Combinatoire and Bio-informatique : Comparaison de structures d'ARN et calcul de distances intergénomiques

Guillaume Blin

ÉCOLE DOCTORALE STIM

« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DES MATÉRIAUX »

Année 2005

Combinatoire et Bio-informatique: Comparaison de structures d'ARN et calcul de distances intergénomiques

---

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE NANTES

Discipline : Informatique

présentée et soutenue publiquement par

Guillaume BLIN

le 17 novembre 2005

à l'UFR Sciences et Techniques, Université de Nantes

devant le jury ci-dessous

Président	:	Guillaume FERTIN, Professeur d'Université	Université de Nantes
Rapporteurs	:	Marie-France SAGOT, Directrice de Recherche	INRIA/INRIA Rhône-Alpes
		Hélène TOUZET, Chargée de Recherche CNRS (CR1)	Université de Lille
Examinateurs	:	Romeo RIZZI, Professeur d'Université	Università degli Studi di Trento
		Irena RUSU, Professeur d'Université	Université de Nantes
		Stéphane VIALETTE, Maître de conférences	Université Paris-Sud

Directeur de thèse : Pr. Irena RUSU
Encadrants de thèse : Pr. Guillaume FERTIN
Laboratoire : LINA (Laboratoire d'Informatique de Nantes Atlantique)

N° ED 0366-214

COMBINATOIRE ET BIO-INFORMATIQUE: COMPARAISON DE STRUCTURES D'ARN ET CALCUL DE DISTANCES INTERGÉNOMIQUES

Combinatorics and Computational Biology: RNA structure comparison and genomic distance computation

Guillaume BLIN

favet neptunus eunti

Université de Nantes

Guillaume BLIN

Combinatoire et Bio-informatique: Comparaison de structures d'ARN et calcul de distances intergénomiques

xxi+170 p.

Ce document a été préparé avec LaTeX2$_\varepsilon$ et la classe `these-IRIN` version 0.92 de l'association de jeunes chercheurs en informatique LOGIN, Université de Nantes. La classe `these-IRIN` est disponible à l'adresse :

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

Impression : these.tex – 9/12/2005 – 20:06

Révision pour la classe : $Id: these-IRIN.cls,v 1.3 2000/11/19 18:30:42 fred Exp

À ma maman chérie que j'aime et que j'adore!
À papi et nanou que j'aime tout autant
À ma belle famille adorée ;o)
Et à zoé – mon amour

— Guillaume BLIN,
Combinatoire et Bio-informatique: Comparaison de structures d'ARN et calcul de distances intergénomiques.

Remerciements

Un grand merci à Irena RUSU d'avoir accepté il y a trois ans de diriger ma thèse. Merci pour son accueil toujours chaleureux, ses nombreux conseils et son aide. Merci d'avoir toujours dit sans détour ce qui allait et n'allait pas; contribuant ainsi à mon épanouissement en tant que chercheur.

Merci à Guillaume FERTIN d'avoir accepté d'encadrer ma thèse. Merci de m'avoir donné goût à la complexité et l'algorithmique. Mais surtout merci de m'avoir initié au plaisir de relever des défis. Merci pour ses nombreux conseils.

Merci à Marie-France SAGOT et Hélène TOUZET d'avoir accepté de rapporter ma thèse. Merci à Stéphane VIALETTE, Romeo RIZZI, Danny HERMELIN et Cedric CHAUVE de m'avoir, de par nos collaborations, initié aux activités de recherches et d'avoir rendu possible cette thèse.

Je salue également les membres du Laboratoire d'Informatique de Nantes-Atlantique, et en particulier les membres de l'association Login et ceux de l'équipe Combi que je quitterai avec regrets après ces trois années. Merci aux membres de l'Institut d'électronique et d'informatique Gaspard-Monge de l'Université de Marne-la-vallée de m'avoir permis, en m'accueillant pour la rentrée 2006, de revenir sur la région parisienne.

Un merci à CEDRIC, DUDE, FREDDY, VIDAL, ANTOINE, POLLITO, GALLO, ADEL, FABRICE pour leur indéniable non-contribution à ce manuscrit. Un grand merci à CEDRIC et FREDDY de m'avoir permis de squatter chez eux durant ces six derniers mois. Merci à DUDE, VIDAL et POLLITO & GALLO de m'avoir fait découvrir de nouvelles merveilles gastronomiques.

Un grand merci à tous les membres de ma famille . . . Je ne vais pas faire de liste de peur d'en oublier. Merci à mon Amour de m'avoir supporté jusqu'au bout. Merci à ma maman chérie que j'aime et que j'adore encore après toutes ces années . . .

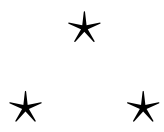
Notes de lecture

Glossaire Un glossaire se trouve à la fin de ce manuscrit (p. 162). Il contient les mots employés dans ce manuscrit dont nous avons jugé utile de rappeler la définition. Lorsqu'un mot est cité dans le glossaire, il apparaît, lors de sa première occurrence, en italique et est suivi de $\star$ dans le corps du texte. Une partie des définitions est issue du site `http://www.interstices.info`

Illustrations Toute illustration dont l'origine n'est pas mentionnée est l'oeuvre de l'auteur de ce manuscrit.

Collaborations La majorité des résultats présentés dans ce manuscrit est issue de plusieurs collaborations avec des chercheurs français et étrangers. Ci-après, vous trouverez la liste des chercheurs avec qui j'ai collaboré.

Guillaume FERTIN, Irena RUSU et Christine SINOQUET
LINA - FRE CNRS 2729 Université de Nantes,
2 rue de la Houssinière BP 92208 44322 Nantes Cedex 3 - FRANCE
{fertin,rusu,sinoquet}@lina.univ-nantes.fr



Cédric CHAUVE
LaCIM et Département d'Informatique
Université du Québec À Montréal
CP 8888, Succ. Centre-Ville
H3C 3P8, Montréal (QC), CANADA
chauve@lacim.uqam.ca

Danny HERMELIN
Department of Computer Science
University of Haifa,
Mount Carmel,
Haifa 31905 - ISRAEL
danny@cri.haifa.ac.il

Romeo RIZZI
Università degli Studi di Trento Facoltà di Scienze
Dipartimento di Informatica e Telecomunicazioni
Via Sommarive, 14 - I38050 Povo
Trento (TN) - ITALY
romeo.rizzi@unitn.it

Stéphane VIALETTE
LRI - UMR CNRS 8623
Faculté des Sciences d'Orsay
Université Paris-Sud
Bât 490, 91405 Orsay Cedex - FRANCE
vialette@lri.fr

Sommaire

Introduction

La bio-informatique, domaine de recherche interdisciplinaire par nature, est une branche théorique de la biologie qui puise ses fondements de la biologie, bien entendu, mais aussi des mathématiques, de la physique et de l'informatique. Les données exploitées en bio-informatique sont issues essentiellement d'expérimentations biologiques et sont stockées en masse dans des banques de données.

Du point de vue biologique, la bio-informatique, du fait des nombreux outils informatiques disponibles et de ses fondements théoriques, est un outil indispensable aux biologistes. Du point de vue informatique, la bio-informatique est un nouveau domaine d'application et constitue un nouveau domaine de recherche.

Plus précisément, dans le contexte algorithmique, la bio-informatique propose un ensemble de problèmes appliqués et constituant, pour certains, de véritables défis algorithmiques. Dans le cadre de cette thèse, nous nous focaliserons sur l'étude de deux entités biologiques particulières pour lesquelles de nombreux problèmes ont été proposés ces dernières années, à savoir: l'ARN et le gène (notions définies en Section 1.2).

