$
$a(j,k) - d(i,j,k,\ell) \geq 0$
$c_0(i,j) - d(i,j,k,\ell) \geq 0$
$c_1(k,\ell) - d(i,j,k,\ell) \geq 0$

(D.09) $\forall\, 1 \leqslant i < j \leqslant \eta_{G_0},\ \forall\, 1 \leqslant k < \ell \leqslant \eta_{G_1}$,
tel que $\{|G_0[i]|, |G_0[j]|\} \neq \{|G_1[k]|, |G_1[\ell]|\}$ ou $G_0[i] - G_0[j] \neq G_1[k] - G_1[\ell]$,
$d(i,j,k,\ell) = 0$

(D.10) $\forall\, 1 \leqslant i < j \leqslant \eta_{G_0}$,
$\sum_{1\leqslant k<\eta_{G_1}} \sum_{k<\ell\leqslant\eta_{G_1}} d(i,j,k,\ell) \leqslant 1$

Domaines des variables :

$\forall\, x \in \{0,1\}, \forall\, 1 \leqslant i < j \leqslant \eta_{G_0}, \forall\, 1 \leqslant k < \ell \leqslant \eta_{G_1}, \qquad a(i,k), b_x(i), c_x(i,k), d(i,j,k,\ell) \in \{0,1\}$

Figure 4.4 – Programme LPB−ADJ$_M$ calculant le nombre maximum d'adjacences entre deux génomes pour le modèle *couplage maximal* (*ADJ$_M$*).

Variables. Nous présentons ici les différentes variables du programme.

- Les variables $a(i,k)$, $1 \leqslant i \leqslant \eta_{G_0}$ et $1 \leqslant k \leqslant \eta_{G_1}$, définissent le couplage $\mathcal{M}$: nous aurons $a_{i,k} = 1$ si et seulement si $G_0[i]$ est couplé avec $G_1[k]$ dans $\mathcal{M}$.
- Les variables $b_x(i)$, $x \in \{0,1\}$ et $1 \leqslant i \leqslant \eta_{G_x}$, indiquent si le gène $G_x[i]$ est couplé dans $\mathcal{M}$: nous aurons $b_x(i) = 1$ si et seulement si $G_x[i]$ est couplé dans $\mathcal{M}$. Nous aurons obligatoirement $\sum_{1 \leqslant i \leqslant \eta_{G_0}} b_0(i) = \sum_{1 \leqslant k \leqslant \eta_{G_1}} b_1(k)$, ceci correspondant exactement à la taille du couplage $\mathcal{M}$.
- Les variables $c_x(i,j)$, $x \in \{0,1\}$ et $1 \leqslant i < j \leq \eta_{G_x}$, représentent les *gènes consécutifs* selon le couplage $\mathcal{M}$: nous aurons ainsi $c_x(i,j) = 1$ si et seulement si $G_x[i]$ et $G_x[j]$ sont couplés dans $\mathcal{M}$ et si aucun gène $G_x[p]$, $i < p < j$, n'est couplé dans $\mathcal{M}$.
- Les variables $d(i,j,k,\ell)$, $1 \leqslant i < j \leqslant \eta_{G_0}$ et $1 \leqslant k < \ell \leqslant \eta_{G_1}$, correspondent aux *adjacences* selon le couplage $\mathcal{M}$: nous aurons $d(i,j,k,\ell) = 1$ si et seulement si les trois conditions suivantes sont vérifiées :
  - Soit les paires $(G_0[i], G_1[k])$ et $(G_0[j], G_1[\ell])$ appartiennent au couplage $\mathcal{M}$, avec $G_0[i] = G_1[k]$ et $G_0[j] = G_1[\ell]$, soit les paires $(G_0[i], G_1[\ell])$ et $(G_0[j], G_1[k])$ appartiennent à $\mathcal{M}$, avec $G_0[i] = -G_1[\ell]$ et $G_0[j] = -G_1[k]$
  - $G_0[i]$ et $G_0[j]$ sont consécutifs dans G_0 selon le couplage $\mathcal{M}$
  - $G_1[k]$ et $G_1[\ell]$ sont consécutifs dans G_1 selon le couplage $\mathcal{M}$

Contraintes. Nous présentons ici les contraintes du programme $\mathsf{LPB{-}ADJ}_M$.

Supposons que $x \in \{0,1\}$, $1 \leqslant i < j \leqslant \eta_{G_0}$ et $1 \leqslant k < \ell \leqslant \eta_{G_1}$.

- Les contraintes (`D.01`) assurent que chaque gène de G_0 et de G_1 est couplé au plus une fois, condition nécessaire pour que le couplage $\mathcal{M}$ soit valide. Rappelons que $b_0(i) = 1$ (resp. $b_1(k) = 1$) si et seulement si le gène $G_0[i]$ (resp. $G_1[k]$) est couplé ; ainsi pour un i (resp. un k) fixé, la somme calculée (somme des $b_0(i)$ à vrai, resp. somme des $b_1(k)$ à vrai) devra nécessairement être inférieure ou égale à 1. La Figure 4.5 illustre cette contrainte. Remarquons que quel que soit le couplage $\mathcal{M}$, deux gènes couplés appartiennent nécessairement à la même famille de gènes. Par conséquent, nous ne générerons pas les variables $a(i,k) = 0$ dans le cas où $G_0[i]$ et $G_1[k]$ seront issus de deux familles différentes (i.e. $|G_0[i]| \neq |G_1[k]|$).
- Les contraintes (`D.02`) définissent le type de couplage (ici le modèle *couplage maximal*). Pour chaque famille de gènes f, exactement $\min(occ(f,G_0), occ(f,G_1))$ occurrences de f dans G_0 et dans G_1 doivent figurer dans le couplage $\mathcal{M}$. La Figure 4.5 illustre précisément cette contrainte.
- Les contraintes (`D.03`) et (`D.04`) indiquent si deux gènes seront successifs après le $\mathcal{M}$-élagage. La variable $c_x(i,j)$ est vraie (i.e. égale à 1) si et seulement si il n'existe aucune position p telle que $i < p < j$ et telle que $b_x(p) = 1$. Notons que, par ces seules contraintes, nous pouvons avoir $c_x(i,j) = 1$ même si l'un des gènes $G_x[i]$ ou $G_x[j]$ n'est pas couplé dans $\mathcal{M}$.
- Les contraintes de (`D.05`) à (`D.10`) définissent les variables d. Dans le cas où $G_0[i] = G_1[k]$ et $G_0[j] = G_1[\ell]$, les contraintes (`D.05`) et (`D.06`) assurent que $d(i,j,k,\ell) = 1$ si et seulement si toutes les variables $a(i,k)$, $a(j,\ell)$, $c_0(i,j)$ et $c_1(k,\ell)$ sont vraies (égales à 1). Dans le cas où $G_0[i] = -G_1[\ell]$ et $G_0[j] = -G_1[k]$, les contraintes (`D.07`) et (`D.08`) assurent que $d(i,j,k,\ell) = 1$ si et seulement si toutes les variables $a(i,\ell)$, $a(j,k)$, $c_0(i,j)$ et $c_1(k,\ell)$ sont vraies. Les contraintes (`D.09`) assurent que les variables $d(i,j,k,\ell)$ sont fausses si aucun des deux cas précédents n'est vérifié. Enfin, pour $1 \leqslant i < j \leqslant \eta_{G_0}$ fixés, les contraintes (`D.10`) assurent qu'il existe au plus une variable d concernant i et j pour laquelle l'assignation est 1 (voir Figure 4.6 pour une illustration).

L'objectif du Programme $\mathsf{LPB{-}ADJ}_M$ est de maximiser le nombre d'adjacences entre les deux

génomes considérés. Conformément à ce qui précède, cet objectif se réduit dans notre modèle à maximiser la somme de toutes les variables $d(i,j,k,\ell)$.

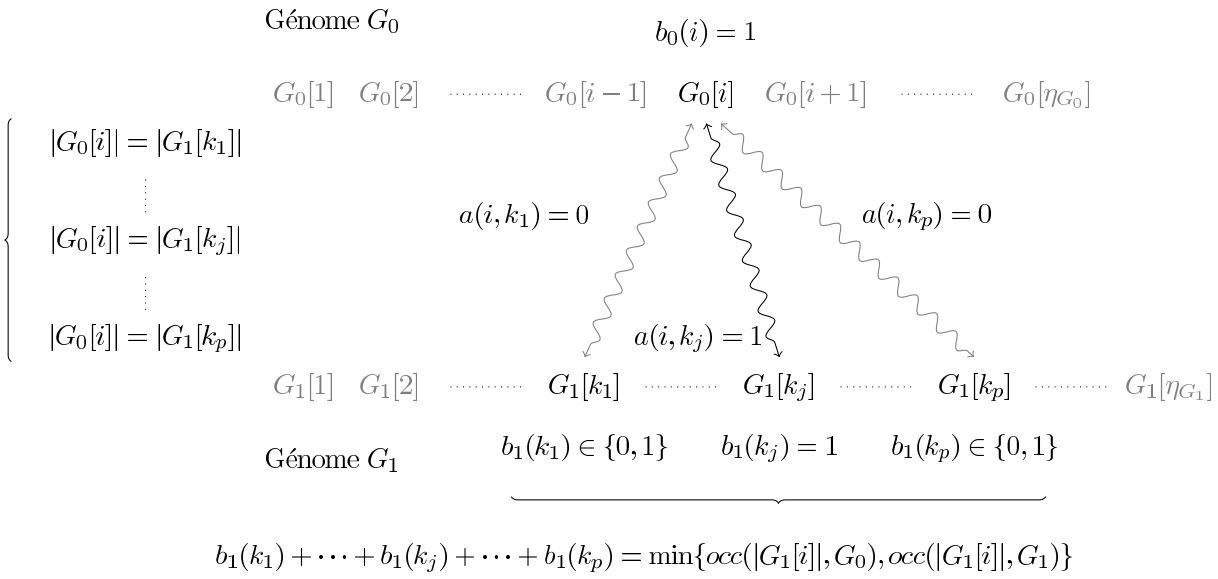


Figure 4.5 – Programme $\mathsf{LPB{-}ADJ}_M$. Illustration des contraintes sur la variable $b_0(i)$, $1 \leqslant i \leqslant \eta_{G_0}$. Si le gène $G_0[i]$ apparaît aux positions $k_1 < k_2 < \ldots < k_p$ dans G_1 et que le gène $G_0[i]$ est couplé avec le gène $G_1[k_j]$, $1 \leqslant j \leqslant p$, dans le couplage correspondant à l'assignation des variables, alors nous avons les quatre propriétés suivantes : (i) $a(i,k_j) = 1$, *i.e.*, le gène $G_0[i]$ est couplé avec le gène $G_1[k_j]$, (ii) $a(i,k_q) = 0$ pour $1 \leqslant q \leqslant p$ et $q \neq j$, *i.e.*, le gène $G_0[i]$ est ainsi couplé à un seul gène dans G_1, (iii) $b_0(i) = 1$, *i.e.*, le gène $G_0[i]$ est couplé avec un gène de G_1 et (iv) $b_1(k_j) = 1$, *i.e.*, le gène $G_1[k_j]$ est couplé avec un gène de G_0. Observons que nous pouvons avoir parfois $b_0(k_q) = 1$ avec $1 \leqslant q \leqslant p$ et $q \neq j$ si $\min(occ(|G_0[i]|,G_0), occ(|G_0[i]|,G_1) \geq 1$ (cette observation n'est plus valable pour le modèle *couplage exemplaire*).

4.2.7 Simplifications du programme $\mathsf{LPB{-}ADJ}_M$

Nous pouvons voir facilement que le Programme $\mathsf{LPB{-}ADJ}_M$ possède $O((\eta_{G_0} \cdot \eta_{G_1})^2)$ variables et $O((\eta_{G_0} \cdot \eta_{G_1})^2)$ contraintes. Afin d'améliorer la rapidité d'exécution de ce programme, nous présentons ici quelques règles permettant de réduire le nombre de variables et de contraintes générées.

Pré-traitement des génomes. En pré-traitement, nous retirons pour chaque paire de génomes les gènes qui n'apparaissent pas dans les deux génomes. Cette opération ne modifie pas le calcul du nombre d'adjacences entre deux génomes, puisque ces gènes ne pourront figurer dans un couplage des gènes en raison de l'absence d'homologue sur l'autre génome. En pratique, ce pré-traitement réduit drastiquement le nombre de variables. Par exemple, sur le jeu de données composé de γ-Protéobactéries étudié dans la Section 4.4, la taille moyenne des génomes se réduit de 3000 à 1300.

Réduction du nombre de variables et de contraintes. Pour les gènes non dupliqués, *i.e.*, les gènes g tels que $occ(g,G_0) = occ(g,G_1) = 1$, la variable correspondante $a_{i,k}$ peut être initialisée à 1, ainsi que

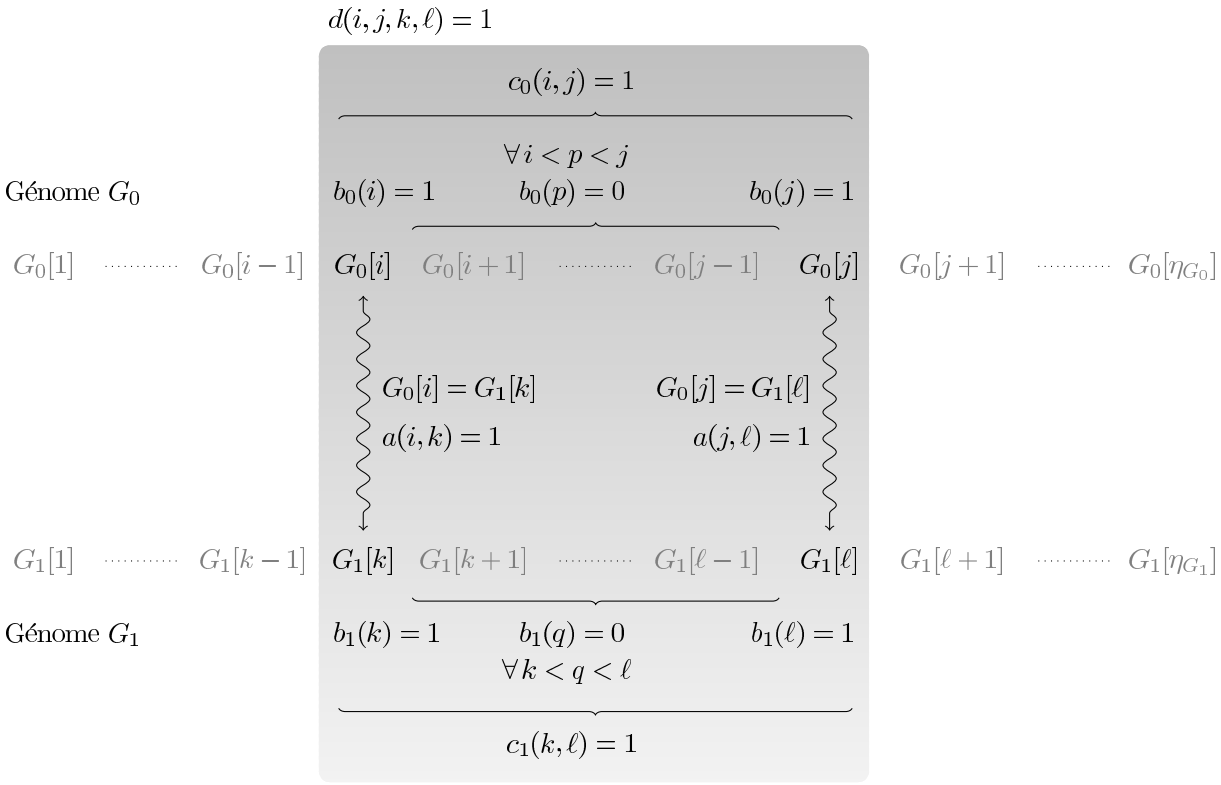


Figure 4.6 – Programme LPB−ADJ$_M$. Illustration des contraintes sur une variable $d(i,j,k,\ell)$ telle que $d(i,j,k,\ell) = 1$, avec $1 \leqslant i < j \leqslant \eta_{G_0}$ et $1 \leqslant k < \ell \leqslant \eta_{G_1}$ et $G_0[i] = G_1[k]$ et $G_0[j] = G_1[\ell]$. Les deux gènes $G_0[i]$ et $G_0[j]$ sont successifs selon le couplage obtenu par l'assignation des variables s'il existe deux gènes $G_1[k]$ et $G_1[\ell]$, avec $G_0[i] = G_1[k]$ et $G_0[j] = G_1[\ell]$, tels que (i) $G_0[i]$ est couplé à $G_1[k]$, *i.e.* $a(i,k) = 1$, (ii) $G_0[j]$ est couplé à $G_1[\ell]$, *i.e.* $a(j,\ell) = 1$, (iii) aucun gène situé entre $G_0[i]$ et $G_0[j]$ n'est couplé avec un gène de G_1, *i.e.* $c_0(i,j) = 1$ et (iv) aucun gène situé entre $G_1[k]$ et $G_1[\ell]$ n'est couplé avec un gène de G_0, *i.e.* $c_1(k,\ell) = 1$.

les variables $b_0(i)$ et $b_1(k)$. Aussi, si deux gènes non dupliqués apparaissent consécutivement ou dans l'ordre inverse avec un signe opposé, la variable correspondante d peut être initialisée directement à 1 et les contraintes correspondantes n'ont pas besoin d'être générées.

Si, pour deux gènes situés aux positions i et j dans G_0, au moins un gène situé entre i et j dans G_0 doit être nécessairement couplé dans un couplage $\mathcal{M}$ (par exemple si une famille de gènes a toutes ses occurrences entre i et j dans G_0), alors les gènes $G_0[i]$ et $G_0[j]$ ne pourront pas induire une adjacence. Ainsi, la variable correspondante $c_0(i,j)$ et les variables $d(i,j,k,\ell)$ avec $1 \leqslant k < \ell \leqslant \eta_{G_1}$ peuvent être directement initialisées à 0 et les contraintes associées peuvent ne pas être générées. Nous pouvons utiliser le même raisonnement pour deux positions k et ℓ dans G_1 et les variables $c_1(k,\ell)$ et $d(i,j,k,\ell)$ avec $1 \leqslant i < j \leqslant \eta_{G_0}$.

4.2.8 Adaptation de $\mathsf{LPB}-\mathsf{ADJ}_M$ aux autres modèles

Le Programme $\mathsf{LPB}-\mathsf{ADJ}_M$ est décrit afin de résoudre le problème *ADJ$_M$*. Dans la Section 3.3, nous avons montré par le Théorème 1 que les problèmes *ADJ$_M$* et *BD$_M$* sont équivalents dans le sens où un couplage maximal maximisant le nombre d'adjacences minimise aussi le nombre de points de cassure. Ainsi le programme $\mathsf{LPB}-\mathsf{ADJ}_M$ permet également de résoudre *BD$_M$*. Il suffira pour cela de réaliser les deux étapes suivantes : (i) appel du programme $\mathsf{LPB}-\mathsf{ADJ}_M$ résolvant *ADJ$_M$* et (ii) calcul du nombre de points de cassure en fonction du nombre d'adjacences obtenu et de la taille du couplage qui est, pour ce modèle, connu (cf. Propriété 2, page 31). Afin de résoudre l'ensemble des problèmes, nous présentons ici comment adapter le programme $\mathsf{LPB}-\mathsf{ADJ}_M$ pour traiter les problèmes *ADJ$_E$*, *BD$_E$*, *ADJ$_I$* et *BD$_I$*. Comme nous le verrons, seulement quelques modifications sont nécessaires.

Adaptation pour *ADJ$_E$* et *BD$_E$*. Comme observé précédemment, les contraintes (`D.02`) définissent le modèle considéré. Pour le modèle *couplage exemplaire*, exactement une occurrence de chaque famille de gènes doit être couplée. Par conséquent, les contraintes (`D.02`) doivent être réécrites de la manière suivante, définissant ainsi le programme $\mathsf{LPB}-\mathsf{ADJ}_E$ résolvant *ADJ$_E$*.

$$(\texttt{D.02'}) \quad \forall\, x \in \{0,1\}, \forall\, f \in \mathcal{F}_{G_x}, \sum_{\substack{1 \leqslant i \leqslant \eta_{G_x} \\ |G_x[i]|=f}} b_x(i) = 1$$

De plus, pour une meilleure efficacité, une simple règle supplémentaire peut être considérée pour simplifier le programme. En effet, nous devons avoir exactement une occurrence de chaque gène dans chaque génome. Par conséquent, pour $0 \leqslant i < j \leqslant \eta_{G_x}$, $x \in \{0,1\}$, si $|G_x[i]| = |G_x[j]|$ alors $c_x(i,j) = 0$. Les variables d correspondantes sont fixées directement à 0 et les contraintes correspondantes ne sont pas générées. Par le Théorème 1 (page 31), nous savons que toute solution optimale de *ADJ$_E$* est solution optimale de *BD$_E$*. Ainsi, pour résoudre le problème *BD$_E$*, il suffira d'appeler le programme $\mathsf{LPB}-\mathsf{ADJ}_E$ résolvant *ADJ$_E$* et de calculer ensuite le nombre de points de cassure en fonction du nombre d'adjacences obtenu et de la taille du couplage (cf. Propriété 2, page 31).