Comme nous le verrons par la suite, les données biologiques basées sur ces notions sont facilement modélisables sous la forme de textes; ce qui en facilite l'étude et le traitement par des systèmes informatiques. Les travaux présentés dans ce manuscrit décrivent les activités de recherche menées durant mes trois années de doctorat au sein de l'équipe Combinatoire et Bio-Informatique du Laboratoire d'Informatique Nantes-Atlantique. Ces travaux de recherche ont consisté, essentiellement, à étudier du point de vue algorithmique des problèmes issus du domaine de la biologie et modélisés à l'aide de formalismes adaptés à leur étude algorithmique.

L'approche adoptée pour l'ensemble de ces problèmes a été de déterminer, si possible, des algorithmes exacts et rapides répondant aux problèmes posés. Lorsque cela ne semblait pas possible, nous avons essayé de prouver que le problème ne peut être résolu de façon rapide. Pour ce faire, nous démontrons que le problème en question est algorithmiquement difficile. Enfin, si possible, nous poursuivons l'étude de tout problème démontré comme algorithmiquement difficile en proposant, essentiellement, trois types de résultats:

1. APPROXIMATION: nous proposons un ou des algorithme(s) rapide(s) renvoyant un résultat approché mais en garantissant un écart limité entre le résultat et la valeur optimale recherchée;
2. COMPLEXITÉ PARAMÉTRÉE: nous proposons un algorithme exact lent en toute généralité mais dont on sait que la lenteur d'exécution est exponentielle en un paramètre qui reste petit en pratique;
3. HEURISTIQUE: nous proposons un algorithme rapide renvoyant un résultat approché dont on ne peut pas garantir l'écart à la valeur optimale recherchée, mais qui peut être exact dans certains cas. Ce dernier type de résultat est alors, généralement, accompagné d'une analyse, basée sur une batterie de tests, tentant d'exhiber l'efficacité de l'algorithme sur des *données réelles*.

Les problèmes biologiques étudiés ont été de deux types:

1. la comparaison des structures de molécules d'ARN et
2. le calcul de distances intergénomiques en présence de gènes dupliqués.

Dans le cadre de la comparaison de structures de molécules d'ARN, nous nous sommes intéressés au calcul de distances entre molécules d'ARN et à la recherche de structures communes à deux molécules d'ARN. Ces travaux ont constitué la plus grande partie de mon activité de recherche.

Dans le cadre du calcul de distances intergénomiques, nous nous sommes intéressés au calcul de distances évolutives prenant compte d'une spécificité des gènes encore peu considérée: *les gènes peuvent être dupliqués, et donc apparaître plusieurs fois dans le génome d'un organisme*. Cette considération n'a été que très peu étudiée car elle complique, généralement, l'étude des distances associées. Ces distances doivent permettre d'inférer l'évolution qui s'est déroulée entre les espèces et ainsi contribuer à l'établissement d'une généalogie entre les différentes espèces (également appelée *phylogénie*).

Ce manuscrit est composé de trois parties, portant respectivement sur

1. les concepts biologiques et informatiques relatifs à nos travaux,
2. la comparaison des structures de molécules d'ARN et
3. le calcul de distances intergénomiques en présence de gènes dupliqués.

La Partie I présente les différents concepts biologiques et informatiques nécessaires à la compréhension de ce manuscrit. Nous présentons, dans le Chapitre 1, les notions centrales d'ADN, ARN, protéine et gène. Nous présentons également leurs rôles, fonctions et interactions. Le Chapitre 2 est, quant à lui, consacré aux concepts algorithmiques et plus précisément aux notions d'algorithme, problème et complexité.

La Partie II de ce manuscrit porte sur nos travaux dans le cadre de la comparaison des structures de molécules d'ARN. Dans le Chapitre 3, nous proposons une présentation générale des résultats existants concernant quatre problèmes impliquant des structures de molécules d'ARN.

Dans le Chapitre 4, nous répondons à un premier problème ouvert en démontrant que le calcul de la distance d'édition entre deux structures secondaires d'ARN – représentées par des séquences construites sur l'alphabet $\Sigma = \{A, C, G, U\}$ et annotées par des arcs (également appelées *séquences arc-annotées*) – est un problème **NP**-complet.

Nous abordons ensuite, dans le Chapitre 5, le problème ARC-PRESERVING SUBSEQUENCE correspondant à la recherche d'une sous-structure (appelée motif) dans une molécule d'ARN représentées toutes deux par des séquences arc-annotées. Nous démontrons la complexité algorithmique, jusqu'alors inconnue, du sous-problème APS(CROSSING,PLAIN). Puis, nous proposons d'analyser plus finement la complexité du problème en proposant un raffinement du modèle.

Dans le Chapitre 6, nous étudions le problème de recherche de motifs de 2-intervalles – représentation macroscopique introduite par Vialette [93, 94] se focalisant sur la structure et ne tenant pas compte de la nature des constituants de la molécule d'ARN. Une nouvelle fois, nous étudions la complexité algorithmique de plusieurs sous-problèmes restés jusqu'alors ouverts.

Finalement, dans le Chapitre 7, nous présentons des résultats, dénotant un peu de ceux présentés dans les Chapitres 4, 5 et 6, et concernant le design d'ARN. Plus précisément, nous nous intéressons au problème MRSO: étant données une structure secondaire S_s et une séquence d'acides aminés S_a, trouver une séquence d'ARN satisfaisant S_s et dont la traduction produit une séquence d'acides aminés T_a telle

que la similarité (évaluée à l'aide de fonctions de score) entre S_a et T_a est maximisée. Nous proposons une étude de la complexité paramétrée du problème MRSO, ainsi qu'un raffinement du résultat de [3] prouvant la **NP**-complétude du problème.

La Partie III de ce manuscrit présente les résultats de nos travaux menés dans le cadre du calcul de distances intergénomiques en présence de gènes dupliqués. Après avoir effectué une présentation générale des résultats existants pour ce problème (Chapitre 8), nous abordons, dans le Chapitre 9, le cas du calcul de la distance de breakpoints entre deux génomes en présence de gènes dupliqués. Nous proposons deux types de résultats pour ce problème:

1. une preuve de **NP**-complétude pour le cas général et
2. un algorithme d'approximation dans un cas restreint, où les génomes considérés sont construits à partir d'un même ensemble de gènes et où il n'existe qu'une seule famille de gènes dupliqués.

Finalement, dans le Chapitre 10, nous nous intéressons à une autre distance évolutive basée sur la notion d'intervalle: la distance d'intervalles conservés. Nous prouvons, là encore, que le problème est **NP**-complet et proposons une heuristique dont nous évaluons l'efficacité sur un ensemble de données réelles composé des génomes de vingt bactéries.

Enfin, nous achevons ce manuscrit en rappelant les principaux résultats obtenus, et en présentant divers perspectives et problèmes ouverts.

PARTIE I

Concepts généraux

CHAPITRE 1

Notions de génétique

1.1 La génétique du $XIX^{ème}$ siècle à nos jours

Le terme *génétique**, du grec *genos*, est défini dans le *Petit Larousse Illustré* comme la science de l'*hérédité**, qui étudie la transmission des caractères anatomiques et fonctionnels entre les générations d'êtres vivants. Employé comme un adjectif, le terme génétique définit ce qui concerne les *gènes** et l'hérédité.

Bien avant que Wilhelm Johannsen ait défini le terme *gène* (au début des années 1900), des scientifiques de disciplines diverses ont tenté d'établir l'existence de *l'hérédité* (*i.e.* la transmission des caractères génétiques d'une génération aux suivantes). Le naturaliste français Jean-Baptiste Lamarck (1744-1829) a défini la première théorie explicative de l'évolution, nommée le *transformisme*, fondée sur l'idée de transformation progressive des populations et des lignées sous l'influence directe du milieu sur les individus. Cette théorie fut reprise par Charles Darwin (1809-1882) et Hugo De Vries (1848-1935) avec une transformation progressive due à des modifications ou mutations suivies de *sélection naturelle*. Cette théorie est mieux connue sous le terme *d'évolutionnisme*.

La théorie de *génération spontanée*, admise pendant l'antiquité et le Moyen-âge pour certains animaux, était fondée, quant à elle, sur la formation spontanée d'êtres vivants à partir de matières minérales ou de substances organiques en décomposition. C'est le célèbre chimiste et biologiste français Louis Pasteur (1822-1895) qui a prouvé l'absence de génération spontanée et a établi qu'un être vivant possède au moins un ancêtre dont il a hérité ses caractéristiques.

C'est également au début du $XIX^{ème}$ siècle que le moine botaniste tchèque Gregor Mendel (1822-1884), considéré comme pionnier de la génétique, a réalisé des expériences sur l'hybridation des plantes et a énoncé, en 1866, les lois de la transmission des caractères héréditaires (lois de Mendel). C'est en observant la transmission des caractéristiques morphologiques de pois à travers quelques générations, qu'il a défini les termes de *phénotype** et *génotype** et a énoncé les lois dites de Mendel, base de la génétique moderne. Mendel a travaillé sur des pois comestibles présentant sept caractères spécifiques illustrés dans la Figure 1.1. À partir de ses expériences, il a publié l'article "*Recherche sur les hybrides végétaux*" [77] où il a énoncé les lois de transmission de certains caractères héréditaires. Malheureusement, Mendel n'eut qu'une reconnaissance post mortem. En effet, le bien-fondé de cet article, publié en 1866, ne fut reconnu qu'en 1907.

C'est vers la fin du $XIX^{ème}$ siècle (1880), que les biologistes allemands Oskar Hertwig (1849-1922) et Eduard Strasburger (1844-1912) ont découvert que le noyau des cellules est le siège de l'hérédité. Ce sont les progrès techniques qui ont permis peu à peu de définir la notion de *gène* (définie par Johannsen vers 1900 mais dérivée du terme *pagen* de De Vries, lui même dérivé de la notion de *pangenesis* de Darwin) et ceux de la microscopie de localiser le support des gènes: le *chromosome**.

La théorie chromosomique de l'hérédité fut proposée par le cytologiste Walter Stanborough Sutton (1877-1916) en 1902. Sutton s'appuya sur une étude morphologique des chromosomes de la sauterelle

Caractère	Valeurs possibles	
Forme des graines	Lisse	Ridée
Couleur des cotylédons	Jaune	Vert intense
Coloration des enveloppes des graines	Blanche	Grise
Forme des gousses mûres	Rectiligne	Irrégulière
Couleur des gousses non parvenues à maturité	Verte	Jaune vif
Position des fleurs	Axiale	Terminale
Longueur des tiges	Longue	Courte

Figure 1.1 – Les sept caractères des pois étudiés par Mendel.

Brachystola magna et suggéra que les chromosomes allaient par paires et pouvaient être le support de l'hérédité. Il suggéra que les caractéristiques de Mendel étaient situées sur les chromosomes et qu'une copie de chaque chromosome était héritée d'un parent durant la *méiose** (*i.e.* la division de la cellule aboutissant à la réduction de moitié du nombre de chromosomes).

C'est en 1910, que le premier gène lié au sexe est découvert par le biologiste Thomas Hunt Morgan (1866-1945). En effet, en étudiant les *drosophiles* sauvages – mouches ayant pour caractéristique d'avoir des yeux rouges –, Morgan a découvert une drosophile *mutante* aux yeux blancs. Cette découverte démontra qu'à la suite d'une *mutation*, une modification d'un caractère héréditaire pouvait subvenir. Ces résultats ont étayé les théories de Mendel et ont permis à Morgan, en établissant la théorie de l'hérédité liée au sexe, de consolider la théorie chromosomique de l'hérédité de Sutton.

Ce n'est qu'en 1929, que l'*Acide DésoxyriboNucléique** (ADN) est découvert par le chimiste Phoebus Aaron Levene (1869-1940). En 1944, le bactériologiste et physicien Oswald Avery (1877-1955) a démontré, avec Colin Mc Leod (1909-1972) et Mc Lyn McCarthy (1911) que l'ADN est une molécule transportant une information héréditaire. En 1953, les biologistes James Watson (1928) et Francis Crick (1916-2004) ont proposé le modèle à double hélice de l'ADN, expliquant ainsi que l'information génétique puisse être portée par cette molécule.

C'est en 1957 que Crick et le physicien nucléaire George Gamov (1904-1968) ont défini le *dogme central de la biologie moléculaire** (*cf.* Section 1.2) qui explique les mécanismes permettant de passer de l'ADN aux protéines. François Jacob (1920), André Lwoff (1902-1994) et Jacques Monod (1910-1976) ont, pour leur part, montré:

1. comment l'ADN se structure en *codon** pour programmer la synthèse de *protéines** à partir d'*acides aminés**;
2. le mécanisme des mutations;
3. la présence d'un code de fin de lecture.

Dès lors, la recherche sur les mécanismes de synthèse de protéines, et sur l'étude du code génétique est devenue un objet d'intérêts économiques importants. Cela a eu pour conséquence une accélération des découvertes dans ces domaines, qui ont conduit, entre autres, au décodage (appelé également *séquençage**) des trois milliards de paires de bases du génome humain: le *Human Genome Project**[1].

C'est le 14 avril 2003, près d'un siècle et demi après les travaux de Mendel, que la fin du séquençage du génome humain fut annoncée. Il reste alors aux scientifiques le travail rigoureux consistant à:

1. rechercher de nouveaux gènes;
2. déterminer la fonction de ces derniers;

[1] $http://www.ornl.gov/sci/techresources/Human\_Genome/home.shtml$

3. affiner le séquencage qui peut contenir des erreurs.

Comme nous l'avons vu dans ce court historique, il existe plusieurs domaines de recherche dans la génétique:

- l'*hérédité*, qui étudie le phénotype et tente de déterminer le génotype sous-jacent en se basant toujours sur les lois de Mendel;
- la *biologie cellulaire* et la *biologie moléculaire* qui étudient les gènes et leur support matériel (ADN ou ARN) au sein de la cellule;
- le *génie génétique* (ou ingénierie de la génétique) qui étudie la modification du génome, l'implantation, la suppression ou la modification de nouveaux gènes (les Organismes Génétiquement Modifiés) dans des organismes vivants;
- la *thérapie génique* qui étudie l'introduction de nouveaux gènes dans l'organisme afin de pallier une déficience héréditaire.

Les travaux exposés dans ce manuscrit appartenant essentiellement au domaine de la biologie moléculaire, nous allons préciser les notions citées précédemment, à savoir l'ADN, l'ARN et les protéines. Ces notions seront présentées dans leur ordre d'utilisation dans le déroulement des mécanismes du dogme central de la biologie moléculaire.

1.2 Le dogme central de la biologie moléculaire

Dans cette section, nous présentons le dogme central de la biologie moléculaire qui n'est autre que le modèle schématique de la conservation et de l'utilisation de l'information génétique. L'un des acteurs principaux de la transmission de l'information génétique est l'ADN. L'ADN est le support stable et transmissible de l'information génétique qui définit les fonctions biologiques d'un organisme. Lors de la *méiose*, l'ADN se réplique. Il est alors transcrit en Acide RiboNucléique (ARN). L'ARN peut alors être traduit en protéine à l'aide d'une entité biologique nommée *ribosome**. On parle alors de *traduction** de l'ARN en protéine. Le dogme central (*cf.* Figure 1.2) se résume ainsi: l'ADN se réplique, se transcrit en ARN qui est éventuellement traduit en protéine.

Comme nous l'avons précisé précédemment, Mendel a défini en 1866 les lois de base de l'hérédité. Ce n'est que dans le début des années 1900, que le concept de gène fut défini comme unité d'hérédité contrôlant un caractère particulier. Entre l'énoncé des lois de Mendel et la découverte du rôle biologique de l'ADN, il s'est déroulé un peu moins d'un siècle. De nos jours, l'ADN, qui a la faculté de se reproduire et d'être transmis aux descendants lors des processus de reproduction des organismes vivants, est considéré comme le support de l'hérédité.