Adaptation pour *ADJ$_I$*. Encore une fois, nous modifions les contraintes (`D.02`) afin de considérer le modèle *couplage intermédiaire*. Pour ce modèle, au moins un gène de chaque famille doit être couplé. Nous réécrivons donc les contraintes (`D.02`) de la manière suivante, définissant ainsi le programme $\mathsf{LPB}-\mathsf{ADJ}_I$.

$$(\texttt{D.02''}) \quad \forall\, x \in \{0,1\}, \forall\, f \in \mathcal{F}_{G_x}, \sum_{\substack{1 \leqslant i \leqslant \eta_{G_x} \\ |G_x[i]|=f}} b_x(i) \geq 1$$

Adaptation pour *BD$_I$*. Nous savons par le Théorème 2 que les problèmes *BD$_E$* et *BD$_I$* sont équivalents en terme de nombre de points de cassure entre les deux génomes comparés. Le programme $\mathsf{LPB}-\mathsf{ADJ}_E$ présenté auparavant et qui résout *ADJ$_E$* et *BD$_E$* permet ainsi de résoudre également *BD$_I$*. Nous proposons néanmoins le programme nommé $\mathsf{LPB}-\mathsf{BD}_I$ permettant également de résoudre *BD$_I$* et qui autorise les

couplages intermédiaires. Par conséquent, les programmes $\mathsf{LPB}-\mathsf{ADJ}_E$ et $\mathsf{LPB}-\mathsf{BD}_I$ donneront systématiquement des résultats égaux en terme de nombre de points de cassure. En revanche, les couplages obtenus ne seront pas nécessairement identiques. Pour résoudre $\mathsf{LPB}-\mathsf{BD}_I$, nous modifions la fonction objectif du programme $\mathsf{LPB}-\mathsf{ADJ}_I$ de manière à minimiser le nombre de points de cassure. Par la Propriété 2 (page 31), nous savons que si la taille du couplage est connue alors nous pouvons calculer le nombre de points de cassure à partir du nombre d'adjacences. Dans notre cas, la taille du couplage est égale ici à $\sum_{1 \leqslant i \leqslant \eta_{G_0}} b_0(i)$. En conséquence, nous modifions $\mathsf{LPB}-\mathsf{ADJ}_I$ en générant la fonction objectif suivante, définissant ainsi le programme $\mathsf{LPB}-\mathsf{BD}_I$:

$$\texttt{Minimiser} \sum_{1 \leqslant i \leqslant \eta_{G_0}} b_0(i) - \sum_{1 \leqslant i < \eta_{G_0}} \sum_{i < j \leqslant \eta_{G_0}} \sum_{1 \leqslant k < \eta_{G_1}} \sum_{k < \ell \leqslant \eta_{G_1}} d(i,j,k,\ell) - 1$$

4.2.9 Conclusion

Dans cette partie, nous avons proposé diverses méthodes exactes pour le calcul des mesures *nombre de points de cassure* et *nombre d'adjacences* entre deux génomes. En particulier, nous avons défini les quatre programmes exacts $\mathsf{LPB}-\mathsf{ADJ}_E$, $\mathsf{LPB}-\mathsf{ADJ}_M$, $\mathsf{LPB}-\mathsf{ADJ}_I$ et $\mathsf{LPB}-\mathsf{BD}_I$ permettant de résoudre respectivement les problèmes ADJ_E et BD_E, ADJ_M et BD_M, ADJ_I et enfin BD_I. Ces programmes sont basés sur une transformation de l'instance à traiter en un problème pseudo-booléen. Nous étudierons dans la Section 4.4 leur qualités respectives et utiliserons les résultats optimaux obtenus afin d'estimer les différentes approches que nous présentons dans la section suivante.

4.3 Méthodes approchées

Nous décrivons dans cette section trois méthodes approchées pour la résolution des problèmes BD_X, ADJ_X, $ICOM_X$ et $ICONS_X$. Ces approches sont génériques dans le sens où elles sont indépendantes des mesures étudiées dans ce mémoire. Nous présenterons pour chacune de ces méthodes leur variante pour le modèle *couplage maximal*. Puis, nous aborderons les différentes modifications à leur apporter pour traiter les modèles *couplage exemplaire* et *couplage intermédiaire*.

4.3.1 L'heuristique ILCS_X

Nous commençons par présenter ici l'heuristique que nous appellerons ILCS_X pour Iterative Longest Common Substring. Cette heuristique, dont le fonctionnement est assez basique, n'est pas nouvelle puisque l'on retrouve son utilisation dès 1984 par Tichy [Tic84] et a depuis été utilisée dans plusieurs travaux dont [MSM04, BCF05].

L'idée directrice de cette heuristique dans le cadre de la maximisation du nombre d'intervalles communs (ou d'intervalles conservés) est qu'une sous-chaîne commune (à une inversion près) de longueur k engendre $\frac{k(k+1)}{2}$ intervalles communs. Par conséquent, la recherche de telles chaînes contribue intuitivement à augmenter le nombre total d'intervalles communs. En ce qui concerne le nombre de points de cassure et le nombre d'adjacences, une sous-chaîne commune de longueur k engendre $k-1$ adjacences et aucun point de cassure, allant ainsi dans le sens de l'optimisation de ces mesures. L'Algorithme 4.1 présente l'heuristique ILCS_X dans sa version *couplage maximal* (ILCS_M).

ALG. 4.1 – L'heuristique ILCS_M

{**Entrée :** Deux génomes G_0 et G_1}
{**Sortie :** Une mesure entre G_0 et G_1}

Tant que Il existe un gène pouvant être couplé **faire**
1 Calcul d'une plus longue sous-chaîne commune S des deux génomes G_0 et G_1 ne contenant aucun gène couplé, à une inversion près. S'il existe plusieurs sous-chaînes candidates, nous en choisissons une au hasard
2 Couplage des gènes de S de manière naturelle (i.e. soit dans le sens de lecture soit dans le sens inverse de lecture)
Fin Tant que
3 Retrait dans chaque génome de tous les gènes non couplés
4 Calcul de la mesure obtenue par le couplage

Pour les modèles *couplage exemplaire* et *couplage intermédiaire*, il est nécessaire d'apporter des modifications à cette heuristique. En effet, pour le modèle *couplage exemplaire* (heuristique ILCS_E), nous devons nous assurer qu'il n'existe pas deux gènes de la même famille dans le couplage. Pour cela, si deux gènes de la même famille apparaissent dans une sous-chaîne commune, nous ne couplons que la première occurrence et nous supprimons, une fois le couplage d'un gène g réalisé, toutes les autres occurrences de la famille $|g|$. Ainsi, par ces deux modifications, deux gènes de la même famille ne peuvent figurer tous les deux dans le couplage. Pour le modèle *couplage intermédiaire* (heuristique ILCS_I), la seule modification est la condition d'arrêt. Dans cette version, nous arrêtons les itérations lorsque, pour chaque famille de gènes, au moins une occurrence figure dans le couplage. Par la suite, l'ensemble des variantes ILCS_E, ILCS_M et ILCS_I sera noté ILCS_X.

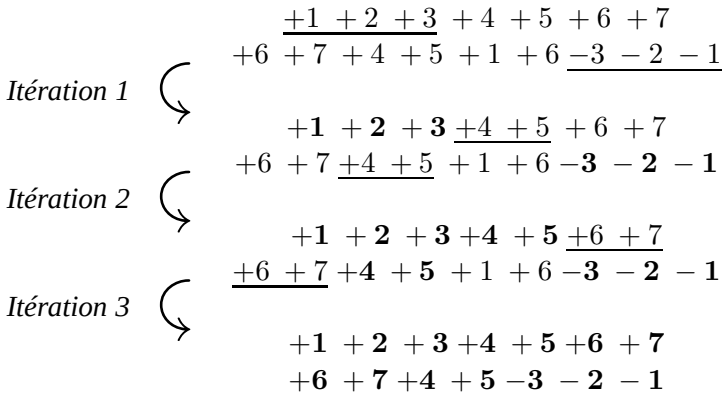
Afin d'illustrer l'heuristique ILCS_M, nous donnons l'exemple suivant.

Exemple 4.11 (Application de l'heuristique ILCS_M).
Soient deux génomes G_0 et G_1 avec :

$$G_0 = -1\ +2\ +3\ +4\ +5\ +6\ +7$$

$$G_1 = +6\ +7\ +4\ +5\ +1\ +6\ -3\ -2\ -1$$

L'exécution de l'heuristique ILCS_M est la suivante, où chaque partie soulignée représente une plus longue sous-chaîne commune et où les gènes en gras correspondent aux gènes couplés.



Si nous nous intéressons au nombre d'intervalles communs, dans cet exemple, nous en obtenons 19.

4.3.2 L'heuristique IILCS_X, une amélioration de l'heuristique ILCS_X

Un examen de l'heuristique ILCS_X nous a conduit à proposer une amélioration à celle-ci, sous la forme d'une nouvelle heuristique que nous appelons IILCS_X (pour *I*mproved *I*terative *L*ongest *C*ommon *S*ubstring). L'Algorithme 4.2 présente l'heuristique IILCS_X dans sa version *couplage maximal* (IILCS_M).

ALG. 4.2 – L'heuristique IILCS_M

{**Entrée :** Deux génomes G_0 et G_1}
{**Sortie :** Une mesure entre G_0 et G_1}

Tant que Il existe un gène pouvant être couplé **faire**
1 Calculer la plus longue sous-chaîne commune S des deux génomes G_0 et G_1 ne contenant aucun gène couplé, à une inversion près. S'il existe plusieurs sous-chaînes candidates, en choisir une au hasard
2 Coupler les gènes de S de manière naturelle (i.e. soit dans le sens de lecture soit dans le sens inverse de lecture)
3 *Retirer dans le génome G_0 (resp. G_1) les gènes non couplés pour lesquels il n'existe plus aucun gène non couplé de la même famille dans G_1 (resp. G_0)*
Fin Tant que
4 Calculer la mesure étudiée

La seule différence entre les heuristiques ILCS_M et IILCS_M se situe à la ligne 3 (cf. Algorithmes 4.2 et 4.1). Après avoir trouvé une plus longue sous-chaîne commune (à une inversion près), nous retirons dans chaque génome les gènes qui ne pourront plus figurer dans le couplage final. De tels gènes sont des gènes non couplés pour lesquels il n'existe aucun gène non couplé de la même famille sur l'autre génome. Ce traitement peut être simplement réalisé en comparant, à chaque itération, le nombre de gènes non couplés pour chaque famille de gènes dans les deux génomes. Cela permet dans la pratique de trouver parfois des sous-chaînes communes plus longues que celles obtenues par l'heuristique IILCS_M. Il est nécessaire

de modifier IILCS_M afin de l'adapter aux modèles *couplage exemplaire* et *couplage intermédiaire*. Nous effectuons pour cela les mêmes modifications que celles proposées pour ILCS_E et ILCS_I, à savoir, (i) pour le modèle *couplage exemplaire*, le couplage uniquement de la première occurrence d'une famille présente plusieurs fois dans une sous-chaîne commune et la suppression des occurrences non couplées d'une famille $|g|$ si le gène g a été couplé (variante IILCS_E) (ii) pour le modèle *couplage intermédiaire*, un arrêt du programme lorsque au moins un gène de chaque famille est couplé (variante IILCS_I). Dans ce mémoire, nous désignerons IILCS_X l'ensemble des variantes IILCS_E, IILCS_M et IILCS_I. Nous verrons dans la Section 4.4 que l'heuristique IILCS_X donne de meilleurs résultats que ILCS_X. Néanmoins, il n'est pas assuré théoriquement que IILCS_X obtienne toujours une solution de qualité supérieure ou équivalente à ILCS_X.

Afin d'illustrer l'heuristique IILCS_X, nous reprenons le même exemple que celui donné précédemment pour l'heuristique ILCS_X, toujours dans la variante *couplage maximal* (IILCS_M).

Exemple 4.12 (Application de l'heuristique IILCS_M).
Soient deux génomes G_0 et G_1 avec :

$$G_0 = +1\ +2\ +3\ +4\ +5\ +6\ +7$$

$$G_1 = +6\ +7\ +4\ +5\ +1\ +6\ -3\ -2\ -1$$

L'exécution de l'heuristique IILCS_M est décrite ci-dessous. Chaque partie soulignée représente une plus longue sous-chaîne commune. Les gènes en gras correspondent aux gènes couplés et les gènes en italique correspondent aux gènes devant être retirés (car il n'existe aucun gène non couplé de la même famille dans l'autre génome).

Itération 1

$+1\ +2\ +3\ +4\ +5\ +6\ +7$
$+6\ +7\ +4\ +5\ +1\ +6\ -3\ -2\ -1$

$\mathbf{+1\ +2\ +3}\ +4\ +5\ +6\ +7$
$+6\ +7\ +4\ +5\ +6\ \mathbf{-3\ -2\ -1}$

Itération 2

$\mathbf{+1\ +2\ +3}\ +4\ +5\ +6\ +7$
$+6\ +7\ +4\ +5\ +6\ \mathbf{-3\ -2\ -1}$

$\mathbf{+1\ +2\ +3\ +4\ +5\ +6}\ +7$
$+7\ \mathbf{+4\ +5\ +6\ -3\ -2\ -1}$

Itération 3

$\mathbf{+1\ +2\ +3\ +4\ +5\ +6}\ +7$
$+7\ \mathbf{+4\ +5\ +6\ -3\ -2\ -1}$

$\mathbf{+1\ +2\ +3\ +4\ +5\ +6\ +7}$
$\mathbf{+6\ +7\ +4\ +5\ -3\ -2\ -1}$

Si l'on s'intéresse au nombre d'intervalles, nous en obtenons dans cet exemple 20.

Remarquons que, lors de l'étape 2 des heuristiques ILCS_X et IILCS_X, nous choisissons arbitrairement une des plus longues sous-chaînes communes existantes. Ce choix peut évidemment modifier le résultat final. De par la rapidité de ces heuristiques, nous pouvons aisément appliquer plusieurs fois ces

heuristiques sur une même instance afin d'éventuellement améliorer la qualité de la solution en gardant la meilleure solution trouvée, et ceci tout en gardant des temps d'exécution très avantageux.

Comme toute heuristique, IILCS_X peut parfois donner de mauvais résultats. Nous donnons ci-après un exemple, pour la mesure *nombre d'intervalles communs*, où IILCS_M donne $O(n)$ intervalles communs alors que l'optimal donne $O(n^2)$ intervalles communs, et où la taille des génomes est $4n-2$ pour G_0 et $5n-3$ pour G_1.

Exemple 4.13 (Une paire de génomes pour lesquels l'heuristique IILCS_X est mauvaise).
Soit un entier n fixé. Nous construisons les deux génomes G_0 et G_1 suivants où tous les signes sont positifs (non représentés pour une meilleure lisibilité) :

$$G_0 = a_1, b_1, a_2, b_2, \ldots, a_n, b_n, c_1, d_1, c_2, d_2, \ldots, c_{n-1}, d_{n-1}$$

$$G_1 = d_1, d_2, \ldots, d_{n-1}, a_1, b_1, c_1, a_2, b_2, c_2, \ldots, a_{n-1}, b_{n-1}, c_{n-1}, a_n, b_n, c_1, c_2, \ldots, c_{n-1}$$

avec :
- $\forall 1 \leqslant i \leqslant n$, $a_i = i$
- $\forall 1 \leqslant i \leqslant n$, $b_i = n + i$
- $\forall 1 \leqslant i < n$, $c_i = 2n + i$
- $\forall 1 \leqslant i < n$, $d_i = 3n + i - 1$

Notons que le génome G_0 ne contient pas de gènes dupliqués et que seuls les gènes c_i, $1 \leqslant i < n$, sont dupliqués dans G_1 et apparaissent exactement deux fois. Les trois modèles de couplage sont donc équivalents sur cette instance (cf. Propriété 1, page 19). Sur cet exemple, la solution optimale du problème ICOM$_M$ est égale à $2n^2 + 5n - 1$. Pour cela, pour tout $1 \leqslant i < n$, nous couplons l'occurrence de c_i dans G_0 avec la seconde occurrence de c_i dans G_1. Nous obtenons ainsi après un $\mathcal{M}$-élagage les génomes suivants :

$$G'_0 = a_1, b_1, a_2, b_2, \ldots, a_n, b_n, c_1, d_1, c_2, d_2, \ldots, c_{n-1}, d_{n-1}$$

$$G'_1 = d_1, d_2, \ldots, d_{n-1}, a_1, b_1, a_2, b_2, \ldots, a_{n-1}, b_{n-1}, a_n, b_n, c_1, c_2, \ldots, c_{n-1}$$

En revanche, l'heuristique IILCS_M peut obtenir un nombre d'intervalles communs bien moindre. Remarquons que l'heuristique va d'abord obtenir des sous-séquences communes de taille 2 (séquences a_i, b_i, $1 \leqslant i \leqslant n$). Ensuite, lorsqu'il ne reste plus que des gènes isolés, l'heuristique couple arbitrairement un des gènes non couplés. Par ce procédé, nous pouvons dans le pire des cas obtenir la configuration suivante, induisant $5n - 1$ intervalles communs :

$$G_0 = a_1, b_1, a_2, b_2, \ldots, a_n, b_n, c_1, d_1, c_2, d_2, \ldots, c_{n-1}, d_{n-1}$$

$$G_1 = d_1, d_2, \ldots, d_{n-1}, a_1, b_1, c_1, a_2, b_2, c_2, \ldots, a_{n-1}, b_{n-1}, c_{n-1}, a_n, b_n$$

Nous obtenons ainsi par IILCS_M, dans le pire des cas, $5n - 1 = O(n)$ intervalles communs, alors que l'optimum est égal à $2n^2 + 5n - 1 = O(n^2)$.

4.3.3 La méthode hybride $\mathsf{HYB}_X(\mathsf{k})$

Nous présentons ici une méthode hybride, dans le sens où elle utilise deux méthodes présentées précédemment : l'heuristique IILCS_X et la méthode exacte basée sur une transformation en un problème pseudo-booléen. L'idée est de commencer la résolution par l'heuristique IILCS_X et d'arrêter cet

algorithme glouton lorsque les sous-chaînes communes sont de taille inférieure à un seuil k donné en paramètre. Cette étape induit une instance fictive, dans le sens où le couplage obtenu n'est pas nécessairement un couplage valide (selon le modèle choisi). Puis, nous appliquons la méthode exacte sur l'instance fictive obtenue. L'algorithme n'est évidemment pas un algorithme exact de par sa première phase utilisant une heuristique. L'algorithme $\mathsf{HYB}_X(\mathsf{k})$, $X \in \{E,M,I\}$, peut se résumer par l'Algorithme 4.3.

ALG. 4.3 – La méthode $\mathsf{HYB}_X(\mathsf{k})$

{**Entrée :** Deux génomes G_0 et G_1. Un entier k}
{**Sortie :** Une mesure entre G_0 et G_1}

1 Appel de l'heuristique IILCS_X jusqu'à ce qu'il n'existe plus de sous-chaîne commune formée de gènes non-couplés de taille au moins k
2 Résolution par le programme pseudo-booléen de l'instance intermédiaire obtenue
3 Calcul de la mesure obtenu par le couplage

L'hypothèse de départ nous suggérant cette approche est que l'heuristique IILCS_X s'écarte de la solution optimale lorsque les tailles des sous-chaînes communes sont petites. En d'autre termes, lors des premières itérations de IILCS_X, c'est-à-dire celles où les sous-chaînes communes calculées sont de grande taille, la plupart des couplages réalisés font intuitivement partie d'une solution optimale. L'intérêt de la méthode hybride est ainsi double. Premièrement, par sa première phase, nous profitons de la rapidité de l'heuristique IILCS_X et espérons faire très peu de mauvais choix dans les couplages réalisés. Deuxièmement, lorsque les sous-chaînes communes sont de petite taille (voire de taille égale à 1), nous profitons de l'efficacité qualitative de la méthode exacte. Néanmoins, l'algorithme $\mathsf{HYB}_X(\mathsf{k})$ n'est évidemment pas un algorithme exact de par sa première phase utilisant une heuristique et est exponentielle de par la résolution du programme pseudo-booléen.

4.4 Résultats expérimentaux

Dans ce chapitre, nous avons défini plusieurs approches pour résoudre les problèmes $ICOM_X$, $ICONS_X$, BD_X et ADJ_X. Dans cette section, nous présentons une étude comparative des résultats obtenus par les approches non exactes ILCS_X, IILCS_X et $\mathsf{HYB}_X(\mathsf{k})$ vis-à-vis des résultats obtenus par la méthode exacte basée sur une transformation en un problème pseudo-booléen. L'idée directrice est d'étudier les performances des différentes approches non exactes en s'appuyant sur les résultats optimaux connus, ceci afin de déterminer quelles méthodes sont les plus efficaces et quelles en sont leurs limites.

Nous commencerons tout d'abord par présenter le jeu de données utilisé. Puis, nous étudierons les résultats obtenus pour la mesure *nombre d'intervalles communs* et nous analyserons enfin les résultats pour les mesures *nombre de points de cassure* et *nombre d'adjacences*.

4.4.1 Données

Afin d'étudier les performances des différents algorithmes proposés précédemment, nous avons utilisé un jeu de données réel composé de douze génomes complets de γ-Protéobactéries. Ce jeu de don-

nées a été étudié initialement dans [LDM03] et repris dans l'étude de Blin et al. [BCF05] dans laquelle l'heuristique ILCS_M a été étudiée. Nous listons ci-dessous les génomes considérés avec leur nom usuel, ainsi que leur identifiant NCBI :

- *Buchnera aphidicola APS* (Baphi, Identifiant 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),
- *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),
- *Yersinia pestis KIM5 P12* (Ypest-KIM, NC_004088) et
- *Wigglesworthia glossinidia brevipalpis* (Baphi NC_004344).

La Table 4.1 présente les différentes caractéristiques des génomes considérés. Ces génomes sont de tailles assez variables, et possèdent en moyenne 3104 gènes, allant de 564 gènes pour Baphi à 5540 gènes pour Paeru, alors que le nombre de familles de gènes oscille entre 549 et 4500 (toujours pour Baphi et Paeru). Enfin, notons que seulement $2{,}55\%$ des gènes de Baphi sont dupliqués alors que $11{,}98\%$ des gènes de Salty sont des gènes dupliqués. Ce jeux de données possèdent donc des paramètres assez variables, en termes de taille de génomes et de nombre de gènes dupliqués.

4.4.2 Intervalles communs : analyse des méthodes proposées

Nous présentons dans cette partie les résultats obtenus pour le problème $ICOM_M$ (défini en Section 2.3). Ces résultats ont été publiés dans [AFRV06] et étendus dans [AFRV07]. Nous présentons tout d'abord les résultats exacts donnés par le programme $\mathsf{LPB-ICOM}_M$ (voir Section 4.2.2). Ces tests ont été effectués sous Linux sur un Intel(R) Pentium(R) Duo de CPU $2{,}80$ GHz et avec 1Gb de mémoire.

Parmi les soixante-six paires de génomes à comparer, nous avons obtenu quarante résultats exacts, soit un peu moins de deux tiers des instances (voir Table 4.2). Le temps d'exécution accordé au programme $\mathsf{LPB-ICOM}_M$ a été d'une semaine. Notons tout de même que, parmi les quarante instances pour lesquelles nous avons obtenu une réponse, seules quatre d'entre elles ont nécessité plusieurs heures, voire plusieurs jours, (Haein/Pmult, Haein/Xfast, Paeru/Pmult et Paeru/XFast), alors que les autres instances ont nécessité au plus quelques minutes (voir Table 4.7, page 94). Ces résultats montrent la limite de l'approche exacte pour la mesure *nombre d'intervalles communs*, de par le manque de résultats obtenus en un temps acceptable.

L'intérêt principal de notre approche pseudo-booléenne est d'obtenir le maximum de résultats exacts, ceci afin d'apprécier les performances des différentes heuristiques. Nous avons ainsi confronté les quarante résultats exacts à ceux obtenus par les méthodes ILCS_M, IILCS_M et $\mathsf{HYB}_M(2)$. Rappelons que les heuristiques ILCS_M, IILCS_M peuvent être itérées plusieurs fois tout en gardant des temps d'exécution raisonnables. Dans notre étude, nous avons paramétré le nombre d'itérations à 10 pour chacune de ces deux heuristiques.

Table 4.1 – Caractéristiques des douze génomes de γ-Protéobactéries étudiés

Génome	taille du génome	nombre de gènes	pourcentage de gènes dupliqués
Baphi	564	549	2,55
Ecoli	4183	3423	11,92
Haein	1709	1531	7,12
Paeru	5540	4500	11,04
Pmult	2015	1811	6,29
Salty	4203	3456	11,98
Wglos	653	627	3,19
Xaxon	4192	3634	2,83
Xcamp	4029	3468	8,56
Xfast	2680	2346	8,48
Ypest-CO92	3599	3021	9,07
Ypest-KIM	3879	3236	9,21

	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92	Ypest-KIM
Baphi	2882	1109	1524	1224	2849	1275	1186	1186	979	2585	2141
Ecoli		2784		3342		2328			1877		
Haein			2036	3936	2820	1085	1474	1461	1295	2694	2500
Paeru				2337		1558			2377		
Pmult					3376	1214				3298	3092
Salty						2335			1981		
Wglos							1228	1226	994	2318	2093
Xaxon											
Xcamp											
Xfast										1949	1891
Ypest-CO92											

Table 4.2 – Nombres d'intervalles communs obtenus par $\mathsf{LPB}-\mathsf{ICOM}_M$

Nous nous focalisons maintenant sur les trois génomes pour lesquels nous avons obtenu tous les résultats exacts : les résultats pour *Buchnera aphidicola* (Baphi) sont présentés dans la Table 4.3, ceux de *Haemophilus influenzae* (Haein) dans la Table 4.4, et enfin ceux de concernant *Wigglesworthia glossini-*

Baphi vs				
	ILCS_M	IILCS_M	$\mathsf{HYB}_M(2)$	$\mathsf{LPB{-}ICOM}_M$
Ecoli	2605	2869	**2882**	**2882**
Haein	1103	**1109**	**1109**	**1109**
Paeru	1492	1518	**1524**	**1524**
Pmult	1216	**1224**	**1224**	**1224**
Salty	2641	**2849**	**2849**	**2849**
Wglos	1261	1267	**1275**	**1275**
Xaxon	1127	**1186**	**1186**	**1186**
Xcamp	1147	**1186**	**1186**	**1186**
Xfast	975	**979**	**979**	**979**
Ypest-CO92	2387	2541	**2585**	**2585**
Ypest-KIM	1965	2124	**2141**	**2141**

Table 4.3 – Comparaison du nombre d'intervalles communs obtenu par les différentes méthodes de résolution pour le génome *Buchnera aphidicola* (Baphi). Les résultats optimaux sont notés en gras.

Haein vs				
	ILCS_M	IILCS_M	$\mathsf{HYB}_M(2)$	$\mathsf{LPB{-}ICOM}_M$
Baphi	1103	**1109**	**1109**	**1109**
Ecoli	2758	2774	2783	**2784**
Paeru	1980	2016	2026	**2036**
Pmult	3901	3910	3911	**3936**
Salty	2794	2809	2819	**2820**
Wglos	1077	**1085**	**1085**	**1085**
Xaxon	1446	1468	1469	**1474**
Xcamp	1441	1455	**1461**	**1461**
Xfast	1285	1293	**1295**	**1295**
Ypest-CO92	2668	2686	**2694**	**2694**
Ypest-KIM	2458	2487	2499	**2500**

Table 4.4 – Comparaison du nombre d'intervalles communs obtenu par les différentes méthodes de résolution pour le génome *Haemophilus influenzae* (HAEIN). Les résultats optimaux sont notés en gras.

Wglos vs				
	ILCS_M	IILCS_M	$\mathsf{HYB}_M(2)$	$\mathsf{LPB{-}ICOM}_M$
Baphi	1261	1267	**1275**	**1275**
Ecoli	2102	2184	**2328**	**2328**
Haein	1077	**1085**	**1085**	**1085**
Paeru	1496	1506	1552	**1558**
Pmult	1204	1211	**1214**	**1214**
Salty	2083	2274	2331	**2335**
Xaxon	1120	1198	**1228**	**1228**
Xcamp	1151	1196	**1226**	**1226**
Xfast	963	973	**994**	**994**
Ypest-CO92	2037	2301	**2318**	**2318**
Ypest-KIM	1833	2086	**2093**	**2093**

Table 4.5 – Comparaison du nombre d'intervalles communs obtenu par les différentes méthodes de résolution pour le génome *Wigglesworthia glossinidia brevipalpis* (WGLOS). Les résultats optimaux sont notés en gras.

dia brevipalpis (Wglos) dans la Table 4.5. Au vu de ces résultats, nous pouvons voir, sans surprise, que l'algorithme $\mathsf{HYB}_M(2)$ est toujours au moins aussi performant que l'heuristique IILCS_M, elle même

toujours meilleure ou égale à l'heuristique ILCS_M. En particulier, $\mathsf{HYB}_M(2)$ retourne la valeur optimale dans tous les cas pour *Buchnera aphidicola*, dans 5 cas sur 11 pour *Haemophilus influenzae*, et dans 9 cas sur 11 pour *Wigglesworthia glossinidia brevipalpis*.

La Table 4.6 présente une synthèse des performances des trois méthodes approchées, en précisant, pour chacune des quarante instances pour lesquelles nous avons la solution exacte, la qualité de la solution par le ratio $\frac{IC(\mathcal{H})}{IC(OPT)}$ donné sous forme de pourcentage, où $IC(\mathcal{H})$ est le nombre d'intervalles communs trouvé par l'heuristique $\mathcal{H}$ et où $IC(OPT)$ est la solution optimale. La Figure 4.7 illustre graphiquement ces résultats. Ces résultats confirment que $\mathsf{HYB}_M(2)$ est toujours meilleur (ou aussi bon) que IILCS_M, et que IILCS_M est toujours meilleure (ou aussi bonne) que ILCS_M : plus précisément, en moyenne, IILCS_M améliore de $2{,}66\%$ les performances de ILCS_M, alors que $\mathsf{HYB}_M(2)$ améliore de $0{,}98\%$ les performances de IILCS_M.

Génome 1	**Génome 2**	ILCS_M	IILCS_M	$\mathsf{HYB}_M(2)$	**Génome 1**	**Génome 2**	ILCS_M	IILCS_M	$\mathsf{HYB}_M(2)$
Baphi	Ecoli	90.39	99.55	**100**	Haein	Xcamp	98.83	99.79	**100**
Baphi	Haein	99.46	**100**	**100**	Haein	Xfast	99.23	99.85	**100**
Baphi	Paeru	97.9	99.61	**100**	Haein	Ypest-CO92	99.03	99.7	**100**
Baphi	Pmult	99.35	**100**	**100**	Haein	Ypest-KIM	98.32	99.48	99.96
Baphi	Salty	92.7	**100**	**100**	Paeru	Pmult	97.48	99.32	99.74
Baphi	Wglos	98.9	99.37	**100**	Paeru	Wglos	96.02	96.66	99.61
Baphi	Xaxon	95.03	**100**	**100**	Paeru	Xfast	98.11	99.5	99.71
Baphi	Xcamp	96.71	**100**	**100**	Pmult	Salty	98.16	99.23	99.56
Baphi	Xfast	99.59	**100**	**100**	Pmult	Wglos	99.18	99.75	**100**
Baphi	Ypest-CO92	92.34	98.3	**100**	Pmult	Ypest-CO92	98.15	99.03	99.42
Baphi	Ypest-KIM	91.78	99.21	**100**	Pmult	Ypest-KIM	97.83	98.19	99.22
Ecoli	Haein	99.07	99.64	99.96	Salty	Wglos	89.21	97.39	99.83
Ecoli	Pmult	98.65	99.61	99.97	Salty	Xfast	98.74	99.44	99.9
Ecoli	Wglos	90.29	93.81	**100**	Wglos	Xaxon	91.21	97.56	**100**
Ecoli	Xfast	97.39	98.4	99.79	Wglos	Xcamp	93.88	97.55	**100**
Haein	Paeru	97.25	99.02	99.51	Wglos	Xfast	96.88	97.89	**100**
Haein	Pmult	99.11	99.34	99.36	Wglos	Ypest-CO92	87.88	99.27	**100**
Haein	Salty	99.08	99.61	99.96	Wglos	Ypest-KIM	87.58	99.67	**100**
Haein	Wglos	99.26	**100**	**100**	Xfast	Ypest-CO92	92.2	93.18	**100**
Haein	Xaxon	98.1	99.59	99.66	Xfast	Ypest-KIM	98.04	98.31	99.89
Méthode				ILCS_M			IILCS_M		$\mathsf{HYB}_M(2)$
Performance moyenne				**96.21**			**98.9**		**99.88**

Table 4.6 – Performances des trois approches non-exactes ILCS_M, IILCS_M et $\mathsf{HYB}_M(2)$ par rapport aux résultats exacts connus (40 instances) (mesure *nombre d'intervalles communs*, modèle *couplage maximal*).

Les deux approches $\mathsf{HYB}_M(2)$ et IILCS_M donnent d'excellents résultats sur ce jeu de données. En effet, parmi les quarante instances pour lesquelles nous connaissons la solution optimale, IILCS_M renvoie cette solution optimale dans sept cas, et retourne un nombre d'intervalles communs supérieur à 99% de l'optimum dans 22 autres cas. Le pire résultat obtenu par IILCS_M correspond à 93.18% de l'optimum (Xfast/Ypest-CO92). En moyenne, sur les quarante paires de génomes, IILCS_M donne un nombre d'intervalles communs correspondant à $98{,}9\%$ de la solution optimale.
Concernant l'approche $\mathsf{HYB}_M(2)$, les résultats sont encore une fois de meilleure qualité. Parmi les quarante instances où l'optimum est connu, l'algorithme retourne l'optimum dans 23 cas, et obtient dans les 17 autres cas restants au moins 99% de l'optimum. Le pire cas obtenu par $\mathsf{HYB}_M(2)$ donne un nombre d'intervalles communs correspondant à $99{,}22\%$ de l'optimum (Pmult/Ypest-KIM). Enfin, en moyenne,

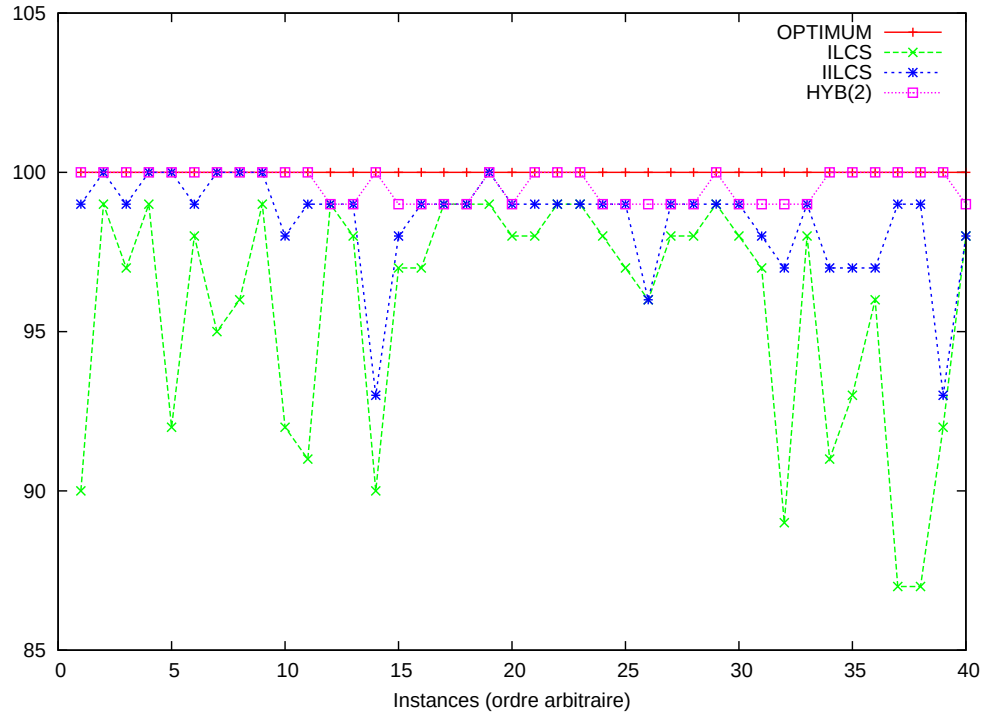


Figure 4.7 – Comparaison des résultats obtenus par $\mathsf{LPB{-}ICOM}_M$ (courbe OPTIMUM à 100%), ILCS_M, IILCS_M et $\mathsf{HYB}_M(2)$ pour les quarante instances pour lesquelles la solution optimale est connue.

$\mathsf{HYB}_M(2)$ obtient $99{,}88\%$ de la solution optimale.

Notre étude montre sans ambiguïté que, sur ce jeu de données, l'heuristique IILCS_M donne d'excellents résultats, d'un point de vue qualitatif et de rapidité d'exécution. L'idée simple, consistant à calculer itérativement une plus longue sous-chaîne commune afin de mettre en correspondance les gènes qui la composent, s'est révélée être très efficace pour la mesure *nombre d'intervalles communs*. De plus, $\mathsf{HYB}_M(2)$ améliore ses résultats. Néanmoins, le temps d'exécution de $\mathsf{HYB}_M(2)$ reste exponentiel, engendrant ainsi moins de résultats en un temps acceptable (23 résultats manquants, cf. Table 4.7). En définitive, cette analyse suggère que l'heuristique IILCS_M se révèle être la meilleure des méthodes proposées pour compter le nombre d'intervalles communs entre deux génomes, cumulant à la fois des temps d'exécution très rapides et une qualité des résultats élevés (même s'il existe des cas où elle se comporte mal).

Cette étude s'est portée uniquement sur le modèle *couplage maximal* pour lequel nous avons obtenu un bon nombre de résultats optimaux par la méthode pseudo-booléenne. En revanche, pour les modèles *couplage exemplaire* et *couplage intermédiaire*, la plupart des programmes pseudo-booléens générés se sont révélés trop ardus pour être résolus par `minisat+`, même si la taille des programmes était du même ordre de grandeur que celles obtenues par le modèle *couplage maximal*. Nous pensons que la complexité provient en grande partie de la structure même du programme généré, mal adaptée au solveur `minisat+`. Par divers essais, nous nous sommes aperçus que l'ajout de redondance dans les contraintes (i.e. certaines contraintes booléennes peuvent être écrites de plusieurs manières) ou bien un changement de l'ordre des contraintes peut modifier considérablement les temps de calculs, en bien ou en mal. Il

serait intéressant d'étudier dans le futur ces modifications afin de mieux comprendre quels types de programmes sont les mieux adaptés à `minisat+` ainsi qu'aux autres solveurs existants.

Nous portons maintenant notre étude aux mesures *nombre de points de cassure* et *nombre d'adjacences*.

4.4.3 Points de cassure et adjacences : analyse des méthodes proposées

Nous nous intéressons dans cette section aux problèmes BD_X et ADJ_X. Nous avons prouvé auparavant que les problèmes BD_X et ADJ_X sont **APX**-Difficiles (Théorèmes 2 et 4, pages 41 et 44). Malgré cette complexité forte, il peut être important d'établir des méthodes exactes de résolution, notamment afin d'évaluer les méthodes approchées. Dans cette optique, nous avons réalisé une étude comparée des résultats obtenus par les programmes dédiés à ces problèmes, à savoir les programmes exacts $\mathsf{LPB-ADJ}_X$ pour $X \in \{E,M,I\}$ et $\mathsf{LPB-BD}_I$ (cf. Section 4.2.6) et les méthodes approchées IILCS_X et $\mathsf{HYB}_X(\mathsf{k})$ (Section 4.3). Nous présentons tout d'abord les résultats exacts obtenus pour chacun des trois modèles de couplage : *couplage exemplaire*, *couplage maximal* et *couplage intermédiaire*. Puis, nous présentons les résultats obtenus par les différentes heuristiques et analysons ensuite leur performances. Cette étude a été publiée dans [AFR+07], puis étendue dans [AFR+08b].

Résultats de l'approche pseudo-booléenne. Nous présentons ici les résultats obtenus par les programmes pseudo-booléens suivants présentés en Section 4.2.6 :

- $\mathsf{LPB-ADJ}_E$ résolvant BD_E et ADJ_E
- $\mathsf{LPB-ADJ}_M$ résolvant BD_M et ADJ_M
- $\mathsf{LPB-ADJ}_I$ résolvant ADJ_I
- $\mathsf{LPB-BD}_I$ résolvant BD_I

Nous avons pu, de par notre collaboration avec Annelyse Thévenin et Stéphane Vialette, utiliser le solveur CPLEX† sur une machine spécifiquement dédiée. Tous nos tests ont été réalisés sous Linux sur un Quadri Intel(R) Xeon(TM) de CPU 3 GHz et avec 16Gb de mémoire. Bien que CPLEX ne soit pas spécifiquement dédié aux problèmes pseudo-booléens mais soit plus générique, celui-ci s'est révélé plus rapide que le solveur `minisat+` pour les problèmes BD_X et ADJ_X. En revanche, pour d'autres problèmes étudiés récemment mais non exposés dans ce mémoire, `minisat+` reste bien meilleur que CPLEX. Il nous semble très difficile de comprendre les raisons de l'efficacité des différents solveurs. Afin de choisir a priori le solveur adéquat, il faudrait, pour chaque outil de résolution, connaître les caractéristiques des instances difficiles à traiter. Nous présentons dans ce mémoire les résultats obtenus par CPLEX qui sont exposés dans la Table 4.8 en séparant les résultats selon le modèle de couplage utilisé.

Modèle *couplage maximal*. Parmi les problèmes étudiés, les problèmes BD_M et ADJ_M se sont révélés être les plus faciles à résoudre sur notre jeu de données de γ-Protéobactéries (i.e. les plus rapides en temps d'exécution). En effet, **toutes** les instances ont été résolues par le solveur CPLEX et le programme pseudo-booléen $\mathsf{LPB-ADJ}_M$ (voir Table 4.8), ceci en moins de deux minutes (1.7 secondes en moyenne par comparaison).

Modèle *couplage exemplaire*. À l'opposé du modèle *couplage maximal*, nous n'avons pu résoudre par le programme $\mathsf{LPB-ADJ}_E$ toutes les instances pour les problèmes BD_E et ADJ_E. Plus précisément,

† \texttt{http://www.ilog.com/products/cplex/}

		Nombre d'intervalles communs				Temps d'exécution (en secondes)			
G_0	G_1	LPB−ICOM$_M$	ILCS$_M$	IILCS$_M$	HYB$_M$(2)	LPB−ICOM$_M$	ILCS$_M$	IILCS$_M$	HYB$_M$(2)
BAPHI	ECOLI	2882	2605	2869	2881	18	5	0	0
BAPHI	HAEIN	1109	1103	1109	1109	1	3	0	0
BAPHI	PAERU	1524	1492	1518	1524	3	7	0	0
BAPHI	PMULT	1224	1216	1224	1224	1	3	0	1
BAPHI	SALTY	2849	2641	2849	2849	6	5	1	0
BAPHI	WGLOS	1275	1261	1267	1275	0	1	0	1
BAPHI	XAXON	1186	1127	1186	1186	1	6	0	0
BAPHI	XCAMP	1186	1147	1186	1186	1	6	0	0
BAPHI	XFAST	979	975	979	979	0	4	0	0
BAPHI	YPESTCO92	2585	2387	2541	2585	5	4	0	0
BAPHI	YPESTKIM	2141	1965	2124	2141	6	5	0	0
ECOLI	HAEIN	2784	2758	2774	2783	71	77	3	1
ECOLI	PAERU		3821	3973			378	14	
ECOLI	PMULT	3342	3297	3329	3340	87	95	4	1
ECOLI	SALTY		65634	66025			84	34	
ECOLI	WGLOS	2328	2102	2184	2311	14	16	0	1
ECOLI	XAXON		2485	2537			242	6	
ECOLI	XCAMP		2459	2524			234	5	
ECOLI	XFAST	1877	1828	1847	1872	63	111	2	3
ECOLI	YPESTCO92		13786	14355			144	5	
ECOLI	YPESTKIM		13762	14121			169	6	
HAEIN	PAERU	2036	1980	2016	2023	18	77	2	2
HAEIN	PMULT	3936	3901	3887	3934	352383	17	3	1
HAEIN	SALTY	2820	2794	2809	2819	74	53	3	1
HAEIN	WGLOS	1085	1077	1085	1085	1	5	0	0
HAEIN	XAXON	1474	1446	1468	1469	20	55	2	2
HAEIN	XCAMP	1461	1441	1455	1461	1153	53	1	15
HAEIN	XFAST	1295	1285	1293		513423	32	1	
HAEIN	YPESTCO92	2694	2668	2686	2692	243	44	3	2
HAEIN	YPESTKIM	2500	2458	2487	2497	155	51	3	0
PAERU	PMULT	2337	2278	2321	2330	73510	136	3	3
PAERU	SALTY		3826	3956			412	12	
PAERU	WGLOS	1558	1496	1506	1552	5	21	0	0
PAERU	XAXON		3788	3948			395	10	
PAERU	XCAMP		3746	3898			382	10	
PAERU	XFAST	2377	2332	2365	2369	102788	159	2	1
PAERU	YPESTCO92		3861	3989			323	9	
PAERU	YPESTKIM		3757	3857			364	10	
PMULT	SALTY	3376	3314	3350	3355	72	68	3	1
PMULT	WGLOS	1214	1204	1211	1214	1	6	0	0
PMULT	XAXON		1626	1641	1644		72	2	118
PMULT	XCAMP		1603	1621			69	2	
PMULT	XFAST		1453	1463	1462		41	1	4
PMULT	YPESTCO92	3298	3237	3266	3282	140	58	3	1
PMULT	YPESTKIM	3092	3025	3036	3073	129	67	4	1
SALTY	WGLOS	2335	2083	2274	2318	13	15	1	0
SALTY	XAXON		2606	2684			248	5	
SALTY	XCAMP		2595	2673			239	5	
SALTY	XFAST	1981	1956	1970	1978	86	112	2	2
SALTY	YPESTCO92		14972	15191			146	5	
SALTY	YPESTKIM		14770	15199			174	6	
WGLOS	XAXON	1228	1120	1198	1228	2	9	0	0
WGLOS	XCAMP	1226	1151	1196	1226	2	9	0	0
WGLOS	XFAST	994	963	973	994	1	5	1	1
WGLOS	YPESTCO92	2318	2037	2301	2318	6	6	0	0
WGLOS	YPESTKIM	2093	1833	2086	2093	7	7	0	1
XAXON	XCAMP		106681	108345			61	24	
XAXON	XFAST		6440	6890	6945		100	2	1
XAXON	YPESTCO92		2395	2482			193	4	
XAXON	YPESTKIM		2471	2542			211	4	
XCAMP	XFAST		6339	6732	6745		97	1	1
XCAMP	YPESTCO92		2358	2433			185	4	
XCAMP	YPESTKIM		2407	2509			202	4	
XFAST	YPESTCO92	1949	1797	1816	1947	49	86	2	2
XFAST	YPESTKIM	1891	1854	1859	1885	81	94	2	3
YPESTCO92	YPESTKIM		242963	261345			40	4	

Table 4.7 – Nombre d'intervalles communs et temps d'exécution des programmes LPB−ICOM$_M$, ILCS$_M$, IILCS$_M$ et HYB$_M$(2).