1.2.1 Les gènes

Un gène désigne une unité d'information génétique transmise d'une génération à une autre. L'ensemble des gènes d'un individu constitue son génome. Un gène est une unité d'information génétique à l'origine de la synthèse des protéines (*cf.* Section 1.2.4). Un gène est défini, généralement, comme un enchaînement de *nucléotides**, *i.e.* une portion d'ADN (*cf.* Section 1.2.2), situé à un endroit bien précis (appelé *locus**) sur un chromosome et porteur d'une information génétique (*cf.* Figure 1.3)[2]. Comme l'illustre la Figure 1.3, un gène est constitué de sous-séquences de nucléotides alternativement porteuses et non porteuses d'informations génétiques, appelées respectivement *exons** et *introns** (pour plus de détails, se référer à la fin de la Section 1.2.4).

[2] Il existe plusieurs définitions du terme gène. Dans ce manuscrit nous utiliserons celle définie ici.

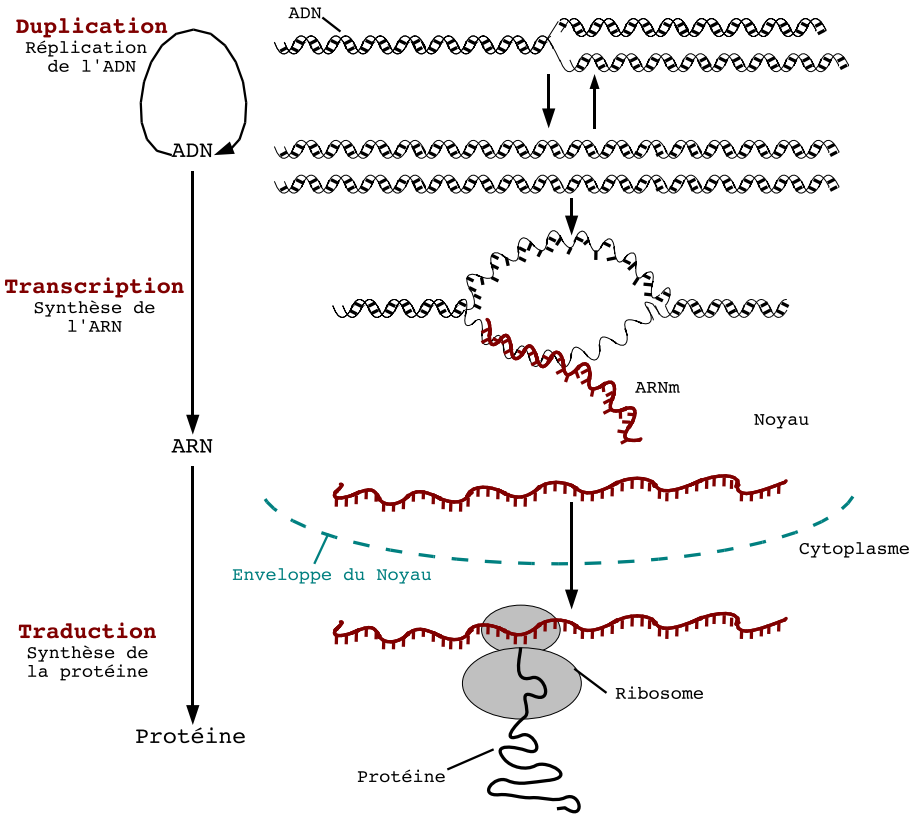


Figure 1.2 – Le Dogme Central de la Biologie Moléculaire.

1.2.2 L'Acide DésoxyriboNucléique

L'ADN, acronyme d'*acide désoxyribonucléique*, est une longue molécule présente dans tous les organismes vivants. L'ADN est présent dans le noyau des cellules *eucaryotes*⋆, dans les cellules *procaryotes*⋆, les *mitochondries*⋆ ainsi que les *chloroplastes*⋆. La structure de l'ADN se présente sous la forme d'une double hélice. Chaque hélice est constituée d'un enchaînement d'unités de construction appelées

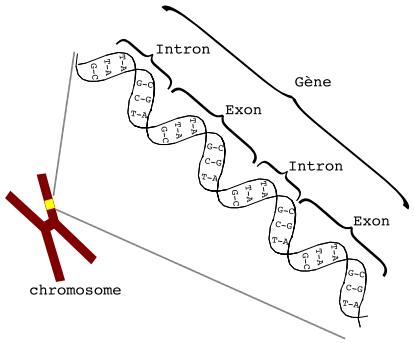


Figure 1.3 – Gène et chromosome.

nucléotides. Un nucléotide est composé de trois substances fondamentales (*cf.* Figure 1.4): (1) un *groupe phosphate**, (2) un sucre à cinq atomes de carbone, le *désoxyribose** et (3) une *base azotée**.

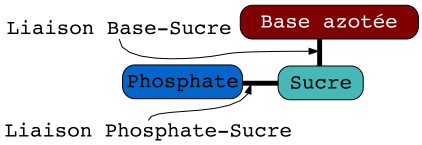


Figure 1.4 – Vue schématique d'un nucléotide.

Les bases azotées entrant dans la composition des nucléotides sont séparées en deux familles: les *purines* et les *pyrimidines*, comme illustré dans la Figure 1.5 (b).

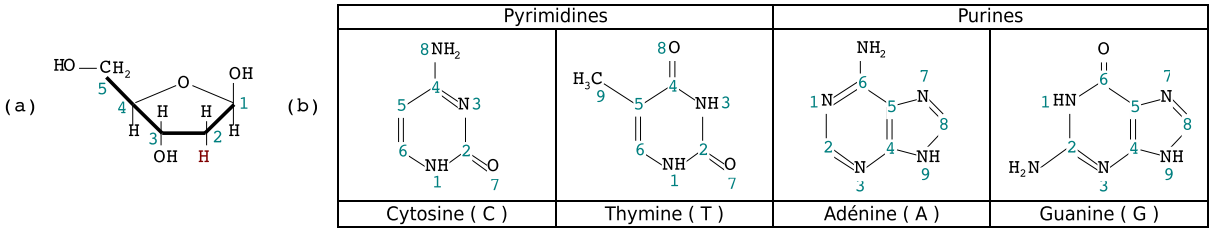


Figure 1.5 – (a) le désoxyribose et (b) les bases azotées. Les chiffres indiquent des positions d'atomes: par exemple, la position 7 de la base azotée de Cytosine est un atome d'oxygène.

La famille des purines est constituée des bases azotées d'*Adénine** (A) et de *Guanine** (G), tandis que la famille des pyrimidines est constituée des bases azotées de *Cytosine** (C) et de *Thymine** (T). Les phosphates et les sucres étant identiques pour tous les nucléotides, c'est la base azotée contenue dans le nucléotide qui va déterminer sa nature. Les bases azotées sont dites complémentaires: l'Adénine s'associant toujours avec la Thymine (liaison A-T) et la Guanine s'associant toujours avec la Cytosine (liaison G-C). La Figure 1.6 illustre l'assemblage du groupement phosphate, du désoxyribose et de la base azotée.

La stabilité de la molécule d'ADN est due à des liaisons chimiques. Ces liaisons ont la particularité d'être assez fortes pour maintenir la stabilité de la molécule, mais également assez faibles pour faciliter la manipulation de l'ADN par diverses entités biologiques lors des processus de transcription et de traduction. Les liaisons chimiques de l'ADN sont de trois types: *phosphodiester*, *covalentes* et *hydrogènes*.