	LPB−ADJ$_E$		LPB−ADJ$_M$		LPB−BD$_I$		LPB−ADJ$_I$	
$G_0 - G_1$	adj	bkp	adj	bkp	adj	bkp	adj	bkp
BAPHI – ECOLI	368	152	377	156	372	152	378	154
BAPHI – HAEIN	157	265	161	270	158	265	162	267
BAPHI – PAERU	226	232	229	240	227	232	230	237
BAPHI – PMULT	182	254	188	259	184	254	189	256
BAPHI – SALTY	367	154	376	158	372	154	377	155
BAPHI – WGLOS	201	168	203	170	201	168	203	168
BAPHI – XAXON	183	222	188	226	184	222	188	223
BAPHI – XCAMP	183	222	188	226	183	222	188	223
BAPHI – XFAST	158	231	162	236	158	231	162	234
BAPHI – YPEST-CO92	352	166	361	170	355	166	362	167
BAPHI – YPEST-KIM	339	172	348	176	343	172	349	172
ECOLI – HAEIN	489	610	550	665	507	610	551	624
ECOLI – PAERU	570	847	651	1082	597	847	668	906
ECOLI – PMULT	593	622	662	703	612	622	670	647
ECOLI – SALTY			2875	277				
ECOLI – WGLOS	371	183	379	194	374	183	382	187
ECOLI – XAXON	380	675	425	842	390	675	437	718
ECOLI – XCAMP	378	678	420	845	386	678	432	717
ECOLI – XFAST	301	491	324	564	308	491	329	503
ECOLI – YPEST-CO92	1559	426	1744	596			1789	463
ECOLI – YPEST-KIM			1747	607			1798	464
HAEIN – PAERU	301	550	333	615	309	550	337	567
HAEIN – PMULT	755	497	849	525	794	497	849	509
HAEIN – SALTY	492	612	550	676	515	612	553	635
HAEIN – WGLOS	159	267	165	277	163	267	165	271
HAEIN – XAXON	217	473	241	533	221	473	243	482
HAEIN – XCAMP	216	473	238	530	218	473	239	485
HAEIN – XFAST	191	424	205	468	194	424	207	440
HAEIN – YPEST-CO92	465	597	522	649	481	597	523	617
HAEIN – YPEST-KIM	460	598	517	653	479	598	518	622
PAERU – PMULT	340	592	373	681	347	592	380	627
PAERU – SALTY	559	862	644	1091	585	862	658	918
PAERU – WGLOS	246	248	251	260	249	248	253	249
PAERU – XAXON	536	802	610	1016	562	802	620	863
PAERU – XCAMP	536	801	597	1012	557	801	609	847
PAERU – XFAST	372	499	406	572	389	499	410	522
PAERU – YPEST-CO92	571	790	671	990	604	790	685	853
PAERU – YPESTKIM	566	786	662	1004	598	786	681	855
PMULT – SALTY	597	622	670	704	620	622	677	650
PMULT – WGLOS	188	262	197	270	190	262	197	265
PMULT – XAXON	245	495	274	557	248	495	277	506
PMULT – XCAMP	241	495	267	555	245	495	270	507
PMULT – XFAST	213	436	234	481	221	436	238	451
PMULT – YPEST-CO92	582	597	648	671	602	597	654	621
PMULT – YPEST-KIM	574	601	639	676	588	601	644	631
SALTY – WGLOS	372	181	380	192	376	181	383	185
SALTY – XAXON	375	684	434	854	391	684	445	718
SALTY – XCAMP	375	684	427	854	389	684	440	716
SALTY – XFAST	300	497	325	569	310	497	329	509
SALTY – YPEST-CO92	1560	439	1758	591			1793	483
SALTY – YPEST-KIM			1761	606			1800	477
WGLO – SXAXON	189	261	194	269	189	261	195	266
WGLOS – XCAMP	189	260	194	268	189	260	195	266
WGLOS – XFAST	158	264	163	272	160	264	164	267
WGLOS – YPEST-CO92	369	182	377	193	373	182	380	186
WGLOS – YPEST-KIM	356	186	364	197	361	186	367	187
XAXON – XCAMP			3257	181				
XAXON – XFAST	980	375	1076	400	1026	375	1077	380
XAXON – YPEST-CO92	372	624	420	760	386	624	432	654
XAXON – YPEST-KIM	368	624	422	760	385	624	432	661
XCAMP – XFAST	969	373	1061	404	1005	373	1065	383
XCAMP – YPEST-CO92	369	620	412	755	380	620	424	644
XCAMP – YPEST-KIM	365	618	412	749	379	618	420	655
XFAST – YPEST-CO92	298	473	323	542	306	473	327	485
XFAST – YPEST-KIM	292	477	315	545	298	477	318	487
YPEST-CO92 – YPEST-KIM			3328	59				

Table 4.8 – Nombres d'adjacences (colonnes adj) et de points de cassure (colonnes bkp) obtenus par l'approche pseudo-booléenne (programmes LPB−ADJ$_E$, LPB−ADJ$_M$, LPB−BD$_I$ et LPB−ADJ$_I$).

nous avons obtenu 61 résultats en un temps acceptable, sur les 66 possibles (voir Table 4.8). Hormis deux instances ayant nécessité plusieurs heures, tous les résultats ont été obtenus en moins d'une minute. Nous notons une explosion combinatoire du même ordre que celle observée pour le calcul des intervalles communs, présenté précédemment en Section 4.4.2.

La seule observation évidente que l'on peut faire pour ces cinq cas non résolus est que ces instances correspondent à des paires de génomes relativement de grande tailles (entre 2500 et 3500 gènes dans chacun des génomes). Cependant, cette remarque n'est pas une explication suffisante puisqu'il existe des paires de génomes de grande taille pour lesquelles nous avons obtenu la solution optimale par le biais de notre programme (comme par exemple Salty/Ypest-CO92).

Modèle *couplage intermédiaire*. Contrairement aux deux modèles étudiés précédemment, pour le modèle *couplage intermédiaire*, les problèmes BD_I et ADJ_I ne sont pas équivalents (cf. Exemple 3.3.1, page 31). Nous avons, parmi les 66 comparaisons possibles, obtenu 63 résultats par l'approche pseudo-booléenne $\mathsf{LPB}-\mathsf{ADJ}_I$ pour le problème ADJ_I, ceci en 3 minutes. Pour le problème BD_I, nous avons obtenu, par $\mathsf{LPB}-\mathsf{BD}_I$, 59 résultats en moins de 2 minutes.

Synthèse des résultats exacts. Pour chacun des trois modèles (*couplage exemplaire*, *couplage maximal* et *couplage intermédiaire*), nous avons obtenu la plupart des résultats exacts (plus de 95% des instances ont été résolues). Contrairement au calcul des intervalles communs, le programme pseudo-booléen supporte bien la combinatoire engendrée par les instances de notre jeu de données. Nous pouvons maintenant étudier les différentes performances des approches non exactes.

Résultats des approches non exactes. Nous présentons dans cette partie les résultats obtenus par les différentes méthodes non exactes proposées pour les problèmes ADJ_X, à savoir l'heuristique IILCS_X et l'algorithme $\mathsf{HYB}_X(\mathsf{k})$ pour une valeur de k égale à 2 et 3. Tout comme l'étude précédente portant sur la mesure *nombre d'intervalles communs*, nous avons choisi de paramétrer le nombre d'itérations de l'heuristique IILCS_X à dix essais. Nous donnons par la suite leurs performances vis-à-vis des résultats exacts obtenus par les programmes pseudo-booléeens.

	ADJ_E			ADJ_I			ADJ_M		
Heuristique	IILCS_E	$\mathsf{HYB}_E(2)$	$\mathsf{HYB}_E(3)$	IILCS_I	$\mathsf{HYB}_I(2)$	$\mathsf{HYB}_I(3)$	IILCS_M	$\mathsf{HYB}_M(2)$	$\mathsf{HYB}_M(3)$
Moyenne	99,35%	99,97%	99,99%	90,56%	99,48%	99,82%	99,00%	99,89%	99,97%
pire des cas	98,07%	99,73%	99,73%	82,09%	98,09%	98,78%	97,46%	99,38%	99,53%
meilleur cas	100%	100%	100%	98,52%	100%	100%	100%	100%	100%
Nombre d'instances pour lesquelles la solution est l'optimal	15	52 (sur 61)	59	0	17 (sur 63)	35	12	35 (sur 66)	55

Table 4.9 – Résumé de la qualité des différentes heuristiques étudiées par rapport aux solutions optimales pour les problèmes ADJ_E, ADJ_I et ADJ_M. Pour illustration, pour le modèle *couplage intermédiaire*, les résultats de l'algorithme $\mathsf{HYB}_I(2)$ sont en moyenne à $99{,}48\%$ de la solution optimale, et oscillent entre $98{,}09\%$ et 100%. De plus, pour 17 cas sur 63, $\mathsf{HYB}_I(2)$ retourne la solution optimale.

Une synthèse des résultats des méthodes non exactes est donnée en Table 4.9. Nous montrons dans cette table la qualité des solutions obtenues vis-à-vis des solutions optimales connues, en précisant, pour chaque instance, le rapport entre la solution obtenue et la solution exacte, ainsi que le meilleur et le

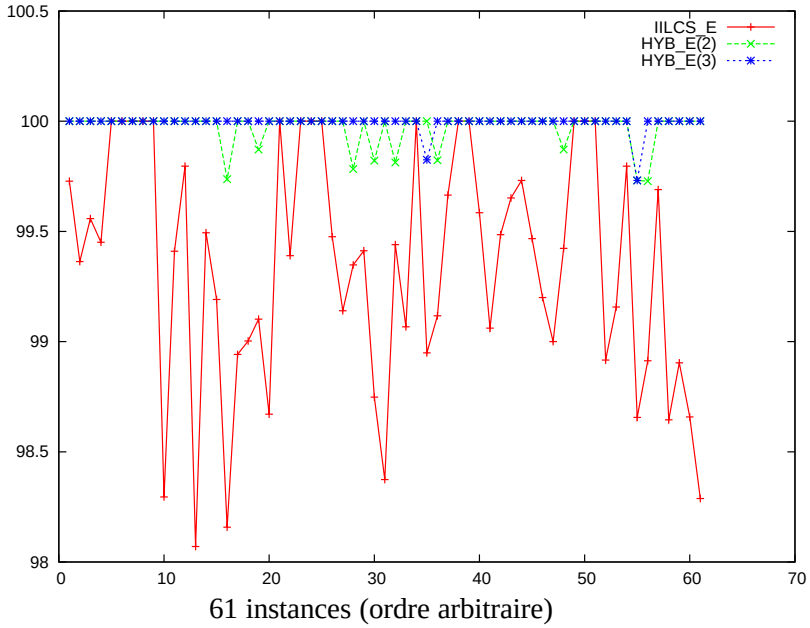


Figure 4.8 – Représentation graphique du ratio $\mathtt{adj}_E^{\mathsf{opt}}/\mathtt{adj}_E^H$ pour le problème ADJ_E, pour chacun des 61 résultats exacts obtenus. H correspond à l'heuristique considérée (*i.e.*, soit IILCS_E, $\mathsf{HYB}_E(2)$ ou $\mathsf{HYB}_E(3)$), $\mathtt{adj}_E^H$ correspond au nombre d'adjacences calculé par H et $\mathtt{adj}_E^{\mathsf{opt}}$ correspond au nombre d'adjacences d'une solution optimale.

pire des cas. L'ensemble complet des résultats est présenté dans les Tables 4.10, 4.11 et 4.12 respectivement pour les problèmes ADJ_E, ADJ_M et ADJ_I. De plus, pour une meilleure clarté des résultats, les Figures 4.8, 4.9 et 4.10 montrent, pour chacun de ces problèmes, l'écart entre les solutions obtenues et la solution optimale. Notons que, dans chacune de ces figures, les instances en abscisse sont données dans le même ordre (ordre lexicographique) que celui utilisé dans les Tables 4.10, 4.11 et 4.12.

En terme de temps d'exécution, l'heuristique IILCS_X a nécessité approximativement quarante minutes pour chacun des trois problèmes étudiés, soit un peu moins de quatre minutes pour chaque comparaison. Concernant la méthode hybride $\mathsf{HYB}_X(\mathsf{k})$, le temps moyen d'exécution est de cinq minutes pour k égal à 2 et k égal à 3.

En définitive, l'heuristique IILCS_X et l'algorithme $\mathsf{HYB}_X(\mathsf{k})$ donnent de très bons résultats sur notre jeu de données. Pour chacun des trois modèles de couplage, l'heuristique IILCS_X appropriée retourne en moyenne au moins 90% de la solution optimale. Nous pouvons voir que IILCS_I est moins efficace que IILCS_E et IILCS_M : IILCS_I obtient en moyenne $90{,}56\%$ de l'optimum (problème ADJ_I), alors que IILCS_E obtient $99{,}35\%$ (problème ADJ_E) et IILCS_M 99% (problème ADJ_M). Sans surprise, la méthode hybride, avec un paramètre k égal à 2, donne de meilleurs résultats que ceux obtenus par IILCS_X. En effet, sur l'ensemble des problèmes ADJ_X, nous obtenons en moyenne avec cette méthode $99{,}78\%$ de l'optimum. Lorsque le paramètre k est égal à 3, les performances sont encore accrues et permettent d'obtenir en moyenne $99{,}93\%$ de l'optimum.

	LPB−ADJ$_E$		IILCS$_E$		HYB$_E$(2)		HYB$_E$(3)	
$G_0 - G_1$	adj	bkp	adj	bkp	adj	bkp	adj	bkp
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			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			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 – YPESTKIM	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	375	684	373	687	375	684	375	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			1553	448	1561	440	1562	439
WGLO – SXAXON	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			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			2815	32	2815	32	2815	32

Table 4.10 – Nombre de points de cassure (colonnes bkp) et d'adjacences (colonnes adj) obtenus par LPB−ADJ$_E$ (optimal), IILCS$_E$, HYB$_E$(2) et HYB$_E$(3) pour le problème ADJ_E.

	LPB−ADJ$_M$		IILCS$_M$		HYB$_M$(2)		HYB$_M$(3)	
$G_0 - G_1$	adj	bkp	adj	bkp	adj	bkp	adj	bkp
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	188	226	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	546	669	548	667	549	666
ECOLI – PAERU	651	1082	637	1096	648	1085	650	1083
ECOLI – PMULT	662	703	660	705	662	703	662	703
ECOLI – SALTY	2875	277	2861	291	2875	277	2875	277
ECOLI – WGLOS	379	194	377	196	379	194	379	194
ECOLI – XAXON	425	842	417	850	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	1722	618	1742	598	1743	597
ECOLI – YPEST-KIM	1747	607	1724	630	1745	609	1745	609
HAEIN – PAERU	333	615	328	620	332	616	333	615
HAEIN – PMULT	849	525	845	529	848	526	849	525
HAEIN – SALTY	550	676	547	679	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	632	1103	642	1093	644	1091
PAERU – WGLOS	251	260	248	263	251	260	251	260
PAERU – XAXON	610	1016	604	1022	608	1018	610	1016
PAERU – XCAMP	597	1012	586	1023	594	1015	595	1014
PAERU – XFAST	406	572	402	576	406	572	406	572
PAERU – YPEST-CO92	671	990	655	1006	669	992	671	990
PAERU – YPESTKIM	662	1004	646	1020	660	1006	662	1004
PMULT – SALTY	670	704	668	706	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	263	559	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	630	685	637	678	637	678
SALTY – WGLOS	380	192	378	194	380	192	380	192
SALTY – XAXON	434	854	423	865	432	856	433	855
SALTY – XCAMP	427	854	419	862	425	856	425	856
SALTY – XFAST	325	569	324	570	324	570	325	569
SALTY – YPEST-CO92	1758	591	1743	606	1757	592	1758	591
SALTY – YPEST-KIM	1761	606	1737	630	1758	609	1759	608
WGLO – SXAXON	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	193	373	197	377	193	377	193
WGLOS – YPEST-KIM	364	197	361	200	364	197	364	197
XAXON – XCAMP	3257	181	3253	185	3257	181	3257	181
XAXON – XFAST	1076	400	1070	406	1074	402	1076	400
XAXON – YPEST-CO92	420	760	412	768	419	761	420	760
XAXON – YPEST-KIM	422	760	417	765	422	760	422	760
XCAMP – XFAST	1061	404	1056	409	1059	406	1061	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	315	550	321	544	323	542
XFAST – YPEST-KIM	315	545	309	551	314	546	315	545
YPEST-CO92 – YPEST-KIM	3328	59	3328	59	3328	59	3328	59

Table 4.11 – Nombre de points de cassure (colonnes bkp) et d'adjacences (colonnes adj) obtenus par LPB−ADJ$_M$ (optimal), IILCS$_M$, HYB$_M$(2) et HYB$_M$(3) pour le problème ADJ_M.

	LPB−ADJ$_I$		IILCS$_I$		HYB$_I$(2)		HYB$_I$(3)	
$G_0 - G_1$	adj	bkp	adj	bkp	adj	bkp	adj	bkp
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			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 – YPESTKIM	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
WGLO – SXAXON	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			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			2815	32	3330	56	3330	56

Table 4.12 – Nombre de points de cassure (colonnes bkp) et d'adjacences (colonnes adj) obtenus par LPB−ADJ$_I$ (optimal), IILCS$_I$, HYB$_I$(2) et HYB$_I$(3) pour le problème ADJ_I.

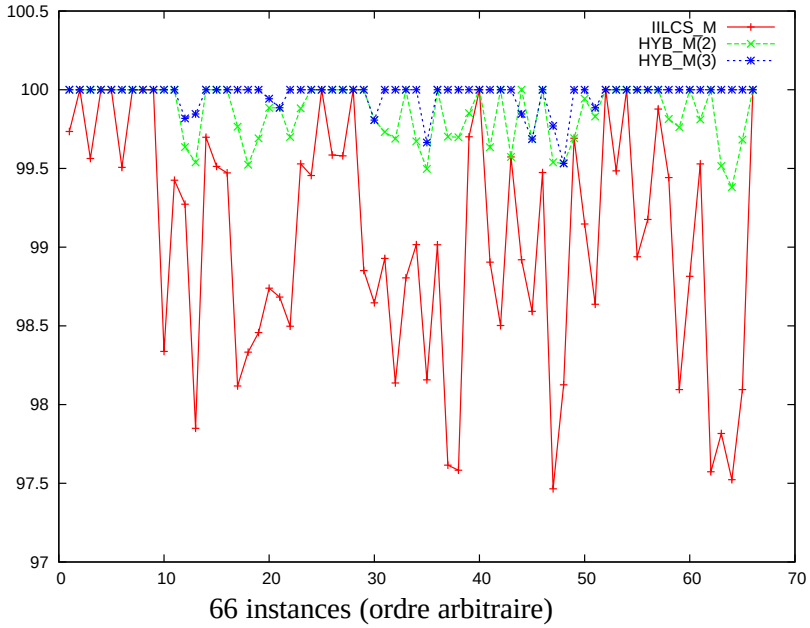


Figure 4.9 – Représentation graphique du ratio $\mathtt{adj}_M^{\mathsf{opt}}/\mathtt{adj}_M^H$ pour le problème ADJ_M, pour chacun des 66 résultats exacts obtenus. H correspond à l'heuristique considérée (*i.e.*, soit IILCS_M, $\mathsf{HYB}_M(2)$ ou $\mathsf{HYB}_M(3)$), $\mathtt{adj}_M^H$ correspond au nombre d'adjacences calculé par H et $\mathtt{adj}_M^{\mathsf{opt}}$ correspond au nombre d'adjacences d'une solution optimale.