Une liaison phosphodiester est une liaison dans laquelle un groupe phosphate (*i.e.* l'association d'un phosphate et de quatre atomes d'oxygène) est lié à un groupement *hydroxyle* (OH) du sucre. Le désoxyribose est composé de trois groupements hydroxyles: un est utilisé pour lier le désoxyribose à la base azotée, les deux autres sont utilisés pour lier les nucléotides entre eux. Dans les acides nucléiques, les différents nucléotides sont placés bout à bout et liés les uns aux autres par des liens $3'$- $5'$ phosphodiester: le phosphate se lie aux groupements hydroxyles en positions[3] $3'$ et $5'$. Le phosphate est donc le lien (ou le pont) entre chaque sucre comme illustré en Figure 1.7.

En chimie, une liaison covalente est une liaison chimique dans laquelle chacun des atomes liés met en commun un ou plusieurs électrons. C'est ce type de liaison qui lie la base azotée et le sucre.

[3] Par convention, pour ne pas confondre les positions des molécules de la base azotée avec celles du sucre, la $i^{ème}$ position du sucre est renommée en i' pour tout i.

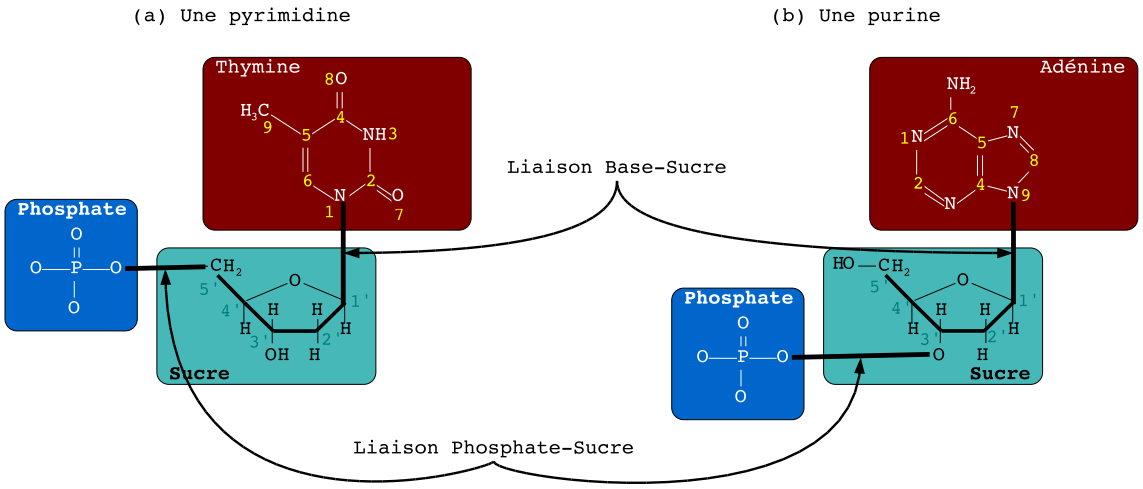


Figure 1.6 – Assemblage d'un nucléotide de la famille (a) des pyrimidines (la Thymine) ou (b) des purines (l'Adénine) .

L'alternance des phosphates et des sucres produit le *squelette* de l'acide nucléique sur lequel s'attachent les bases azotées (communément appelé *brin*). Le squelette est une partie relativement rigide de la molécule puisqu'il est composé de liens chimiques très forts.

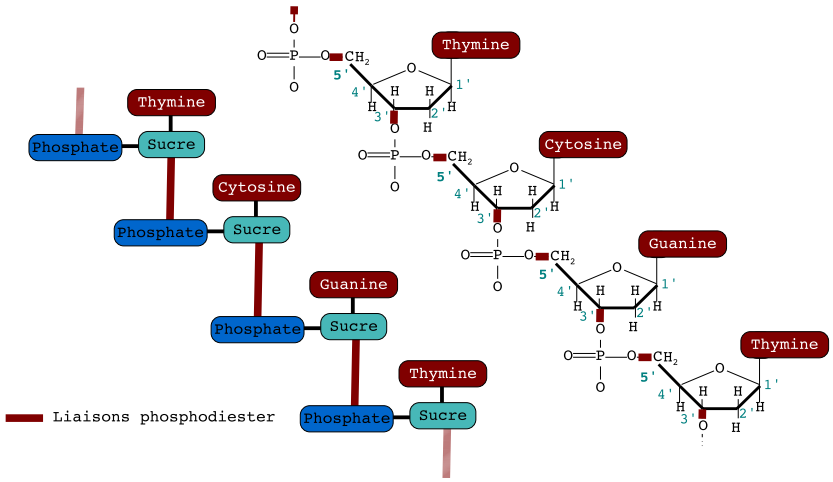


Figure 1.7 – Les liaisons phosphodiester dans la composition d'un brin d'ADN.

Dans le cas de l'ADN, les deux brins sont disposés de telle sorte que toutes les bases azotées se retrouvent au centre de la structure. Cette structure, appelée double hélice, est maintenue par des liaisons dites *hydrogènes* (liens chimiques faibles) qui se forment entre les bases azotées complémentaires: l'Adénine s'associant avec la Thymine à l'aide de deux liens hydrogènes et la Guanine s'associant avec la Cytosine à l'aide de trois liens hydrogènes, comme illustré par la Figure 1.8. Les deux brins ont une structure hélicoïdale rendue possible grâce à la souplesse des liens hydrogènes. C'est cette souplesse qui

permettra la réplication de l'ADN et sa transcription en ARN.

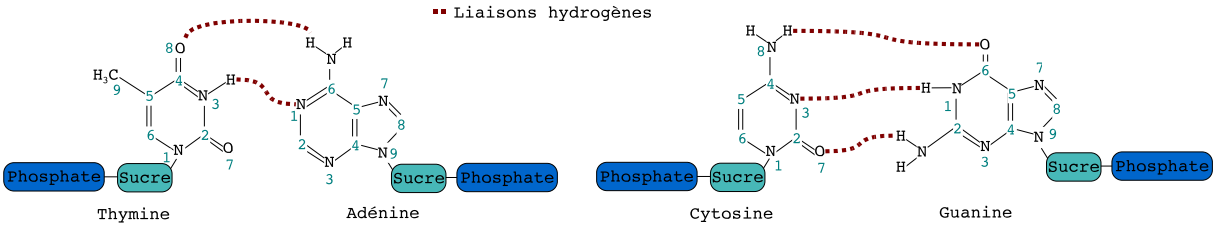


Figure 1.8 – Les liaisons hydrogènes entre bases azotées complémentaires.

L'ADN, comme nous l'avons déjà précisé, contient l'information génétique. En se dupliquant (processus détaillé ci-après et illustré en Figure 1.9), l'ADN assure la préservation de cette information.

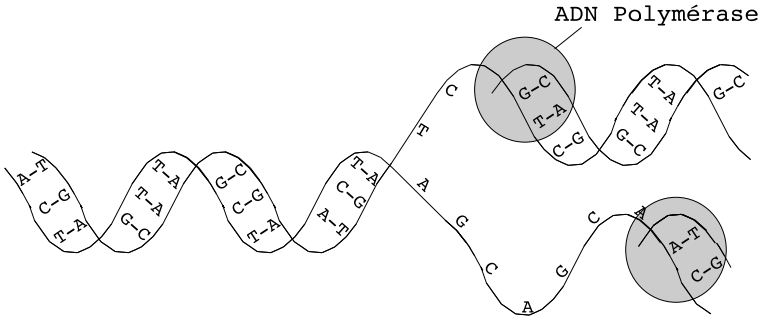


Figure 1.9 – Le processus de réplication de l'ADN.

L'ADN a l'importante propriété de se reproduire à l'identique, ce qui permet à l'information de se transmettre d'une cellule mère aux cellules filles. Dans un premier temps, la molécule d'ADN s'ouvre telle une fermeture éclair - par l'action de l'*ADN polymérase*⋆, qui casse les liens hydrogènes - libérant ainsi deux brins complémentaires. Dans un second temps, l'ADN polymérase va fixer des nucléotides disponibles dans le noyau de la cellule sur les bases complémentaires d'un des brins libérés. Deux nouvelles molécules d'ADN vont ainsi être construites, composée chacune d'un brin de l'ancienne molécule et d'un brin nouvellement formé. On dit que la *réplication*⋆ se fait suivant un mode semi-conservatif (*i.e.* chaque molécule d'ADN fille hérite d'un brin de l'ADN mère).