Deux conclusions peuvent être tirées de ces résultats. Premièrement, notons que l'heuristique IILCS_X donne pour chacun des modèles d'excellents résultats, en terme de temps d'exécution et de qualité des solutions obtenues. Bien entendu, ces résultats peuvent être améliorés en utilisant l'approche hybride, d'autant plus fortement que le paramètre k est grand. Toutefois, la méthode hybride ne peut, de par sa nature, résoudre en un temps acceptable certaines instances. En particulier, cette situation peut se produire lorsque les génomes sont de très grande taille. Dans ce cas, les bonnes performances de l'heuristique IILCS_X démontrent que celle-ci peut être utilisée afin d'obtenir des résultats précis et rapides. Deuxièmement, ces résultats montrent, pour chacun des trois modèles de couplage, que l'idée intuitive consistant à rechercher itérativement une plus longue sous-séquence commune et de coupler les gènes qui la composent, est très efficace pour la mesure *nombre d'adjacences*, tout comme l'a montré l'étude réalisée pour la mesure *nombre d'intervalles communs*.

4.4.4 Conclusion

Nous avons dans ces études exécuté les différents algorithmes proposés dans ce chapitre, sur des données réelles composées de douze génomes de γ-Protéobactéries. Bien que la complexité des problèmes soit forte (**APX**-Difficulté prouvé dans le Chapitre 3), nous avons obtenu une grande majorité des résultats. En particulier, plus de 95% des comparaisons ont pu être traitées pour les mesures *nombre de points de cassure* et *nombre d'adjacences*. Grâce à ces résultats exacts, nous avons pu étudier les qualités respectives des nouvelles heuristiques proposées dans ce chapitre, à savoir l'heuristique IILCS_X et l'algorithme $\mathsf{HYB}_X(\mathsf{k})$. Une analyse détaillée des performances des différentes méthodes a montré un réel

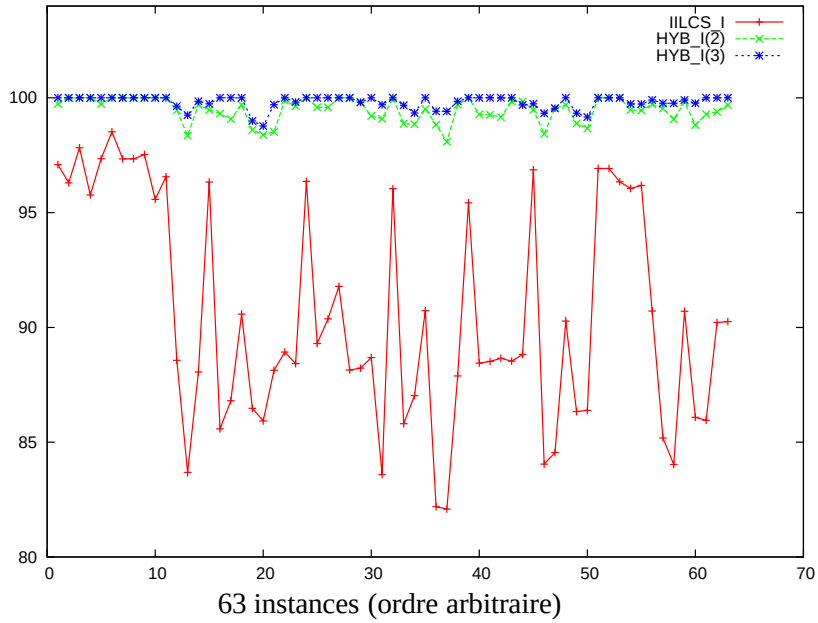


Figure 4.10 – Représentation graphique du ratio $\mathtt{adj}_I^H/\mathtt{adj}_I^{\mathsf{opt}}$ pour le problème ADJ_I, pour chacun des 63 résultats exacts obtenus. H correspond à l'heuristique considérée (*i.e.*, soit IILCS_I, $\mathsf{HYB}_I(2)$ ou $\mathsf{HYB}_I(3)$), $\mathtt{adj}_I^H$ correspond au nombre d'adjacences calculé par H et $\mathtt{adj}_I^{\mathsf{opt}}$ correspond au nombre d'adjacences d'une solution optimale.

intérêt pour l'heuristique IILCS_X. D'une part, celle-ci améliore significativement les résultats obtenus par l'heuristique ILCS_X proposée par Blin et al. dans [BCF05], et d'autre part elle obtient d'excellents résultats vis-à-vis des solutions optimales obtenues. Enfin, l'algorithme $\mathsf{HYB}_X(\mathsf{k})$ se présente comme un bon compromis afin d'améliorer la qualité des solutions, mais reste parfois trop lent pour les instances les plus difficiles.

4.5 Conclusion

Dans ce chapitre, nous avons proposé plusieurs algorithmes de résolution des problèmes BD_X, ADJ_X, $ICOM_X$ et $ICONS_X$. En particulier, nous avons défini un algorithme exact basé sur une transformation en un problème de contraintes à variables booléennes. Puis nous avons présenté deux nouvelles approches : la première (IILCS_X) est une heuristique améliorant l'heuristique ILCS_X étudiée dans [Tic84, MSM04, BCF05] ; la seconde, nommée $\mathsf{HYB}_X(\mathsf{k})$, est une méthode hybride utilisant l'heuristique précédente et la méthode exacte.

Une analyse détaillée sur des données réelles a montré la haute qualité de l'heuristique IILCS_X à la fois pour maximiser le nombre d'intervalles communs et le nombre d'adjacences. Il est étonnant que la même heuristique donne de très bons résultats pour des mesures pourtant différentes. Ce résultat peut suggérer que les notions de *nombre d'intervalles communs* et de *nombre d'adjacences* sont deux mesures finalement assez proches et que celles-ci captent la même information quant à l'ordre des gènes au sein des génomes. Sankoff et Haque ont proposé dans [SH05] les mesures nommées *nombre MAD* (pour Maximum Adjacency Disruption number) et *nombre SAD* (pour Summed Adjacency Disruption

number). Étant donnés deux génomes, les problèmes consistant à optimiser ces nombres en présence de gènes dupliqués ont été prouvés **NP**-Complets et **APX**-Difficiles [CFRV06]. Il serait intéressant de réaliser la même démarche que nous avons suivie dans notre étude afin d'établir si l'heuristique IILCS_X est aussi de bonne qualité pour ces deux mesures.

Pour finir, la méthode hybride s'est révélée de bonne qualité et améliore les résultats qualitatifs de IILCS_X, mais elle atteint néanmoins ses limites pour la maximisation du nombre d'intervalles communs. De même, la méthode pseudo-booléenne n'a pu permettre d'obtenir tous les résultats escomptés, notamment pour les intervalles communs sous les modèles *couplage exemplaire* et *couplage intermédiaire*. Une des perspectives de travail serait d'étudier les différents comportements des solveurs de programmes pseudo-booléens et d'identifier quel type de contraintes leur est le plus adapté, dans le but d'orienter l'écriture du programme et le choix du solveur vers celui le plus adéquat.

CHAPITRE 5

MATCH&WATCH, un outil de comparaison de génomes bactériens

5.1 Introduction

La recherche des ensembles de gènes fonctionnels est un problème fondamental pour comprendre les génomes bactériens. Cette recherche peut être effectuée par une comparaison de génomes basée sur la conservation des gènes. Une telle comparaison peut en effet permettre de mettre en évidence les ensembles de gènes qui sont conservés durant l'évolution. En partant de l'hypothèse que les zones conservées entre deux génomes correspondent à des fonctions biologiques, chaque ensemble de gènes mis en évidence par une comparaison de deux génomes comme étant conservé pourrait alors correspondre à une fonction spécifique. De plus, cette approche peut aussi être intéressante pour aider à l'annotation des génomes. En effet, lorsque nous comparons deux génomes par une mesure qui s'appuie sur la conservation de gènes, les ensembles de gènes conservés définissent un couplage des gènes des deux génomes. Par conséquent, les annotations connues pour l'un des génomes peuvent être transposables à l'autre génome, apportant ainsi une aide précieuse aux biologistes pour l'annotation.

Dans les chapitres précédents, nous avons étudié les problèmes d'optimisation consistant à trouver un couplage des gènes $\mathcal{M}$ de deux génomes G_0 et G_1 tel que la paire $\mathcal{M}$-*élaguée* (G'_0,G'_1) de (G_0,G_1) optimise la mesure étudiée entre (G'_0,G'_1). Plus précisément, nous nous sommes focalisés sur les mesures *nombre d'intervalles communs* (problèmes $ICOM_X$), *nombre d'intervalles conservés* (problèmes $ICONS_X$), *nombre de points de cassure* (problèmes BD_X) et sur la mesure *nombre d'adjacences* (problèmes ADJ_X). Ces différentes mesures se basent toutes sur la conservation des gènes et permettent d'identifier des zones conservées au sein des génomes. En effet, les points de cassure permettent de diviser le génome en blocs de synténie (i.e. des ensembles de gènes qui sont conservés de façon contiguë sur les deux génomes). De même, les intervalles communs ou conservés correspondent également à une zone conservée et permettent quant à eux des réarrangements génomiques en leur sein. La notion d'intervalle commun apparaît comme la plus souple de ces deux mesures, les intervalles conservés possédant certaines restrictions sur leurs extrémités. De récents travaux [LHX09] proposent un algorithme basé sur la notion d'*équipe de gènes* [BCR02] pour détecter les ensembles de gènes conservés. Les équipes de gènes étendent la définition des intervalles communs et nécessitent de fixer un paramètre correspondant au nombre maximum de gènes non conservés que l'on s'autorise entre deux gènes conservés consécutifs. Nous nous intéressons dans ce chapitre aux intervalles communs, qui apparaissent comme un bon compromis entre les points de cassure d'une part, et les équipes des gènes d'autre part. Nous proposons ainsi

un outil logiciel nommé MATCH&WATCH, qui calcule les intervalles communs entre deux génomes et facilite leur étude par le biais d'un outil de visualisation.

Ce chapitre est organisé de la manière suivante. Pour commencer, nous présentons le programme MATCH&WATCH, en spécifiant le protocole utilisé pour le calcul des intervalles communs, puis nous donnons une description de l'outil de visualisation. Par la suite, nous proposons une première application du programme réalisée sur des données réelles et analysons en détails plusieurs des intervalles communs obtenus par MATCH&WATCH.

Le travail exposé dans ce chapitre a été en partie publié dans [AEFR09] et a été réalisé en collaboration avec Damien Éveillard, Guillaume Fertin et Irena Rusu.

5.2 Description détaillée de MATCH&WATCH

Nous décrivons dans cette section le programme nommé MATCH&WATCH*. Celui-ci a pour objectifs de calculer les intervalles communs entre deux génomes à partir des données brutes (i.e. sous la forme de séquences protéiques), et d'offrir aux biologistes un outil permettant de mettre en évidence les régions conservées des deux génomes. Il est à noter que MATCH&WATCH est spécifiquement dédié aux génomes bactériens. Dans ce cadre, les génomes bactériens sont circulaires et sont donc représentés par des séquences signées circulaires. La notion d'intervalle commun doit donc s'appliquer sur des séquences circulaires (ainsi que nous l'avons présenté à la Section 4.2.5, page 72). Les méthodes IILCS_X et $\mathsf{HYB}_X(\mathsf{k})$ (voir Section 4.3) ont pu être facilement adaptées, en autorisant simplement des plus longues sous-chaînes chevauchant les extrémités (fictives) de la séquence de gènes. Nous noterons $\mathsf{IILCS}-\mathsf{CIRC}_X$ (resp. $\mathsf{HYB}-\mathsf{CIRC}_X(\mathsf{k})$) la variante de IILCS_X (resp. $\mathsf{HYB}_X(\mathsf{k})$) appliquée à des séquences circulaires. Nous souhaitons mettre en évidence les fonctions communes de deux génomes et supposons donc que chaque duplication s'est produite avant la spéciation. Ainsi, nous utiliserons le modèle *couplage maximal*, celui-ci permettant d'obtenir des régions conservées plus grandes et plus nombreuses.

Nous présentons tout d'abord le protocole suivi par MATCH&WATCH. Nous définirons ensuite quelques notations caractérisant les intervalles communs et plusieurs méthodes de filtrage permettant de réduire le nombre d'intervalles communs potentiellement intéressants. Enfin, nous décrirons l'outil de visualisation.

5.2.1 Description du protocole

Nous présentons ici chaque étape du protocole complet du programme MATCH&WATCH. La Figure 5.1 illustre les différentes étapes de la méthode que nous expliquons plus en détail ci-dessous :

Étape 1. Les entrées.
Nous prenons en entrée deux génomes G_0 et G_1 issus de la base Genbank† afin d'obtenir deux listes de gènes avec pour chacun d'eux, son identifiant et la séquence protéique associée.

*http://www.sciences.univ-nantes.fr/info/perso/permanents/angibaud/
†http://www.ncbi.nlm.nih.gov/Genbank/

Étape 2. Détection des homologies.
Nous utiliserons la méthode `InParanoid` proposée dans [RSS01]. Celle-ci calcule les homologies des gènes en quatre étapes :
– Application de Blastp [AMS+97] pour les paires de génomes (G_0,G_0), (G_0,G_1), (G_1,G_0) et (G_1,G_1)
– Choix des meilleures paires d'orthologues en gardant uniquement les paires de bon score (valeur statistique *bit score* > 50) et hautement chevauchantes, c'est-à-dire, dont le chevauchement est strictement supérieur à 50%. Chaque paire d'orthologues est alors considérée comme une famille de gènes distincte.
– Ajout des in-paralogues aux familles de gènes
– Résolution des conflits d'intersection : fusion de familles de gènes et retrait de certains in-paralogues
Pour plus de détails, se reporter à [RSS01].

Étape 3. Renommage des gènes.
Dans cette étape, nous renommons simplement les gènes selon l'étiquette de la famille à laquelle ils appartiennent. Nous obtenons ainsi deux génomes intermédiaires G'_0 et G'_1.

Étapes 4,5,6. Calcul des d'intervalles communs.
Nous utilisons ici l'une des méthodes proposées dans le Chapitre 4 afin d'obtenir un couplage $\mathcal{M}$ de (G'_0,G'_1) (Étape 4). Nous utilisons pour cela la méthode exacte basée sur la transformation en pseudo-booléen $\mathsf{LPB-ICOM-CIRC}_M$, l'heuristique $\mathsf{IILCS-CIRC}_M$ ou bien l'algorithme $\mathsf{HYB-CIRC}_M(\mathsf{k})$. Suivant la méthode utilisée, un ou deux paramètres doivent être spécifiés :
– *La méthode employée :* la méthode pseudo-booléenne $\mathsf{LPB-ICOM-CIRC}_M$, l'heuristique $\mathsf{IILCS-CIRC}_M$ ou bien la méthode hybride $\mathsf{HYB-CIRC}_M(\mathsf{k})$.
– *Le paramètre k :* la méthode $\mathsf{HYB-CIRC}_M(\mathsf{k})$ nécessite de plus le paramètre k bornant la taille des sous-chaînes communes considérées (voir Section 4.3.3, page 86).
Une fois le couplage obtenu, nous construisons la paire (G''_0,G''_1) $\mathcal{M}$-élaguée de (G'_0,G'_1) (Étape 5) et enfin, nous calculons le nombre d'intervalles communs de (G''_0,G''_1) (Étape 6).

Étape 7. Visualisation.
Cette dernière étape n'est pas nécessaire à l'obtention des intervalles communs mais permet de visualiser et de manipuler ceux-ci. Nous décrirons cet outil plus loin dans cette section.

5.2.2 Caractérisation des intervalles communs

Notre programme calcule un ensemble d'intervalles communs à deux génomes. Cet ensemble peut être filtré, par exemple en retirant des intervalles communs que l'on juge moins intéressants. Ce filtrage permet notamment de ne pas noyer l'utilisateur devant la masse d'intervalles communs qui peut être obtenue. Nous proposons tout d'abord un premier filtrage basé sur les caractéristiques intrinsèques de l'intervalle commun. Puis nous en proposerons deux autres qui s'appuient sur l'outil `GO-TermFinder` proposé dans [BWG+04], lequel calcule les annotations communes d'un ensemble de gènes. Enfin, nous définirons quelques notations caractérisant les intervalles communs.

Filtrage des intervalles communs maximaux. Quelle que soit la méthode de résolution utilisée, certains intervalles communs obtenus peuvent se révéler d'un moindre intérêt selon leurs caractéristiques.

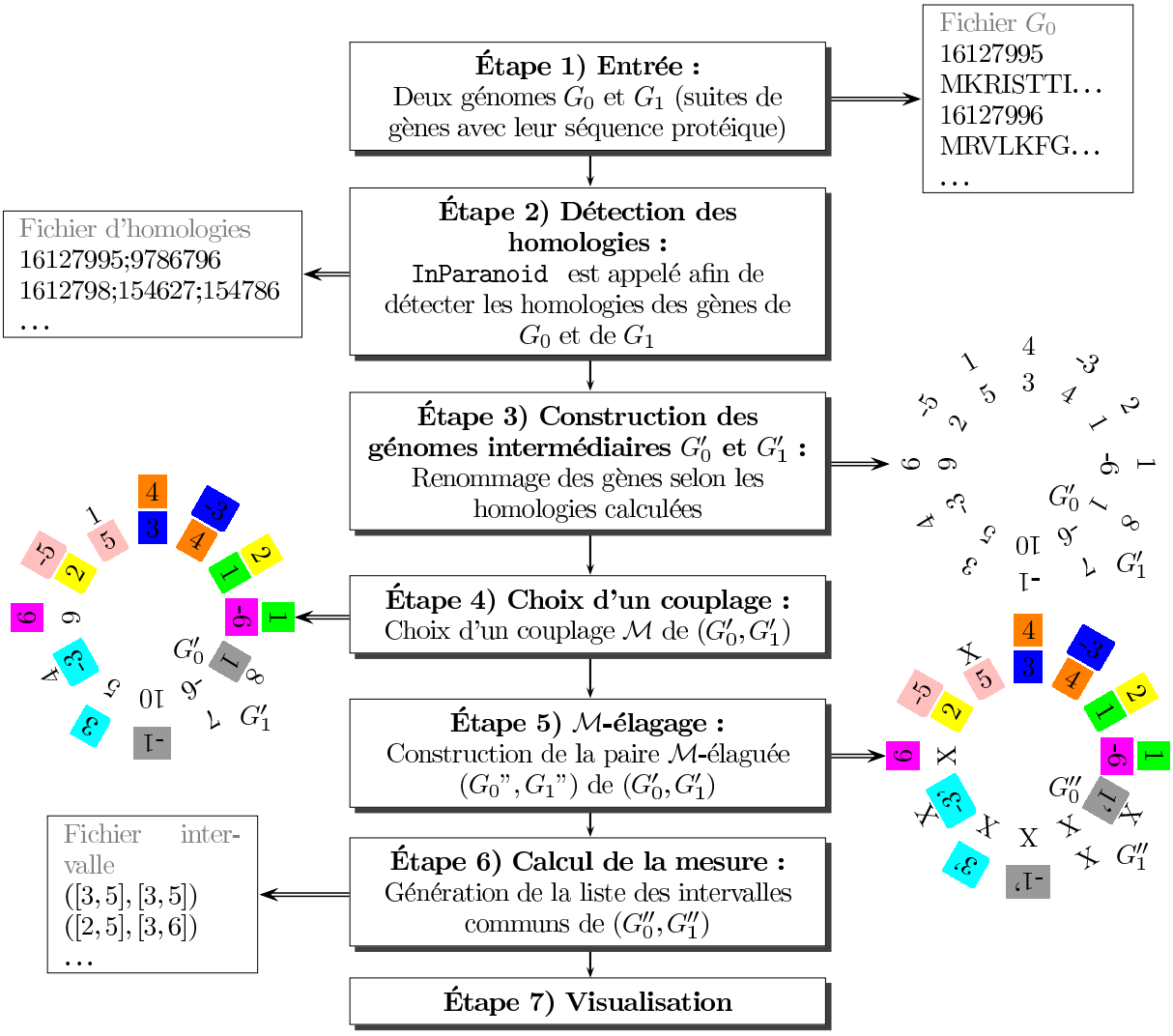


Figure 5.1 – Description des différentes étapes du protocole utilisé par le programme MATCH&WATCH.

Par exemple, lorsqu'un intervalle commun est inclus dans un intervalle commun de plus grande taille, il peut être intéressant de ne garder que les plus grands. Nous établissons la procédure suivante qui filtre les intervalles communs afin de ne garder que les intervalles communs « intéressants », que nous appellerons intervalles *maximaux*. Parmi l'ensemble des intervalles communs $\mathcal{I}$ que nous avons obtenus, nous sélectionnons les intervalles communs de la manière suivante :

1. Nous retirons de $\mathcal{I}$ les intervalles communs contenant tous les gènes couplés, le génome entier étant un intervalle commun trivial. Remarquons qu'en raison de la prise en compte de la circularité des génomes, le nombre de ces intervalles communs est égal au nombre de gènes couplés. En effet, l'intervalle commençant par un gène couplé et se terminant par le gène couplé précédent couvre tout le génome et est par conséquent un intervalle commun.
2. Remarquons, également par considération de la circularité des génomes, que pour chaque intervalle commun $I \in \mathcal{I}$, les gènes couplés n'appartenant pas à I forment également un intervalle

commun, noté I'. Par la propriété des bords (i.e. les extrémités des intervalles communs doivent être couplés), l'intervalle I' est unique. La paire (I,I') est appelé une *paire complémentaire*. Pour toute paire complémentaire $(I,I') \in \mathcal{I}^2$, nous supposons que le plus petit intervalle commun (i.e., celui qui contient le moins de gènes) est celui apportant le plus d'informations biologiques, et par conséquent, nous retirons de $\mathcal{I}$ le plus grand intervalle entre I et I'. Par exemple, si I est de taille égale à 2, l'intervalle I' couvrira tout le génome hormis les deux gènes de I. Dans ce cas, il est naturel d'estimer que seul l'intervalle I correspond à une zone conservée sur les deux génomes.