Ce sont des milliers d'enzymes ADN polymérases qui, opérant ainsi tout au long des molécules d'ADN contenues dans le noyau des cellules, vont permettre la préservation de l'information génétique.

1.2.3 L'Acide RiboNucléique

L'ARN, acronyme d'*acide ribonucléique*, est un acide nucléique résultant de la transcription de l'ADN. On peut dénombrer cinq différences essentielles entre l'ARN et l'ADN:

1. le sucre composant les nucléotides de l'ARN est le *ribose* (illustré en Figure 1.10 (a));
2. les bases azotées composant les nucléotides de l'ARN sont l'Adénine, la Cytosine, la Guanine et l'*Uracile*⋆ (*U*) (illustré en Figure 1.10 (b));

3. l'ARN est simple brin, à quelques exceptions près;
4. l'ARN est court (au plus quelques milliers de nucléotides et non pas des millions comme dans l'ADN);
5. l'ARN est produit en masse et ceci pour une utilisation de courte durée (l'ADN doit, pour sa part, conserver une information longtemps mais n'est produit que rarement).

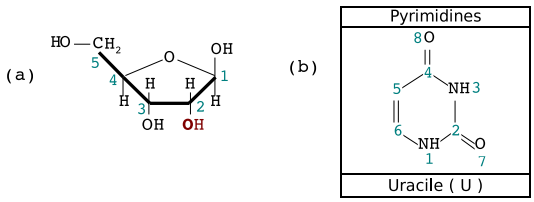


Figure 1.10 – Le ribose et la base azotée d'Uracile

La séquence d'une molécule d'ARN code deux types d'informations: (1) celles nécessaires au codage de la fonction d'une protéine ou (2) celles nécessaires au codage de sa propre fonction. Alors que dans le premier cas, seule la séquence semble avoir une importance, dans le second cas, il est avéré que c'est la structure de l'ARN qui détermine les différentes interactions que la molécule pourra avoir avec d'autres molécules.

La stabilité de la molécule d'ARN est due aux même liaisons chimiques que l'ADN. Mais étant généralement simple brin, la molécule d'ARN a tendance à se replier sur elle-même pour former des liaisons hydrogènes entre des couples de ses propres nucléotides. Ces liaisons correspondent très souvent à des interactions entre bases complémentaires comme celles de l'ADN mais où la Thymine est remplacée par l'Uracile (*i.e.* A-U et C-G). Il existe, en nombre moindre, d'autres interactions parmi lesquelles on peut citer les interactions de type *Wobble* entre l'Adenine et la Cytosine ou la Guanine et l'Uracile, comme illustré en Figure 1.11.

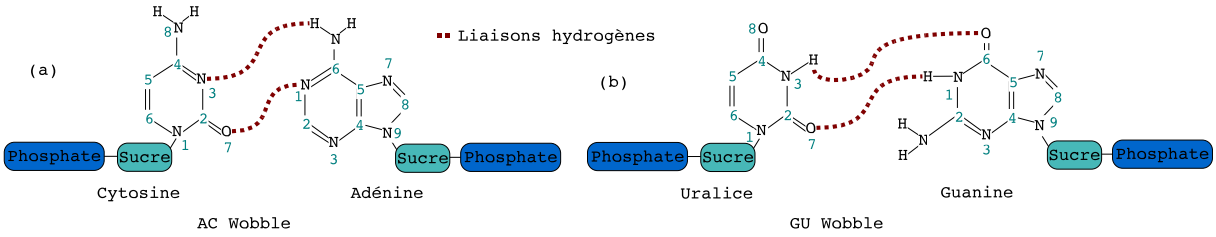


Figure 1.11 – Les interactions (a) A-C et (b) G-U de type Wobble.

On dénombre généralement trois niveaux de structures pour une molécule d'ARN. Hormis la structure dite *primaire* de l'ARN, également appelée *séquence*, qui désigne la chaîne de nucléotides, on dénombre deux autres niveaux de structures, dites *secondaire* et *tertiaire*.

La structure secondaire d'une molécule d'ARN désigne le repliement planaire (*i.e.* que l'on peut dessiner sur le plan sans croisements) de la structure primaire, qui met en évidence les liaisons covalentes et la plupart des interactions entre les nucléotides complémentaires. Un exemple de structure secondaire d'une molécule d'ARN est donné en Figure 1.12. Ce repliement peut se décrire à l'aide des éléments structuraux suivants:

- *région simple brin*: une suite de nucléotides non appariées;
- *hélice*: une suite continue de liaisons entre nucléotides complémentaires;
- *boucle terminale*: une région simple brin formant une boucle à l'extrémité d'une hélice;
- *boucle interne*: deux hélices distinctes reliées à l'aide de deux régions simple brin;
- *boucle multiple*: au moins trois hélices distinctes reliées à l'aide de régions simple brin;
- *renflement*: boucle interne dont l'une des régions simple brin est de longueur nulle;
- *les extrémités terminales*: les régions simple brin présentes en début et fin de la séquences de nucléotides, représentées respectivement par $5'$ et $3'$.

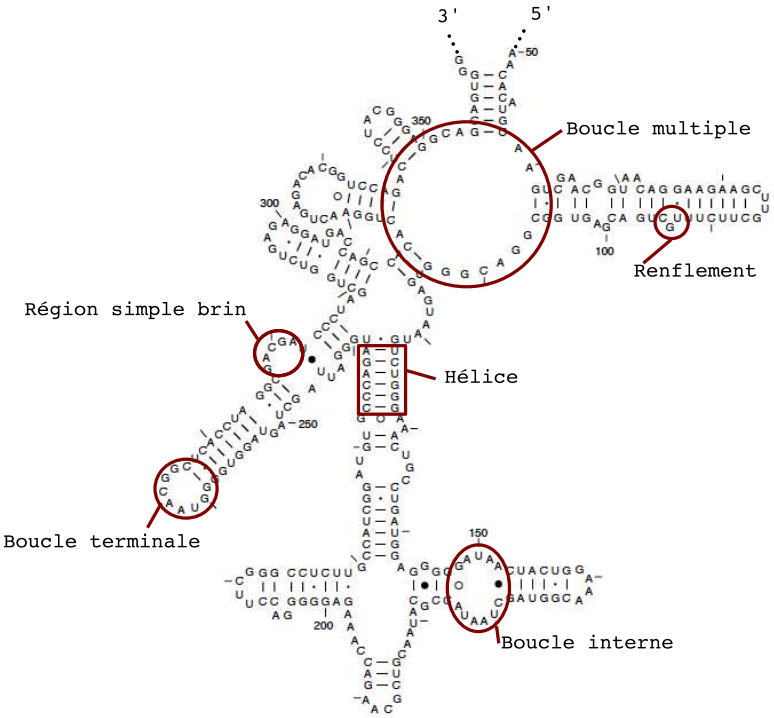


Figure 1.12 – Structure secondaire de l'ARN ribosomal 16s d'Escherichia Coli [60] (vue partielle).

Contrairement à la structure secondaire qui peut toujours être représentée en deux dimensions, la structure tertiaire d'une molécule d'ARN fait explicitement référence à un repliement dans un espace à trois dimensions. Ce dernier niveau de représentation de la molécule d'ARN est plus proche de la réalité puisqu'il décrit l'organisation spatiale de la molécule au niveau atomique et contient des interactions supplémentaires par rapport à la structure secondaire.

Le *pseudo-noeud* fait partie de ces interactions supplémentaires. Il existe plusieurs définitions d'un pseudo-noeud [82]. Dans ce manuscrit, nous considérerons la suivante: soient i, j, k, l les positions, par rapport à l'extrémité terminale $5'$, de quatre bases de la structure primaire telles que $i < j < k < l$. Un pseudo-noeud est formé lors de la présence d'un lien hydrogène entre les $i^{ème}$ et $k^{ème}$ bases et un lien hydrogène entre les $j^{ème}$ et $l^{ème}$ bases. Une illustration d'un pseudo-noeud est donné en Figure 1.13.

En plus des quelques différences précisées précédemment entre l'ADN et l'ARN, l'ARN se distingue de l'ADN par son rôle essentiel de *messager de l'information génétique*. En effet, comme nous allons le voir dans la prochaine section, l'ARN est un intermédiaire entre l'ADN et les protéines. Il existe

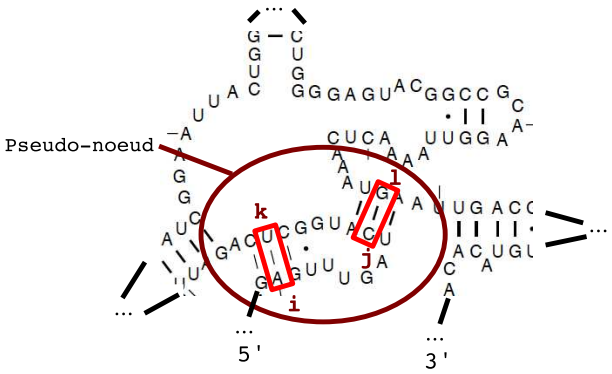


Figure 1.13 – Structure tertiaire d'une molécule d'ARN: un pseudo-noeud de l'ARN ribosomal 16s d'Escherichia Coli [60] (vue partielle).

différents types d'ARN: *ARN messager*⋆ (ARNm), *ARN de transfert*⋆ (ARNt), *ARN ribosomal*⋆ (ARNr), *micro ARN*⋆, *etc*. Comme nous le verrons, certains de ces types d'ARN ont un rôle particulier dans le processus complexe de synthèse des protéines.

1.2.4 Les protéines

Les protéines sont les molécules les plus complexes et les plus variées des êtres vivants. On estime que tout être humain fabrique à peu près 100 000 types de protéines différentes, chaque cellule en fabriquant en moyenne 15 000 types différents. Le terme *protéine* désigne une longue molécule formée d'*acides aminés*⋆. Le terme acide aminé désigne une molécule formée d'un atome de carbone auquel sont liés (1) un groupement *amine* (NH_2), (2) un groupement *acide* ($COOH$) et (3) une portion variable d'un acide aminé à l'autre (*cf.* Figure 1.14).

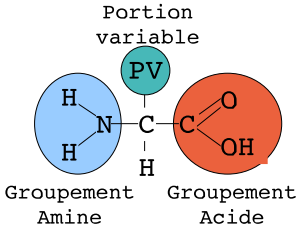


Figure 1.14 – Composition d'un acide aminé.

C'est cette portion variable qui permet de différencier les 22 acides aminés existant[4]. Il existe deux notations (à une ou trois lettres) identifiant les acides aminés. La *Phenylalanine*, par exemple, peut s'écrire Phe ou F (*cf.* le code génétique illustré en Figure 1.19).

Du point de vue chimique, une protéine est un enchaînement formé par un nombre restreint d'acides aminés liés par des *liaisons peptidiques*. Une *liaison peptidique*⋆ s'effectue entre le groupement acide

[4]Comme nous le verrons dans le Chapitre 7, cela ne fait que quelques années que ce nombre est passé de 20 à 22 avec la découverte, vers la fi n des années 90, de la Selenocystéine et, en 2002, de la Pyrrolysine.

($COOH$) d'un acide aminé et le groupement amine (NH_2) d'un autre (*cf.* Figure 1.15). Au cours de la réaction, une molécule d'eau est éliminée. L'union de plusieurs acides aminés forme ce que l'on appelle un *polypeptide**. La taille des polypeptides peut varier de 5 à plus de 500 acides aminés. En moyenne, une protéine est un polypeptide de 100 à 200 acides aminés.

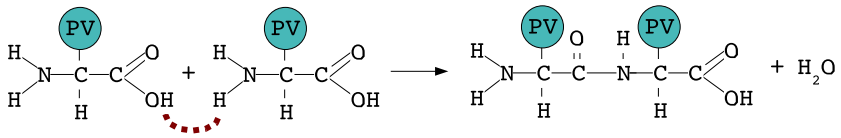


Figure 1.15 – La formation d'une liaison peptidique.

Tout comme l'ADN et l'ARN, la protéine possède une structure dans l'espace. Cette structure peut être classée en trois niveaux:

1. la *structure primaire*: la séquence des acides aminés dans la protéine;
2. la *structure secondaire*: sa conformation spatiale locale. On dénombre deux grands types de structures secondaires:
   - *hélice alpha**: un segment de la séquence d'acides aminés prenant la forme d'une hélice;
   - *feuillet beta**: un ensemble de segments de la séquence d'acides aminés disposés parallèlement et formant une sorte de feuillet plissé.

   Une protéine est donc faite d'hélices alpha et de feuillets bêta reliés par des segments de la séquence d'acides aminés qui n'ont pas de structure secondaire particulière;
3. *la structure tertiaire*: le repliement de la protéine dans l'espace tridimensionnel. Cette structure rend compte de l'organisation entre eux des éléments de la structure secondaire qui forment un ensemble compact.

Les protéines sont synthétisées par un processus à partir de l'information contenue dans l'ADN. La synthèse des protéines est décomposable en deux étapes: la *transcription** de l'ADN en ARN messager (*cf.* Figure 1.17) et la *traduction* de l'ARNm en protéine (*cf.* Figure 1.18). Suivant la nature de la cellule (eucaryote ou procaryote), ces étapes peuvent se dérouler en parallèle.

En effet, chez les eucaryotes, ces deux étapes sont successives car elles se déroulent dans des lieux différents de la cellule. La transcription se déroule dans le noyau de la cellule, la traduction, dans le cytoplasme. Alors que chez les procaryotes, les deux étapes ont lieu dans le *cytoplasme** et peuvent donc être exécutées simultanément, la traduction débutant alors que la transcription n'est pas encore achevée. Le lieu de la transcription n'étant pas identique chez les eucaryotes et les procaryotes, le traitement de la molécule d'ARN synthétisée dépendra de la nature de la cellule.

La transcription permet de préserver l'ADN des détériorations causées par trop de manipulations, mais également d'accélérer, par le nombre de copies produites, la synthèse d'une protéine. Succinctement, on peut voir la transcription comme le processus qui transcrit un gène de l'ADN en ARNm. Avant de définir plus clairement ce processus, rappelons qu'un gène est défini comme une portion d'ADN, destinée à être transcrite en ARN. Généralement, la séquence de nucléotides d'un gène commence par une sous-séquence particulière, appelée *promoteur**. C'est le promoteur qui va indiquer où la transcription de l'ADN en ARN doit débuter. C'est de nouveau une sous-séquence particulière du gène qui indiquera la fin de la transcription.

Plus en détails, la transcription nécessite la présence d'une entité biologique appelée *l'ARN polymérase*. L'*ARN polymérase** va, dans un premier temps, se fixer au promoteur du gène qu'il doit transcrire.