3. Nous retirons de $\mathcal{I}$ les intervalles communs qui ne contiennent qu'un seul gène, ceux-ci n'apportant aucune information biologique.
4. Enfin, soit $I \in \mathcal{I}$. Nous retirons de $\mathcal{I}$ l'intervalle commun I si I est contenu dans un intervalle commun $I' \in \mathcal{I}$. En effet, dans ce cas, l'intervalle commun I peut être inféré par la présence de I'. Le nouvel ensemble $\mathcal{I}$ obtenu correspond maintenant à l'ensemble des intervalles *maximaux*.

Filtrage des intervalles communs annotés. Afin d'obtenir les intervalles communs les plus prometteurs, nous avons utilisé le logiciel `GO-TermFinder` [BWG+04] et les annotations de la base de données *Ecocyc* [KKS+07]. L'application `GO-TermFinder` génère, pour chaque intervalle, l'ensemble des annotations communes aux gènes de l'intervalle, avec pour chacun d'eux, une *p-valeur* correspondant à la pertinence de l'annotation. Nous effectuons un premier filtrage de la manière suivante : un intervalle maximal I est appelé un intervalle *annoté* si les trois conditions suivantes sont satisfaites :

1. au moins deux gènes de I sont présents dans la base *Ecocyc*
2. au moins la moitié des gènes de I sont présents dans la base d'annotation *Ecocyc*
3. l'application `GO-TermFinder` renvoie au moins une annotation.

L'algorithme de filtrage des intervalles annotés, appelé FILTRE_A est décrit en détail par l'Algorithme 5.1 ci-après.

Par l'application du filtrage FILTRE_A, nous obtenons les intervalles communs pour lesquels l'application `GO-TermFinder` retourne une annotation cohérente. Ainsi, pour ces intervalles annotés, nous mettons en évidence des régions conservées pour lesquelles une ou plusieurs annotation(s) semble(nt) cohérente(s). Par suite, il est maintenant envisageable d'étudier en détail ces régions prometteuses et par exemple de mettre à jour les annotations manquantes sur certains gènes (notamment ceux situés sur les autres génomes que *E. coli*, pour lesquels les annotations sont beaucoup moins nombreuses).

Filtrage des intervalles communs pertinents. Dans le but d'affiner au mieux les intervalles les plus intéressants, nous avons, à partir de l'ensemble des intervalles annotés fournis par la méthode FILTRE_A, sélectionné les intervalles de meilleure *p-valeur*, appelés les intervalles *pertinents*. Pour cela, nous générons des intervalles fictifs, c'est-à-dire des ensembles aléatoires de gènes, que nous utiliserons pour étalonner la qualité des *p-valeurs* obtenues. La procédure utilisée FILTRE_P, décrite par l'Algorithme 5.2, procède de la manière suivante : nous générons tout d'abord dix jeux de données fictifs, chacun d'eux contenant autant d'intervalles fictifs que nous avons d'intervalles communs maximaux (non nécessairement pertinents). Afin de prendre en compte au mieux les tailles des intervalles, nous construisons les dix jeux d'intervalles fictifs avec la même répartition de taille que notre ensemble d'intervalles communs maximaux (non nécessairement pertinents). Une fois les intervalles fictifs construits, nous calculons la meilleure *p-valeur* p sur cet ensemble d'intervalles par `GO-TermFinder`. Enfin, nous sélectionnons

ALG. 5.1 – FILTRE$_A$: algorithme de filtrage des intervalles communs annotés

{**Entrée :** *int_communs* $= \{I_1, I_2, \ldots, I_n\}$ un ensemble de n intervalles communs}
{**Sortie :** *int_annotés* l'ensemble des intervalles annotés calculés}
{**Variable locale :** $\mathcal{T}$ un ensemble de termes (i.e. ensemble de mots)}

Pour tout $intervalle\ commun\ I_i$ **faire**
  {Test du nombre de gènes annotés dans *Ecocyc*}
  Si *nombre_de_gènes_annotés*$(I_i) \geqslant 2$ **Alors**
    Si *nombre_de_gènes_annotés*$(I_i) > \frac{1}{2} \cdot taille(I_i)$ **Alors**
      {Calcul de l'ensemble des termes communs par le logiciel `GO-TermFinder`}
      $\mathcal{T} \leftarrow$ `GO-TermFinder`(I_i)

      Si *existe_au_moins_un_terme_commun*$(\mathcal{T})$ **Alors**
        int_annotés $\leftarrow$ *int_annotés* $\cup\ I_i$
      Fin Si
    Fin Si
  Fin Si
Fin Pour

les intervalles annotés ayant une *p-valeur* inférieure à $p \cdot \epsilon$, où ϵ est un paramètre fixé. Les intervalles que nous obtenons après cette procédure sont appelés des intervalles *pertinents*.

Types d'intervalles communs. Afin de caractériser les intervalles communs maximaux, nous définissons ici les notions d'intervalle commun *droit*, *inversé* ou *non structuré*.

Définition 33. ***(Types d'intervalles communs : intervalle commun droit, inversé ou non structuré)*** *Soit un intervalle commun I de (G_0,G_1). Cet intervalle commun est dit :*

- droit *si l'ordre des gènes et les signes sont identiques dans G_0 et G_1*
- inversé *si l'ordre et les signes sont inversés dans G_1 par rapport à G_0*
- non structuré *s'il n'est ni droit ni inversé.*

5.2.3 Visualisation des intervalles communs

Nous présentons ici la partie visualisation du programme MATCH&WATCH (Étape 7). Cet outil a pour objectif de permettre à l'utilisateur :

(i) d'avoir une vue globale des intervalles communs obtenus
(ii) d'étudier spécifiquement un intervalle commun donné
(iii) d'obtenir différentes informations sur les gènes, comme par exemple les homologies ou l'appartenance à des cartes métaboliques.

L'outil se divise en cinq parties :

- *Experience view*. Celle-ci est la fenêtre principale et permet d'obtenir les informations concernant la comparaison effectuée (génomes comparés, méthode utilisée, etc.) (voir Figure 5.2).

ALG. 5.2 – FILTRE$_P$: algorithme de filtrage des intervalles communs pertinents

{**Entrée :** *int_maximaux* $= \{IM_1, IM_2, \ldots, IM_n\}$ un ensemble de n intervalles communs maximaux}
{**Entrée :** *int_annotés* $= \{IA_1, IA_2, \ldots, IA_m\}$ un ensemble de m intervalles annotés}
{**Entrée :** Le paramètre ϵ fixant le degré du filtrage}
{**Sortie :** *int_pertinents* l'ensemble des intervalles pertinents calculés}
{**Variables locales :** pour tout $1 \leqslant j \leqslant 10$, $\mathcal{F}_j$ est un ensemble d'intervalles fictifs}
{**Variable locale :** $inter$ un intervalle fictif}

{Génération des ensembles d'intervalles fictifs}
Pour tout j *de* 1 *à* 10 **faire**
  {Initialisation du nouvel ensemble d'intervalles fictifs}
  $\mathcal{F}_j \leftarrow \emptyset$

  Pour tout $intervalle\ commun\ maximal\ IM_i$ **faire**
    {Création d'un intervalle fictif de même taille que IM_i}
    $inter \leftarrow$ *générer_intervalle_aléatoirement*$(taille(IM_i))$
    $\mathcal{F}_j \leftarrow \mathcal{F}_j \cup inter$
  Fin Pour
Fin Pour

{Calcul des meilleures p-valeurs parmi tous les intervalles fictifs}
$p \leftarrow meilleure\_p\text{-}valeur(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{10})$

{Sélection des intervalles pertinents}
Pour tout *intervalle annoté* e_i **faire**
  Si $p\text{-}value(IA_i) < p \cdot \epsilon$ **Alors**
    int_pertinents $\leftarrow$ *int_pertinents* $\cup\ IA_i$
  Fin Si
Fin Pour

- *Genomes view*. Cette fenêtre affiche les deux génomes circulaires avec les intervalles communs obtenus (voir Figure 5.3).
- *Interval view*. Cette fenêtre gère l'ensemble des intervalles communs. L'utilisateur peut ici sélectionner un intervalle commun spécifique afin d'en étudier ses caractéristiques (voir Figure 5.4).
- *Gene view*. Nous affichons ici les informations concernant le dernier gène sélectionné, notamment ses homologies détectées et son appartenance à diverses cartes métaboliques (voir Figure 5.2).
- *Pathways view*. Cette fenêtre affiche la dernière carte métabolique sélectionnée, en s'appuyant sur la base kegg (http://www.genome.jp/kegg/) (voir Figure 5.5). Par cet outil, nous pouvons analyser facilement l'appartenance des gènes à un même réseau métabolique, ce qui peut aider à identifier la fonction biologique d'un intervalle commun.

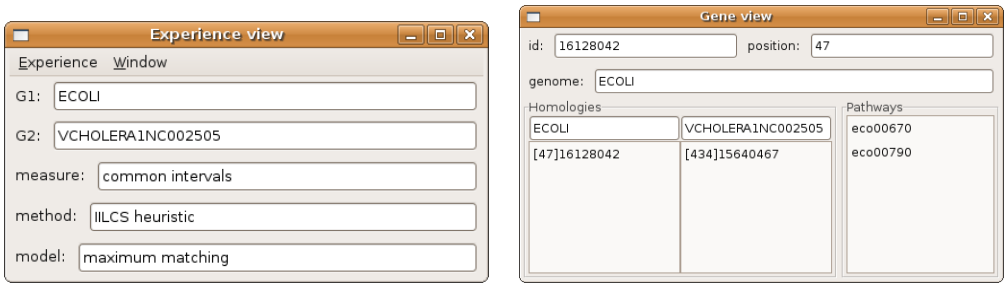


Figure 5.2 – Fenêtres *Experience view* et *Gene view* montrant les informations liées à l'expérience courante et au gène sélectionné.

5.3 Application du programme MATCH&WATCH à quatre génomes bactériens

Nous avons proposé dans la section précédente le programme MATCH&WATCH permettant de mettre en évidence des régions conservées de deux génomes bactériens. Cet outil se base sur la comparaison de deux génomes et calcule un ensemble d'intervalles communs. Dans cette section, nous présentons une première étude visant à montrer l'intérêt des intervalles communs dans la recherche des régions conservées, notamment dans le but d'aider à l'annotation de génomes et à l'identification de nouvelles régions conservées correspondant à des fonctions biologiques spécifiques.

Nous décrivons tout d'abord dans cette étude le jeu de données utilisé. Puis, nous présenterons les intervalles communs obtenus pour chaque comparaison et analyserons ensuite plusieurs cas concrets. Enfin, nous mettrons en évidence les intervalles communs nous semblant les plus prometteurs par les filtrages FILTRE$_A$ et FILTRE$_P$.

5.3.1 Données

Nous avons appliqué le programme MATCH&WATCH sur un jeu de données composé de quatre génomes de protéobactéries, à savoir *Escherichia coli*, *Vibrio fischeri* et deux souches de *Vibrio cholerae*, les génomes *Vibrio fischeri* et *Vibrio cholerae* comprenant chacun deux chromosomes. La Table 5.1 présente les références de ces données en indiquant leur identifiant NCBI. Dans cette étude, nous avons

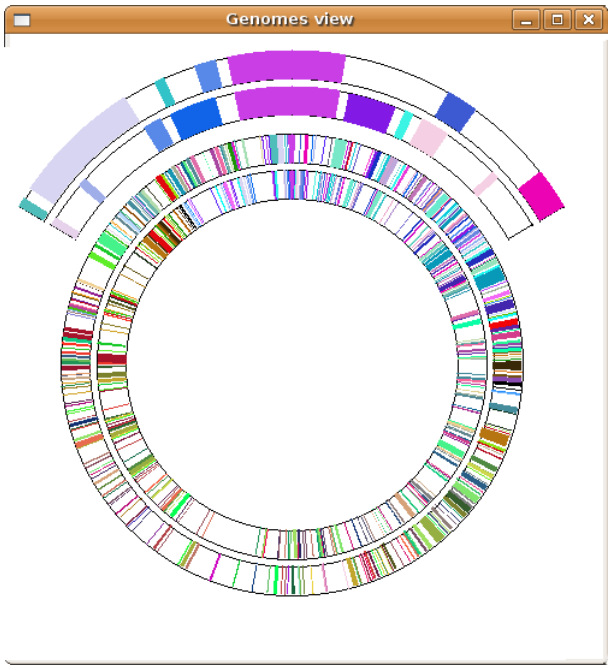


Figure 5.3 – Fenêtre *Genomes view* montrant la répartition des intervalles communs sur les deux génomes.

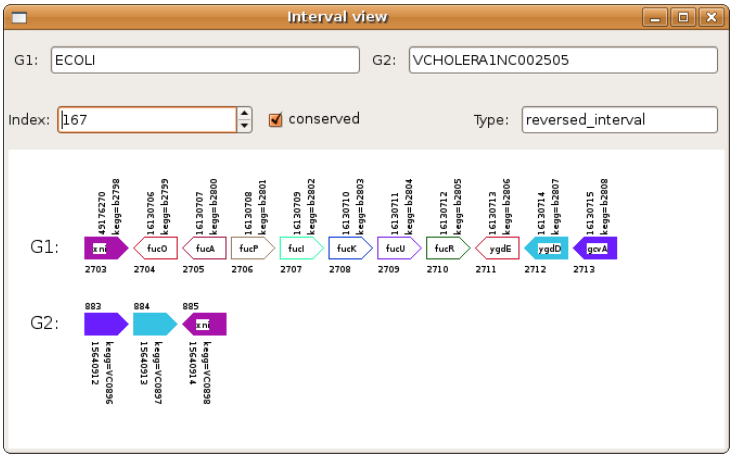


Figure 5.4 – Fenêtre *Interval view* montrant un intervalle commun inversé. Notons que les gènes sans couleur de fond sont ceux pour lesquels aucune homologie n'a été trouvée sur le second génome. Ces gènes sans homologies ne sont pas pris en compte lors du calcul des intervalles communs.

comparé *E. coli* avec chacun des six autres chromosomes afin de bénéficier des nombreuses études déjà réalisées sur *E. coli*, notamment en terme d'annotation.

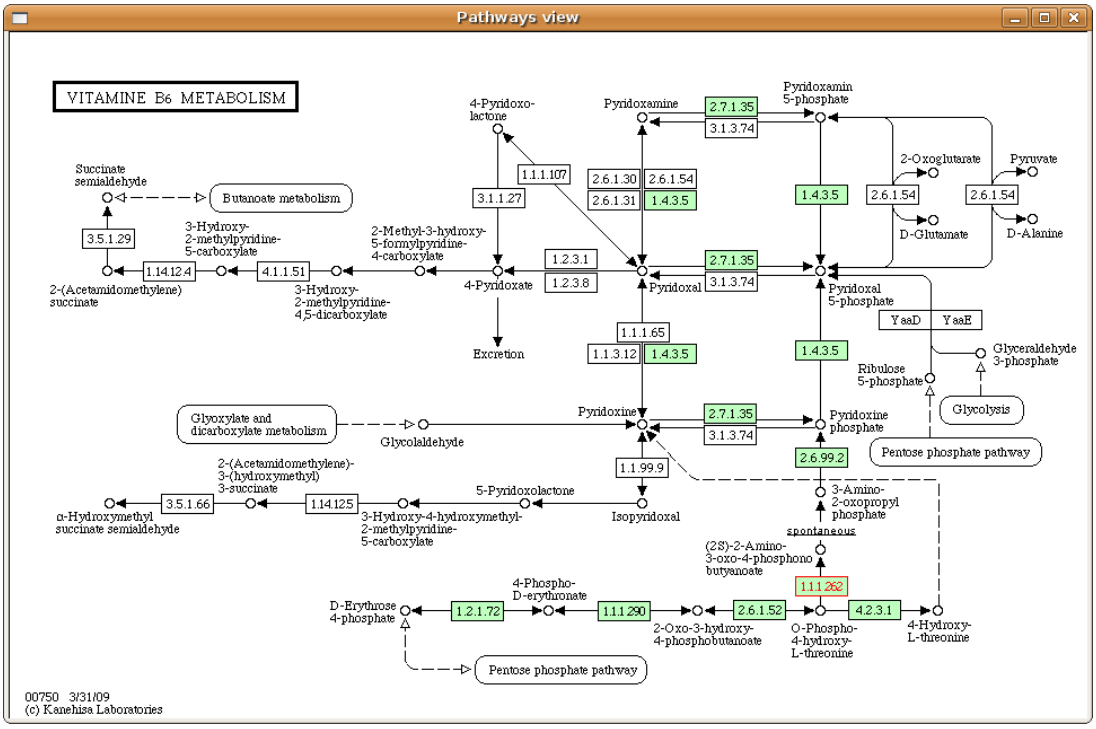


Figure 5.5 – Fenêtre *Pathways view*. Celle-ci présente la carte métabolique (via le site `kegg`) spécifiée par l'utilisateur. Dans cet exemple, il s'agit de la carte métabolique de la vitamine B6. Notons que la réaction correspondant au gène courant sélectionné est mise en évidence en rouge (ici, la réaction 1.1.1.262).

Identifiant NCBI	Chromosome
NC_000913	*Escherichia coli* K12
NC_002505	*Vibrio cholerae* 01 biovar eltor str. N16961 chromosome I
NC_002506	*Vibrio cholerae* 01 biovar eltor str. N16961 chromosome II
NC_009456	*Vibrio cholerae* 0395 chromosome I
NC_009457	*Vibrio cholerae* 0395 chromosome II
NC_006840	*Vibrio fischeri* ES114 chromosome I
NC_006841	*Vibrio fischeri* ES114 chromosome II

Table 5.1 – Ensemble des chromosomes analysés dans l'étude fonctionnelle des intervalles communs.

5.3.2 Résultats expérimentaux

Nous présentons dans cette partie les résultats obtenus sur le jeu de données présenté ci-dessus par le programme MATCH&WATCH sous le modèle *couplage maximal*.

La Table 5.2 montre les résultats quantitatifs obtenus par le programme MATCH&WATCH. Notons que les temps d'exécution du programme `InParanoid` sont beaucoup plus longs que ceux concernant le calcul du nombre d'intervalles communs. Cependant, pour les paires de génomes de grande taille (NC_000913 vs NC_002505, NC_000913 vs NC_009457 et NC_000913 vs NC_006840), seule

l'heuristique IILCS−CIRC$_M$ donne des résultats en un temps acceptable. La Figure 5.6 et la Table 5.2 montrent la distribution des intervalles suivant leur type : droit, inversé ou non structuré. En moyenne, 42% des intervalles communs maximaux sont droits, 48% sont inversés et 10% sont non structurés, montrant une proportion très importante de la conservation de l'ordre des gènes pour chaque comparaison (90%).

	taille des génomes			temps d'exécution		intervalles communs maximaux			
génome G_1	*E. coli*	G_1	méthode	InParanoid (s)	calcul (s)	nombre	droits	inversés	non structurés
NC_002505	4243	2742	IILCS−CIRC$_M$	1144	62	275	117(43%)	134(48%)	24(9%)
NC_002506	4243	1093	LPB−ICOM−CIRC$_M$	638	23	50	18(36%)	23(46%)	9(18%)
NC_009456	4243	1133	LPB−ICOM−CIRC$_M$	651	22	55	17(31%)	27(49%)	11(20%)
NC_009457	4243	2742	IILCS−CIRC$_M$	1199	59	278	114(41%)	141(51%)	23(8%)
NC_006840	4243	2586	IILCS−CIRC$_M$	1012	1	255	118(46%)	119(47%)	18(7%)
NC_006841	4243	1175	IILCS−CIRC$_M$	715	1	62	26(42%)	24(39%)	12(19%)

Table 5.2 – Intervalles communs maximaux obtenus sous le modèle *couplage maximal* en comparant *Escherichia coli* (identifiant NCBI NC_000913) avec quatre chromosomes de *Vibrio cholerae* (NC_002505, NC_002506, NC_009456 and NC_009457) et deux chromosomes de *Vibrio fischeri* (NC_006840 et NC_006841).

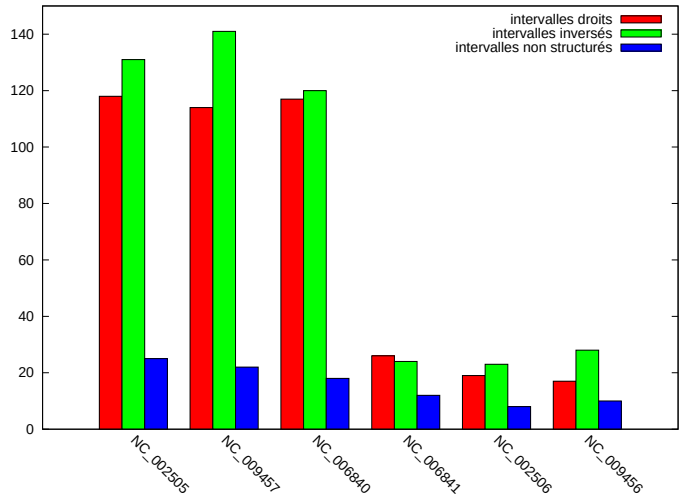


Figure 5.6 – Nombre d'intervalles communs obtenus et distribution selon leur type : droit, inversé ou non structuré, pour chacune des six comparaisons avec *E. coli*.

Nous proposons maintenant une première étude portant sur l'aspect fonctionnel des intervalles communs. En particulier, nous nous concentrons sur la comparaison de la paire de chromosomes *Escherichia coli* (NC_000913) et *Vibrio cholerae* (NC_009457) et étudions tout d'abord en détail plusieurs intervalles

communs. Puis, nous analysons les différents filtrages effectués (FILTRE$_A$ et FILTRE$_P$) permettant de mettre en évidence les intervalles communs les plus prometteurs.

Analyse biologique de quatre intervalles communs.

Nous analysons ici en détail quatre des intervalles communs obtenus par notre méthode, qui sont ceux présentés dans la Figure 5.7.

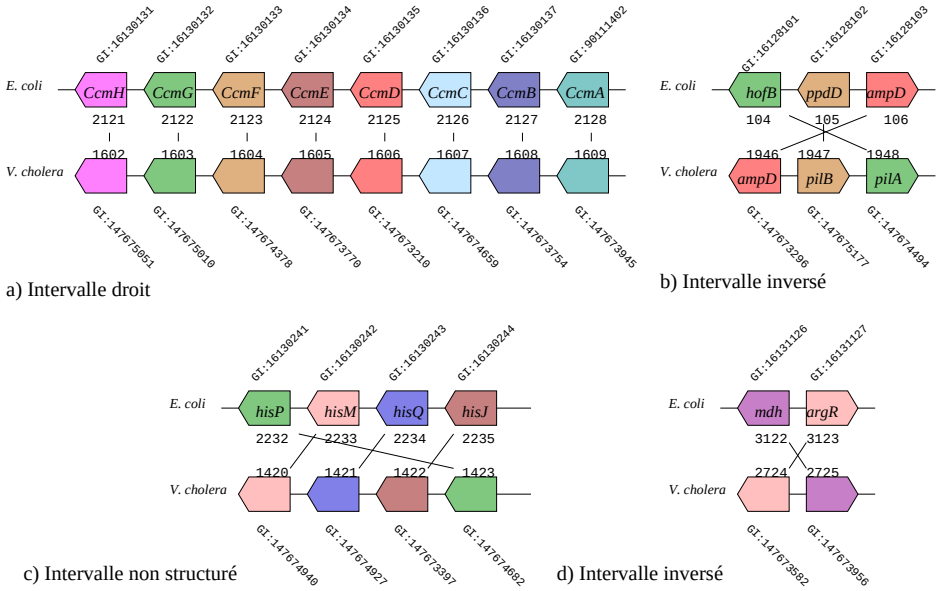


Figure 5.7 – Illustration des trois types d'intervalles observés en comparant *E. coli* (NC_000913) et *V. cholerae* (NC_009457). Chaque représentation d'un intervalle commun indique l'identifiant du gène (par exemple, GI16130131), la position relative des gènes au sein du génome (par exemple, 2121), les noms usuels des gènes lorsque ces noms sont connus (par exemple, $CcmH$), l'orientation du gène (orientation de la flèche). Pour un même intervalle commun, deux gènes homologues sont de même couleur.

Intervalles droits. Les intervalles droits montrent une parfaite conservation de l'ordre des gènes, comme dans la Figure 5.7 a). Sur la base des connaissances expérimentales présentes dans la base EcoCyc [KCVGC$^+$05], les gènes *CcmA*, *CcmB*, *CcmC*, *CcmD*, *CcmE*, *CcmF*, *CcmG* et *CcmH* sont les gènes codant pour des protéines, constituants du complexe de la cytochrome C. Ces gènes appartiennent au même opéron dont la régulation est contrôlée par *ccmAp* [TRCC98]. Par ailleurs, des études expérimentales montrent que les mutants de l'un de ces gènes sont déficients et ne sont plus capables de produire des cytochromes de type C [TM00]. Leur présence dans un intervalle commun droit met l'accent sur le fait que ces gènes sont conservés par leur séquence d'ADN puisque l'homologie a été détectée, mais aussi par leur disposition relative sur les génomes bactériens (i.e. constituants d'un intervalle commun). Ce résultat confirme l'intérêt particulier de trouver ce type d'intervalle commun pour aider à l'identification de gènes hautement fonctionnels.

Intervalles inversés. Les intervalles inversés montrent les ensembles de gènes dont l'ordre est inversé entre les deux génomes, comme ceux de la Figure 5.7 b) et 5.7 d). Pour illustration, l'intervalle de la Figure 5.7 b) correspond à l'ensemble des gènes liés au pilus, appendice se situant à la surface de la paroi de nombreuses bactéries. Fréquents chez les bactéries, ils sont impliqués dans plusieurs fonctions de colonisation, comme par exemple celle de la muqueuse intestinale. Pour des motivations cliniques, l'homologie de ces gènes (entre *E. coli* et *V. cholerae* et entre *E. coli* et *V. fischeri*) a déjà été identifiée expérimentalement dans [RUC+05]. Dans [FM99], Fullner et Mekalanos décrivent par ailleurs l'organisation de ce même opéron grâce à sept gènes. L'intervalle commun indiqué dans la Figure 5.7 b) suggère cependant une conservation de trois gènes seulement. Toutefois, notons que la base EcoCyc [KKS+07] ne montre pas de preuves de haute qualité d'une unité transcriptionnelle de ces sept gènes.

Intervalles non structurés. Les intervalles non structurés correspondent aux intervalles communs qui ne sont ni droits, ni inversés. La Figure 5.7 c) illustre ce cas. Cet intervalle décrit également l'agencement de quatre gènes qui appartiennent à un même opéron : l'opéron `hisJp`. Celui-ci est activé par Hns et réprimé par ArgR. Il produit les sous-unités d'un transporteur ABC. Notons encore que le gène *ArgR* qui encode ArgR correspond, avec *mdh*, à un intervalle commun inversé (voir la Figure 5.7 d)). Le gène *mdh* produit la protéine Mdh qui est une sous-unité d'un malate déshydrogénase. Le produit de celui-ci interagit dans plusieurs processus biologiques (métabolisme des glucides, néoglucogenèse, glycolyse, cycle de l'acide tricarboxylique, cycle intermédiaire du métabolisme de l'acide tricarboxylique, métabolisme du malate, fermentation, respiration anaérobie, catabolisme du glyoxylate, catabolisme des glucides). Encore une fois, ces résultats confirment que les intervalles communs peuvent être associés à des opérons, et peuvent de plus, comme dans ce cas, être associés avec un ensemble de gènes contrôlant l'opéron. En supposant que les intervalles communs correspondent à une conservation d'une fonction biologique pour les génomes comparés, une fonction de régulation apparaît aussi importante que les autres fonctions métaboliques. Comme dans le cas présent, l'intervalle commun montre une unité de régulation contrôlant des processus biologiques distincts via des opérons distincts.

Cette analyse montre qu'un certain nombre d'intervalles communs correspondent à une fonctionnalité précise. Cette étude doit être portée sur l'ensemble des intervalles communs obtenus afin d'estimer au mieux la corrélation entre les intervalles communs et les fonctions biologiques.

Filtrage des intervalles communs.

Nous nous intéressons maintenant aux différentes méthodes de filtrage proposées en Section 5.2.2, à savoir les filtrages FILTRE_A et FILTRE_P. Nous montrons dans la Figure 5.8 les nombres d'intervalles communs maximaux, annotés et pertinents obtenus par les différents processus de filtrage. Nous pouvons constater tout d'abord que le premier filtrage FILTRE_A réduit considérablement le nombre d'intervalles, puisque celui-ci retire environ 75% des intervalles communs maximaux. Cela s'explique en grande partie par le manque d'annotations : la majorité des intervalles communs maximaux ne satisfont pas les deux premières conditions des intervalles annotés, à savoir (i) contenir au moins un gène couplé annoté et (ii) que la moitié des gènes couplés soient annotés. Par conséquent, nous ne pouvons pas conclure que les intervalles communs retirés par le filtre FILTRE_A ne correspondent pas à une fonction biologique spécifique ; en revanche si celle-ci existe, elle n'est pas référencée dans la base d'annotations *Ecocyc*.

Le filtrage FILTRE_P permet quant à lui de sélectionner, parmi les intervalles annotés, les plus certains d'après la *p-valeur* calculée par l'application `GO-TermFinder`. Ce filtrage réduit drastiquement

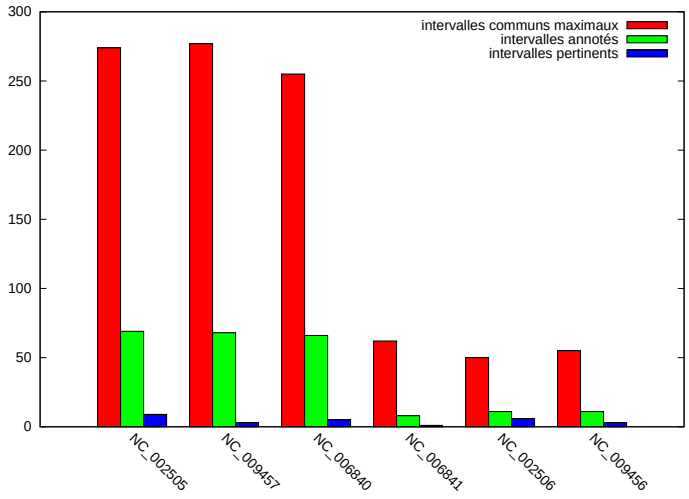


Figure 5.8 – Nombre d'intervalles communs maximaux, annotés et pertinents obtenus par les méthodes de filtrage FILTRE$_A$ et FILTRE$_P$, pour chacune des six comparaisons.

le nombre d'intervalles considérés, en passant de 233 intervalles annotés à 28 intervalles communs pertinents. Ceci s'explique surtout en raison du paramètre ϵ choisi, ici fixé à 10^{-4} afin d'assurer un filtrage strict (cf. algorithme FILTRE$_P$ page 111). En se focalisant sur le génome *E. Coli*, sept intervalles pertinents apparaissent lors de deux ou trois comparaisons, réduisant ainsi le nombre d'intervalles communs pertinents **distincts** sur *E. Coli* à quinze. Nous listons dans la Table 5.9 ces quinze intervalles. Notons tout d'abord que quatre intervalles pertinents sont inclus dans un de plus grande taille. Il s'agit des intervalles de position relative sur *E. coli* [70-73], [689-697], [2106-2107] et [3198-4002]. Pour une comparaison, chacun de ces quatre intervalles était un intervalle maximal. En revanche, pour les autres comparaisons, ces intervalles ont pu être étendus, en raison de l'absence d'une homologie pour certains génomes ou bien, en raison d'un réarrangement génomique induisant une restructuration de l'ordre des gènes de l'intervalle.

Remarquons également que dans la Table 5.9 quatre intervalles pertinents ont été obtenus dans trois comparaisons sur six possibles ([1953-1960], [2186-2191], [3171-3198], [3609-3621] et [3877-3880]). Ces intervalles se retrouvent dans quatre génomes, ce qui montre qu'ils sont fortement conservés. En se basant sur la base *RegulonDB*[‡] qui répertorie les opérons de *E. coli* (Escherichia coli K-12), nous pouvons constater la forte corrélation de ces intervalles à un ou plusieurs opérons :

[1953-1960] : Les huit gènes de l'intervalle (hisC, hisA, hisG, hisF, hisH, hisB, hisD, hisI) appartiennent tous à l'opéron nommé `hisLGDCBHAFI`. Seul le gène hisL appartenant aussi à cet opéron ne figure pas dans cet intervalle. Notons également que la meilleure annotation commune *histidine biosynthesis* n'est pas répertoriée pour le gène hisI.

[2186-2191] : Cinq gènes de l'intervalle sur six appartiennent à l'opéron `menFDHBCE`, plus précisément les gènes menB, menD, menF, menE, menC. Le sixième gène, nommé yfbB, n'apparaît pas dans la base *RegulonDB*. Ceci s'explique par une divergence dans les bases utilisées pour le

[‡]http ://regulondb.ccg.unam.mx/

positions relatives	nombre de comparaisons	nombre de gènes annoté	% gènes annotés	*p-valeur*	Meilleur terme	gènes
[70-73]	1	4	100%	2.3e-15	leucine metabolism	leuD leuC leuB leuA
[70-77]	1	6	86%	1.9e-20	branched chain family amino acid metabolism	ilvI, leuA, leuB, ilvH, leuC, leuD
[679-697]	2	8	73%	8.3e-20	aerobic respiration	sdhB, sucC, sdhD, sdhA, gltA, sucD, sdhC
[689-697]	1	7	70%	1.0e-20	aerobic respiration	sdhB, sucC, sdhD, sdhA, gltA, sucD, sdhC
[736-739]	1	3	75%	3.0e-12	biotin metabolism	bioB, bioF, bioC
[855-857]	1	3	100%	3.0e-12	dimethyl sulfoxide reductase complex	dmsA, dmsB, dmsC
[1953-1960]	3	8	100%	3.15e-26	histidine biosynthesis	hisC, hisA, hisG, hisF, hisH, hisB, hisD
[2106-2108]	1	3	100%	3.0e-12	microcin transport	yejE, yejB, yejA
[2106-2109]	2	4	100%	1.5e-16	microcin transport	yejE, yejF, yejB, yejA
[2186-2191]	3	3	100%	3.2e-22	fat-soluble vitamin metabolism	menB, menD, menF, menE, yfbB, menC
[3171-3198]	3	20	71%	3.1e-23	protein biosynthesis	rpsH, rpsE, rpsC, rpsS, rpsQ, rpsM, rpsN, rpsJ, rpsD, rpsK
[3609-3621]	3	11	92%	2.6e-28	nucleoside phosphate metabolism	atpF, atpA, atpH, atpD, atpG, atpC, atpB, atpE
[3877-3880]	4	4	100%	1.5e-16	disaccharide transport	malG, malE, malF, malK
[3998-4002]	1	5	100%	1.1e-14	fermentation	frdC, frdB, frdD, frdA
[3991-4021]	1	11	52%	7.4e-13	fermentation	frdC, frdB, frdD, frdA

Figure 5.9 – Liste des intervalles pertinents distincts obtenus sur *E. coli*. La colonne *positions relatives* indique la position des intervalles communs dans le génome *E. coli*. Le *nombre de comparaisons* indique le nombre de fois où cet intervalle a été obtenu sur les six comparaisons effectuées. Le *nombre* et le *pourcentage des gènes annotés* indiquent combien de gènes couplés de l'intervalle sont annotés dans la base *Ecocyc*. Enfin, la colonne *gènes* précise les gènes ayant l'annotation du *meilleur terme* trouvé par `GO-TermFinder`.

nom d'un même gène, à savoir menH dans *RegulonDB* et yfbB dans la base *Kegg*.

[3171-3198] : Les 28 gènes de l'intervalle composent exactement trois opérons distincts :
– `rpsMKD-rpoA-rplQ`
– `rplNXE-rpsNH-rplFR-rpsE-rpmD-rplO-secY-rpmJ`
– `rpsJ-rplCDWB-rpsS-rplV-rpsC-rplP-rpmC-rpsQ`
Par notre outil, nous avons détecté que ces trois opérons sont conservés ensemble pour trois comparaisons, ce qui indique une forte connectivité de ces opérons.

[3609-3621] : Les huit gènes annotés par le meilleur terme *nucleoside phosphate metabolism* forment ensemble l'opéron `atpIBEFHAGDC`.

Cette analyse montre que le filtrage FILTRE$_P$, que nous avons voulu strict, engendre un ensemble d'intervalles communs ayant une fonctionnalité biologique précise. De plus, nous mettons en évidence des manques d'annotations ou des incohérences de noms de gènes (comme par exemple pour le gène yfbB). Cette étude, bien que prometteuse, nécessite néanmoins d'être effectuée sur un plus grand nombre d'intervalles communs.

5.3.3 Conclusion de cette étude

Nous avons utilisé dans cette étude le programme MATCH&WATCH sur *Escherichia coli K12* et six chromosomes de *Vibrio cholerae* et *Vibrio fischeri*. Les résultats de ces travaux suggèrent que les intervalles communs fournissent des informations utiles sur les génomes bactériens et peuvent aider à mettre en évidence des ensembles de gènes possédant des propriétés fonctionnelles. En effet, plusieurs cas analysés montrent une réelle fonction biologique associée aux intervalles communs étudiés. Un examen approfondi de l'analyse biologique de chaque intervalle commun maximum et des tests sur de plus grands jeux de données doivent être maintenant réalisés afin de confirmer ces résultats.

5.4 Conclusion

Dans ce chapitre, nous avons présenté un programme nommé MATCH&WATCH permettant de mettre en évidence des ensembles de gènes conservés en calculant les intervalles communs de deux génomes bactériens. Pour cela, nous avons mis en place un protocole permettant de calculer les intervalles communs à partir des données brutes, contenant seulement les séquences protéiques. Ce protocole est spécifiquement dédié aux génomes bactériens, c'est-à-dire aux génomes circulaires. Nous avons également proposé un outil de visualisation permettant à l'utilisateur un meilleur traitement des informations, de par les différentes fonctionnalités proposées : visualisation des intervalles communs, liens sur la base *kegg* et les cartes métaboliques, etc.

De plus, plusieurs filtrages ont été proposés, offrant ainsi la possibilité de réduire graduellement le nombre d'intervalles communs et de ne garder que ceux potentiellement intéressants. Une première expérimentation a permis de montrer l'aspect fonctionnel des intervalles communs en s'appuyant sur la base d'annotation *Ecocyc* et la base *RegulonDB* et suggère un réel intérêt au calcul des intervalles communs pour la mise en évidence des régions fortement conservées.

Conclusion

Dans ce mémoire, nous nous sommes intéressés au problème de la comparaison de génomes. Plus particulièrement, notre travail s'est porté sur les problèmes de calcul de mesures entre deux génomes en présence de gènes dupliqués, en considérant les mesures *nombre de points de cassure* (problèmes BD_X), *nombre d'adjacences* (ADJ_X), *nombre d'intervalles communs* ($ICOM_X$) et *nombre d'intervalles conservés* ($ICONS_X$). Afin de prendre en compte les gènes dupliqués, nous nous sommes appuyés sur la notion de couplage des gènes pour laquelle trois variantes existent : le *couplage exemplaire* proposé par Sankoff [San99] ; le *couplage maximal* introduit par Tang et Moret [TM03] ; le *couplage intermédiaire* que nous introduisons dans ce mémoire.

Dans ce cadre, nous avons exposé plusieurs résultats théoriques pour les différents problèmes étudiés. Tout d'abord, nous avons montré l'équivalence des problèmes BD_X et ADJ_X pour les deux modèles *couplage exemplaire* et *couplage maximal*, alors que ceux-ci se révèlent de nature différente pour le modèle *couplage intermédiaire*. D'autres travaux se sont portés sur les problèmes ZBD_X consistant à trouver un couplage des gènes de deux génomes n'impliquant aucun point de cassure. Le problème ZBD_E (ZBD_X, modèle *couplage exemplaire*), a déjà été prouvé **NP**-Complet dans [CFZ06]. Nous avons démontré de plus la **NP**-Complétude de ZBD_E et ZBD_I (ZBD_X, modèle *couplage intermédiaire*) même dans le cas où aucun gène n'apparaît plus de deux fois sur un des génomes, impliquant de ce fait l'inexistence d'algorithme d'approximation pour BD_E et BD_I sous ces même conditions. À l'opposé, nous avons montré que ZBD_X est dans **P** pour le modèle *couplage maximal*.

Nous avons également prouvé l'**APX**-Difficulté des problèmes $ICOM_X$, $ICONS_X$ et BD_X, déjà prouvés **NP**-Complets, respectivement dans [CFRV06], [BR05] et [Bry00]. Cette complexité tient également dans le cas où l'un des génomes ne contient aucun gène dupliqué et qu'aucun gène n'est présent plus de deux fois sur le second génome. Ce résultat engendre l'impossibilité de résoudre ces problèmes par un schéma d'approximation ; en revanche, des algorithmes d'approximation à ratio constant peuvent être envisagés. Kolman et ses collaborateurs proposent divers ratios d'approximation pour la mesure *nombre de points de cassure* ([GKZ04, KW06]) en se restreignant à deux génomes de même contenu génique. Une première perspective de travail envisageable est de lever cette restriction. Un vaste champ d'étude consiste également à rechercher des algorithmes d'approximation pour les problèmes $ICOM_X$ et $ICONS_X$, en se limitant tout d'abord aux cas de génomes équilibrés. Plusieurs problèmes restent également ouverts. Le premier concerne l'existence d'algorithme d'approximation pour BD_E et BD_I dans le cas où l'un des génomes ne contient aucun gène dupliqué. En effet, sous ces conditions, ZBD_E et ZBD_I sont résolubles polynomialement, ce qui induit la possible existence d'un algorithme d'approximation, mais non découvert à ce jour. Une autre perspective consiste à considérer les approximations dites *faibles* comme proposé dans [CFZ06]. Ces approximations sont utiles pour résoudre les problèmes dont la valeur optimale peut être égale à zéro. Chen et al. ont montré que BD_E n'admet pas d'approximation faible dans le cas général. En revanche, la question consistant à déterminer s'il existe de telles approximations pour un nombre de gènes dupliqués fixé reste encore ouverte.

Il existe plusieurs méthodes exactes de résolution des problèmes BD_X en présence de gènes dupliqués. En particulier, Sankoff a proposé dans [San99] un algorithme de type *Branch and Bound* pour la résolution de BD_E, puis Nguyen et ses collaborateurs dans [NTZ05]. Pour BD_M, Blin et. al ont pro-

posé quant à eux un algorithme de type *branch-and-cut* dans [BCF04]. Nous avons proposé dans ce mémoire une nouvelle approche exacte valable pour toutes les mesures étudiées. Cette méthode, dite *pseudo-booléenne*, se base sur une transformation de l'instance à traiter en une instance d'un problème de contraintes à variables booléennes. Les résultats obtenus pour cette approche lors de tests sur un jeu de données réel sont de deux types. Pour les mesures *nombre de points de cassure* et *nombre d'adjacences*, la méthode pseudo-booléenne s'est révélée efficace puisque plus de 95% des instances ont été résolues. En revanche, pour la mesure *nombre d'intervalles communs*, la méthode atteint sa limite de par le manque de solutions obtenues en un temps acceptable. Cette différence s'explique en partie par la combinatoire beaucoup plus importante pour cette mesure que pour les points de cassure.

Nous avons également proposé deux nouvelles approches non exactes, à savoir (i) une heuristique nommée IILCS_X basée sur l'heuristique ILCS_X étudiée dans [BCF05] (ii) une méthode hybride s'appuyant sur l'heuristique IILCS_X et sur la méthode exacte pseudo-booléenne. Une étude détaillée a permis notamment de montrer (1) que IILCS_X est significativement meilleure que ILCS_X ; (2) que la méthode hybride est significativement meilleure que IILCS_X mais reste dans certains cas trop lente de par sa nature exponentielle pour la mesure *nombre d'intervalles communs*. Il en ressort clairement que l'heuristique IILCS_X offre d'excellents résultats sur le jeu de données étudié, en terme de qualité de réponse par rapport à la solution optimale et d'autre part, en terme de temps d'exécution. Que ce soit pour le calcul du nombre d'intervalles communs ou du nombre d'adjacences, IILCS_X obtient des résultats très proches de la valeur optimale. Il est étonnant qu'une même heuristique donne de très bons résultats pour deux mesures distinctes. Cela semble suggérer que ces mesures sont proches dans le sens où elles mettent en évidence la même information quant à l'ordre des gènes, information retrouvée par IILCS_X.

Les intervalles communs peuvent être vus comme une généralisation des adjacences, ces notions se basant toutes deux sur la contiguïté des gènes, avec pour les intervalles communs une souplesse quant à l'ordre des gènes. Il serait intéressant d'étudier l'impact du choix de telle ou telle mesure lorsque celle-ci est utilisée, par exemple pour réaliser des phylogénies. En d'autres termes, il s'agirait de déterminer s'il existe une différence significative entre les arbres d'évolution construits à partir du nombre d'adjacences et ceux basés sur le nombre d'intervalles communs. Une seconde perspective serait de réaliser la même étude sur les notions de *nombre MAD* (pour Maximum Adjacency Disruption number) et *nombre SAD* (pour Summed Adjacency Disruption number) introduits dans [SH05] et peu étudiés à ce jour. Outre le choix de la mesure, le modèle de couplage des gènes est aussi important. Nous avons dans ce mémoire introduit le modèle *couplage intermédiaire*, celui-ci se plaçant entre les deux modèles existants. Ce modèle est encore peu étudié, et une perspective de travail serait également d'établir le réel impact du choix du modèle lors de la construction de matrices de distances.

Les trois méthodes proposées (approche pseudo-booléenne, heuristique, approche hybride) ont chacune leurs avantages et leurs inconvénients (exact ou non exact, temps d'exécution). Dans un but pratique où nous souhaitons obtenir une solution la meilleure qui soit en un temps acceptable, il serait intéressant de pouvoir choisir *a priori* la méthode à utiliser. Pour cela, il faudrait être capable d'estimer la complexité du programme pseudo-booléen généré. S'il s'avère trop complexe, nous choisirons alors la méthode hybride, voire l'heuristique. Ce choix a priori nous semble difficile à mettre en œuvre. En effet, outre la combinatoire du problème, l'efficacité de la résolution dépend aussi du solveur utilisé. Malheureusement, il n'existe pas de solveur efficace sur tous les problèmes ; chaque solveur possède ses propres caractéristiques. Une perspective serait d'étudier ces caractéristiques afin d'orienter l'écriture du programme pseudo-booléen, ceci dans le but d'obtenir une meilleure efficacité de la résolution.

Dans le Chapitre 5, nous nous sommes intéressés plus particulièrement aux intervalles communs.

Notre objectif a été de montrer leur aspect fonctionnel, et non plus de les considérer uniquement en terme de mesure entre deux génomes. Pour cela, nous avons tout d'abord proposé l'outil MATCH&WATCH permettant de calculer les intervalles communs de deux génomes bactériens, prenant en compte la circularité de ceux-ci, et offrant un outil de visualisation des résultats aidant ainsi à leur analyse. Une première étude a montré que les intervalles communs peuvent être utilisés pour identifier des régions conservées au sein des génomes et que certains correspondent à une fonction biologique précise. Les résultats sont prometteurs et une analyse de chaque intervalle commun est d'ores et déjà planifiée dans le futur afin de confirmer ces résultats.

Une des perspectives majeures serait de remplacer la notion d'intervalle commun par des notions plus souples correspondant également à des ensembles de gènes conservés comme par exemple les *équipes de gènes*, introduits dans [BCR02] et développés plus tard dans [HG04, KCY05, LHXH08]. De même, Amir et al. proposent dans [AGS07] la notion d'*intervalle commun approché* étendant les intervalles communs définis sur des chaînes (et non plus des permutations) par Schmidt et Stoye dans [SS04]. Ces différents travaux s'intéressent plus à l'énumération des ensembles de gènes conservés qu'au calcul de leur nombre. En revanche, le calcul du nombre de ces ensembles peut s'avérer plus rapidement résoluble que leur énumération, comme il a été montré dans [BCdMR08] pour les intervalles communs. Cette différence de complexité est à étudier pour les nouvelles notions présentées ci-dessus.

En définitive, nous nous sommes intéressés dans ce mémoire à la comparaison de génomes deux à deux avec gènes dupliqués, et avons apporté notre contribution par des études théoriques, puis pratiques à travers diverses méthodes de résolution. Devant l'accroissement de plus en plus rapide du nombre d'espèces séquencées, la comparaison de génomes multiples semble s'imposer, ouvrant ainsi la voie à de futurs problèmes tout aussi passionnants.

Bibliographie

[AEFR09] S. Angibaud, D. Eveillard, G. Fertin, and I. Rusu. Comparing bacterial genomes by searching their common intervals. In *Proc. 1st International Conference on Bioinformatics and Computional Biology (BICoB 'O9)*, volume 5462 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2009.

[AFR+07] 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 *Proc. 5th RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG '07)*, volume 4751 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2007.

[AFR08a] S. Angibaud, G. Fertin, and I. Rusu. On the approximability of comparing genomes with duplicates. In *Proc. 2nd International Workshop on Algorithms and Computation (WALCOM '08)*, volume 4921 of *Lecture Notes in Computer Science*, pages 34–45. Springer, 2008.

[AFR+08b] 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.

[AFR+09] 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*, 4(1) :19–53, 2009.

[AFRV06] S. Angibaud, G. Fertin, I. Rusu, and S. Vialette. How pseudo-boolean programming can help genome rearrangement distance computation. In *Proc. 4th RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG '06)*, volume 4205 of *Lecture Notes in Computer Science*, pages 75–86. Springer, 2006.

[AFRV07] S. Angibaud, G. Fertin, I. Rusu, and S. Vialette. A pseudo-boolean general framework for computing rearrangement distances between genomes with duplicates. *Journal of Computational Biology*, 14(4) :379–393, 2007.

[AGS07] A. Amir, L. Gąsieniec, and R. Shalom. Improved approximate common interval. *Information Processing Letters*, 103(4) :142–149, 2007.

[Ajt88] M. Ajtai. The complexity of the pigeonhole principle. In *Proc. 29th Symposium on Foundations of Computer Science (FOCS '88)*, pages 346–355, Los Alamitos, CA, USA, 1988. IEEE Computer Society.

[AK97] P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. In *Proc. of the 3rd Italian Conference on Algorithms and Complexity (CIAC '97)*, pages 288–298. Springer-Verlag, 1997.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast : a new generation of protein database search programs. *Nucleic Acids Research*, 25(17) :3389–3402, September 1997.

[Bar95] P. Barth. A Davis-Putnam based enumeration algorithm for linear pseudo-boolean optimization. Technical Report MPI-I-95-2-003, Max Planck Institut Informatik, 1995. 13 pages.

[BCdMR08] A. Bergeron, C. Chauve, F. de Montgolfier, and M. Raffinot. Computing common intervals of K permutations, with applications to modular decomposition of graphs. *SIAM Journal on Discrete Mathematics*, 22(3) :1022–1039, 2008.

[BCF04] G. Blin, C. Chauve, and G. Fertin. The breakpoint distance for signed sequences. In *Proc. CompBioNets 2004*, volume Text in Algorithms, Volume 3, pages 3–16. King's College London, 2004.

[BCF05] G. Blin, C. Chauve, and G. Fertin. Genes order and phylogenetic reconstruction : Application to γ-proteobacteria. In *Proc. RECOMB international workshop on comparative genomics (RECOMB-CG '05)*, volume 3678 of *Lecture Notes in Bioinformatics*, pages 11–20. Springer-Verlag, 2005.

[BCR02] A. Bergeron, S. Corteel, and M. Raffinot. The algorithmic of gene teams. In *Proc. 2nd International Workshop on Algorithms in Bioinformatics (WABI '02)*, volume 2452 of *Lecture Notes in Computer Science*, pages 464–476. Springer-Verlag, 2002.

[BF94] P. Berman and M. Fürer. Approximating maximum independent set in bounded degree graphs. In *Proc. 5th annual ACM-SIAM symposium on Discrete algorithms (SODA '94)*, pages 365–371, 1994.

[BFSV09] G. Blin, G. Fertin, F. Sikora, and S. Vialette. The exemplar breakpoint distance for non-trivial genomes cannot be approximated. In *Proc. 3rd Workshop on Algorithms and Computation (WALCOM '09)*, volume 5431 of *Lecture Notes in Computer Science*, pages 357–368. Springer, 2009.

[BP93] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. In *Proc. 34th Annual Symposium on Foundations of Computer Science (FOCS '93)*, pages 148–157, Washington, DC, USA, 1993. IEEE Computer Society.

[BP98] V. Bafna and P. A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2) :224–240, 1998.

[BP02] G. Bourque and P. A. Pevzner. Genome-scale evolution : reconstructing gene orders in the ancestral species. *Genome Research*, 12(1) :26–36, 2002.

[BR05] G. Blin and R. Rizzi. Conserved interval distance computation between non-trivial genomes. In *Proc. 11th Annual International Conference on Computing and Combinatorics (COCOON '05)*, volume 3595 of *Lecture Notes in Computer Science*, pages 22–31. Springer, 2005.

[Bry00] D. Bryant. The complexity of calculating exemplar distances. In *Comparative Genomics : Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignement, and the Evolution of Gene Families*, pages 207–212. Kluwer Academic Publisher, 2000.

[BS03] A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. In *Proc. 9th International Computing and Combinatorics Conference (COCOON '03)*, volume 2697 of *Lecture Notes in Computer Science*, pages 68–79, 2003.

[BWG+04] E. I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, M. M. Cherry, and G. Sherlock. Go::termfinder-open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics*, 20(18) :3710–3715, 2004.

[BYHN+02] R. Bar-Yehuda, M. M. Halldórsson, J. Naor, H. Shachnai, and I. Shapira. Scheduling split intervals. In *Proc. 13th annual ACM-SIAM symposium on Discrete algorithms (SODA '02)*, pages 732–741, 2002.

[CFRV06] C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Genomes containing duplicates are hard to compare. In *Proc. International Workshop on Bioinformatics Research and Applications (IWBRA '06)*, volume 3992 of *Lecture Notes in Computer Science*, pages 783–790. Springer, 2006.

[CFX+07] Z. Chen, B. Fu, J. Xu, B. Yang, Z. Zhao, and B. Zhu. Non-breaking similarity of genomes with gene repetitions. In *Proc. 18th Annual Symposium on Combinatorial Pattern Matching (CPM '07)*, volume 4580 of *Lecture Notes in Computer Science*, pages 119–130. Springer, 2007.

[CFZ06] Z. Chen, B. Fu, and B. Zhu. The approximability of the exemplar breakpoint distance problem. In *Proc. 2nd International Conference on Algorithmic Aspects in Information and Management (AAIM '06)*, volume 4041 of *Lecture Notes in Computer Science*, pages 291–302. Springer-Verlag, 2006.

[CHLV05] M. Crochemore, D. Hermelin, G. M. Landau, and S. Vialette. Approximating the 2-interval pattern problem. In *Proc. 13th Annual European Symposium on Algorithms (ESA '05)*, volume 3669 of *Lecture Notes in Computer Science*, pages 426–437. Springer, 2005.

[CK03] D. Chai and A. Kuehlmann. A fast pseudo-boolean constraint solver. In *Proc. 40th Design Automation Conference (DAC '03)*, pages 830–835, 2003.

[CMM07] M. Charikar, K. Makarychev, and Y. Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. In *Proc. 8th annual ACM-SIAM symposium on Discrete algorithms (SODA '07)*, pages 62–68, 2007.

[CW92] 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.

[EMNS98] N. El-Mabrouk, J. H. Nadeau, and D. Sankoff. Genome halving. In *Proc. 9th Annual Symposium on Combinatorial Pattern Matching (CPM '98)*, volume 1448 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 1998.

[ES06] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2 :1–26, 2006.

[Fer05] H. Fernau. Parameterized algorithms : A graph-theoretic approach. Habilitationsschrift, Universität Tubingen, 2005.

[FLR+09] G. Fertin, A. Labarre, I. Rusu, É. Tannier, and S. Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology series. MIT Press, 2009.

[FM99] K. J. Fullner and J. J. Mekalanos. Genetic characterization of a new type IV-a pilus gene cluster found in both classical and El Tor biotypes of *Vibrio cholerae*. *Infection and Immunity*, 67(3) :1393–1404, 1999.

[GKZ04] A. Goldstein, P. Kolman, and Z. Zheng. Minimum common string partition problem : Hardness and approximations. In *Proc. 15th International Symposium on Algorithms and Computation (ISAAC '04)*, volume 3341 of *Lecture Notes in Computer Science*, pages 473–484. Springer, 2004.

[Gol80] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[GW79] J.R. Griggs and D.B. West. Extremal values of the interval number of a graph, I. *SIAM Journal on Algebraic and Discrete Methods*, 1 :1–7, 1979.

[HG04] X. He and M. H. Goldwasser. Identifying conserved gene clusters in the presence of orthologous groups. In *Proc. 8th annual international conference on Research in computational molecular biology (RECOMB '04)*, pages 272–280, New York, NY, USA, 2004. ACM.

[HS01] S. Heber and J. Stoye. Finding all common intervals of K permutations. In *Proc. 12th Annual Symposium on Combinatorial Pattern Matching (CPM '01)*, volume 2089 of *Lecture Notes in Computer Science*, pages 207–218. Springer, 2001.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KCVGC+05] I. M. Keseler, J. Collado-Vides, S. Gama-Castro, J. Ingraham, S. Paley, I. T. Paulsen, M. Peralta-Gil, and P. D. Karp. Ecocyc : a comprehensive database resource for *Escherichia coli*. *Nucleic Acids Research*, 33(Database issue) :D334–7, 2005.

[KCY05] S. Kim, J. Choi, and J. Yang. Gene teams with relaxed proximity constraint. In *Proc. IEEE Computational Systems Bioinformatics Conference (CSB '05)*, pages 44–55, Washington, DC, USA, 2005. IEEE Computer Society.

[KKS+07] P. D. Karp, I. M. Keseler, A. Shearer, Ma. Latendresse, M. Krummenacker, S. M. Paley, I. Paulsen, J. Collado-Vides, S. Gama-Castro, M. Peralta-Gil, A. Santos-Zavaleta, M. I. Peñaloza-Spínola, C. Bonavides-Martinez, and J. Ingraham. Multidimensional annotation of the *Escherichia coli* K-12 genome. *Nucleic Acids Research*, 35(22) :7577–90, 2007.

[KW06] P. Kolman and T. Waleń. Reversal distance for strings with duplicates : Linear time approximation using hitting set. In *Proc. 4th International Workshop Approximation and Online Algorithms (WAOA '06)*, volume 4368 of *Lecture Notes in Computer Science*, pages 279–289. Springer, 2006.

[LDM03] 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.

[LHX09] X. Ling, X. He, and D. Xin. Detecting gene clusters under evolutionary constraint in a large number of genomes. *Bioinformatics*, pages btp027+, 2009.

[LHXH08] X. Ling, X. He, D. Xin, and J. Han. Efficiently identifying max-gap clusters in pairwise genome comparison. *Journal of Computational Biology*, 15(6) :593–609, 2008.

[LPW05] G. M. Landau, L. Parida, and O. Weimann. Using PQ Trees for comparative genomics. In *Proc. 16th Annual Symposium on Combinatorial Pattern Matching (CPM '05)*, volume 3537 of *Lecture Notes in Computer Science*, pages 128–143. Springer, 2005.

[MSM04] M. Marron, K. M. Swenson, and B. M. E. Moret. Genomic distances under deletions and insertions. *Theoretical Computer Science*, 325(3) :347–360, 2004.

[MV80] S. Micali and V. V. Vazirani. Algorithm for finding maximum matching in general graph. In *Proc. 21th Annual IEEE Symposium on Foundation of Computer Science*, pages 17–27. IEEE, 1980.

[NTZ05] C. Thach Nguyen, Y. C. Tay, and Louxin Zhang. Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics*, 21(10) :2171–2176, 2005.

[PY91] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3) :425–440, 1991.

[RSS01] M. Remm, C.E. Strom, and E.L. Sonnhammer. Automatic clustering of orthologs and inparalogs from pairwise species comparisons. *Journal of Molecular Biology*, 314 :1041–1052, 2001.

[RUC+05] E. G. Ruby, M. Urbanowski, J. Campbell, A. Dunn, M. Faini, R. Gunsalus, P. Lostroh, C. Lupp, J. McCann, D. Millikan, A. Schaefer, E. Stabb, A. Stevens, K. Visick, C. Whistler, and E. P. Greenberg. Complete genome sequence of *Vibrio fischeri* : a symbiotic bacterium with pathogenic congeners. *Proceedings of the National Academy of Sciences*, 102(8) :3004–9, 2005.

[San99] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11) :909–917, 1999.

[SB97] D. Sankoff and M. Blanchette. The median problem for breakpoints in comparative genomics. In *Proc. 3rd Annual International Conference on Computing and Combinatorics (COCOON '97)*, volume 1276 of *Lecture Notes in Computer Science*, pages 251–264, London, UK, 1997. Springer-Verlag.

[Sch98] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998.

[SH05] D. Sankoff and L. Haque. Power boosts for cluster tests. In *Proc. 3rd RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG '05)*, volume 3678 of *Lecture Notes in Computer Science*, pages 121–130. Springer, 2005.

[SS04] T. Schmidt and J. Stoye. Quadratic time algorithms for finding common intervals in two and more sequences. In *Proc. 15th Annual Symposium on Combinatorial Pattern Matching (CPM '04)*, volume 3109 of *Lecture Notes in Computer Science*, pages 347–359. Springer, 2004.

[SS06] H.M. Sheini and K.A. Sakallah. Pueblo : A hybrid pseudo-boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2 :165–189, 2006.

[SSK96] D. Sankoff, G. Sundaramand, and J. D. Kececioglu. Steiner points in the space of genome rearrangements. *International Journal of Foundations of Computer Science*, 7(1) :1–9, 1996.

[TH79] W.T. Trotter and F. Harary. On double and multiple interval graphs. *Journal of Graph Theory*, 3 :205–211, 1979.

[Tic84] W. F. Tichy. The string-to-string correction problem with block moves. *ACM Transactions on Computer Systems*, 2(4) :309–321, 1984.

[TM00] L. Thöny-Meyer. Haem-polypeptide interactions during cytochrome C maturation. *Biochimica et Biophysica Acta*, 1459(2-3) :316–24, 2000.

[TM03] J. Tang and B. M. E. Moret. Phylogenetic reconstruction from gene-rearrangement data with unequal gene content. In *Proc. Intl Workshop on Algorithms and Data Structures (WADS '03)*, volume 2748 of *Lecture Notes in Computer Science*, pages 37–46. Springer, 2003.

[TRCC98] S. Tanapongpipat, E. Reid, J. A. Cole, and H. Crooke. Transcriptional control and essential roles of the *Escherichia coli* ccm gene products in formate-dependent nitrite reduction and cytochrome C synthesis. *Biochemical Journal*, 334 (Pt 2) :355–65, 1998.

[UY00] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2) :290–309, 2000.

[Via04] S. Vialette. On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science*, 312(2-3) :223–249, 2004.

[WEHM82] G. A. Watterson, W. J. Ewens, T. E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theoretical Biology*, 99(1) :1–7, 1982.

Liste des tableaux

Liste des figures

Liste des exemples

Table des matières

Index

Comparaisons de génomes avec gènes dupliqués : étude théorique et algorithmes

Sébastien ANGIBAUD

Résumé

La génomique comparative étudie les similarités et/ou les dissimilarités entre génomes et permet d'établir des relations entre les espèces afin notamment de construire des phylogénies. Elle permet également de mettre en évidence des régions conservées au sein des génomes et de trouver ainsi des ensembles de gènes impliqués dans des processus biologiques conservés au cours de l'évolution. Dans ce mémoire, nous nous intéressons au calcul de mesures entre deux génomes en présence de gènes dupliqués, et plus particulièrement aux mesures à base de points de cassure, d'adjacences, d'intervalles communs et d'intervalles conservés.

Suivant une démarche informatique, nous proposons tout d'abord une étude avancée de la complexité algorithmique des problèmes rencontrés, en prouvant notamment pour la plupart d'entre eux soit leur NP-Complétude soit leur APX-Difficulté. Par la suite, nous exposons plusieurs méthodes de calcul de mesures entre deux génomes, à savoir (i) une approche exacte basée sur une transformation en un problème de contraintes à variables booléennes, (ii) une heuristique et (iii) une méthode hybride qui s'appuie sur la méthode exacte et l'heuristique proposées. Par une étude sur un jeu de données réel, nous montrons les qualités respectives de ces méthodes. Enfin, nous proposons un protocole de calcul des intervalles communs et mettons en évidence, par son utilisation et par un outil de visualisation, l'aspect fonctionnel de certains intervalles communs.

Mots-clés : Génomique comparative, Distances intergénomiques, Complexité algorithmique, Point de cassure, Adjacence, Intervalle commun, Intervalle conservé

Abstract

Comparative genomics consists in studying similarities/dissimilarities between genomes, and can be used to find relations between species in order to compute, for example, phylogenetic trees. Moreover, comparative genomics highlights conserved areas in genomes during the evolution and emphasizes sets of functionnal genes. In this thesis, we are interested in computing measures between two genomes with duplicated genes and, more particularly, we investigate breakpoint, adjacency, common interval and conserved interval based measures.

We first present some theoretical results by proving the NP-Completeness or the APX-Hardness of most of the studied problems. We then propose several methods to compute distances between two genomes: (i) an exact approach based on a transformation into a pseudo-boolean problem (ii) a heuristic (iii) a hybrid method using the exact method and the above mentioned heuristic. We next show their respective qualities on real data. Finally, we propose a general protocol to compute common intervals and highlight their functional aspects.

Keywords: Comparative genomics, Genomic distances, Computational complexity, Breakpoint, Adjacency, Common interval, Conserved interval