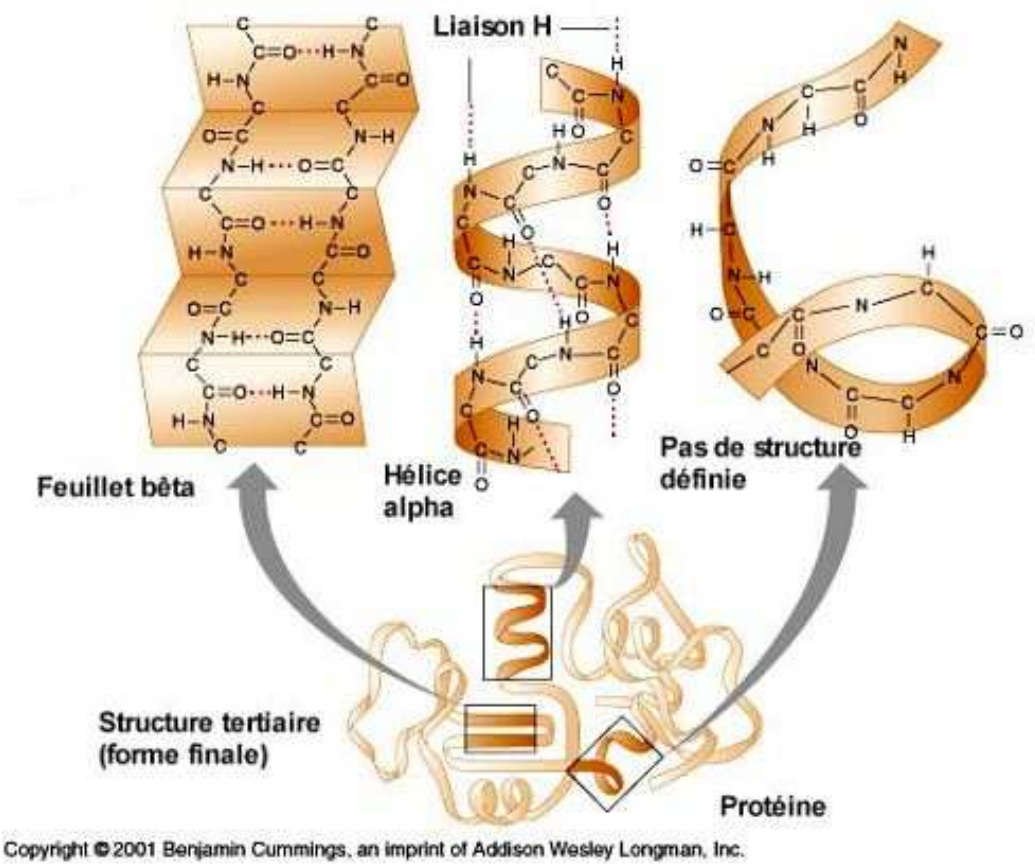


Figure 1.16 – Structure tertiaire d'une protéine. La structure tertiaire d'une protéine rend compte de l'organisation des éléments de la structure secondaire (*i.e.* les hélices alpha et les feuillets bêta) et des segments qui n'ont pas de structure secondaire particulière.

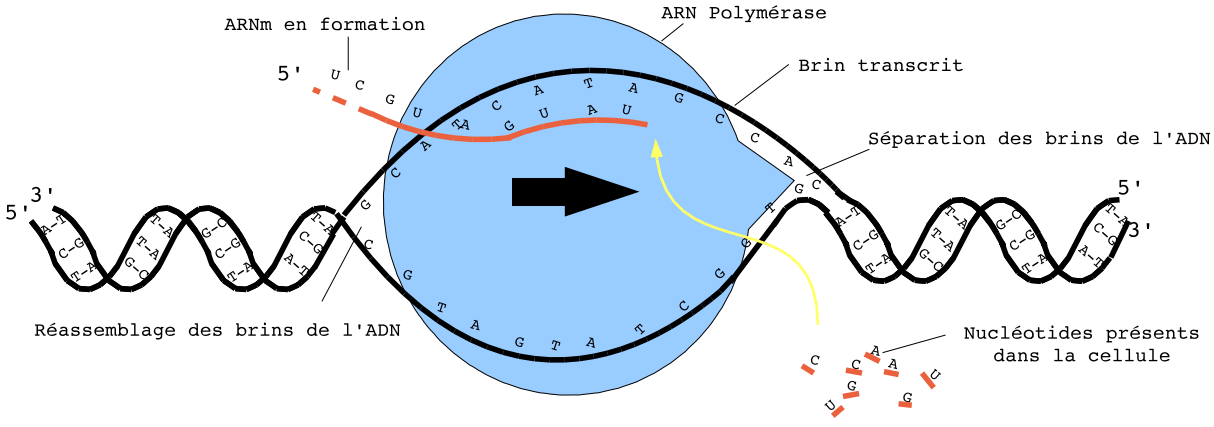


Figure 1.17 – Transcription de l'ADN en ARN messager.

Dans un deuxième temps, l'ARN polymérase va "dérouler" la molécule d'ADN, séparer les deux brins, puis assembler des bases azotées présentes dans la cellule en se servant du brin complémentaire comme négatif (*i.e.* en utilisant les règles de complémentarité des bases de l'ARN) pour aboutir à la molécule d'ARN (*cf.* Figure 1.17). Les deux brins d'ADN se réassemblent après le passage de l'ARN polymérase. La transcription s'achève lorsque l'ARN polymérase rencontre la séquence de fin de transcription, par la séparation de l'ADN et de l'ARN polymérase. Finalement, l'ARN transcrit, que l'on nomme *transcrit primaire**, est libéré de l'ARN polymérase.

Chez les procaryotes, le transcrit primaire peut être utilisé directement. En revanche, chez les eucaryotes il se compose de séquences codantes appelées exons, interrompues par des séquences non codantes appelées introns. Le transcrit primaire reste ainsi dans le noyau de la cellule jusqu'à ce qu'il ait été débarrassé de tous ses introns. Ce processus est appelé l'*epissage**. La molécule d'ARN ainsi traitée va pouvoir être exportée vers le cytoplasme et devenir un ARNm.

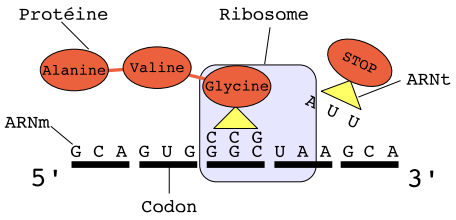


Figure 1.18 – Traduction de l'ARNm en protéine.

La traduction intervient lorsque le brin d'ARNm a atteint le cytoplasme. Dans un premier temps, un ribosome se fixe sur l'ARNm. Le ribosome va produire une séquence d'acides aminés en parcourant le brin d'ARNm codon par codon (un codon est un groupe de trois nucléotides de l'ARNm, correspondant soit à un acide aminé soit à un signal de fin de transcription). Pour ce faire, le ribosome va faire appel à l'ARNt qui, en fixant un acide aminé et en le véhiculant jusqu'à l'ARNm, va faire le lien entre les acides aminés et le message porté par l'ARNm. Le ribosome va donc parcourir le brin d'ARNm codon par codon et va, par l'intermédiaire d'un ARNt, ajouter un acide aminé à la protéine en cours de fabrication selon le codon lu. La traduction s'achève lorsque le ribosome rencontre un codon-stop, par la séparation de la protéine, du ribosome et du brin d'ARNm. Avant d'être détruit, un ARNm peut servir à la fabrication, simultanée ou non, de 10 à 20 protéines. Le code génétique utilisé pour le décodage des codons est illustré en Figure 1.19.

1ère position	2ème position U	2ème position C	2ème position A	2ème position G	3ème position
U	Phe	Ser	Tyr	Cys	U
					C
	Leu		STOP	STOP/Sec	A
			STOP/Pyr	Trp	G
C	Leu	Pro	His	Arg	U
					C
			Gln		A
					G
A	Ile	Thr	Asn	Ser	U
					C
			Lys	Arg	A
	Met				G
G	Val	Ala	Asp	Gly	U
					C
			Glu		A
					G

Code 1 lettre	Acide Aminé	Code 3 lettres	Code 1 lettre	Acide Aminé	Code 3 lettres
A	Alanine	Ala	O	Asparagine	
B	Alanine		P	Proline	Pro
C	Cystéine	Cys	Q	Glutamine	Glu
D	Acide aspartique	Asp	R	Arginine	Arg
E	Acide glutamique	Glu	S	Serine	Ser
F	Phénylalanine	Phe	T	Thréonine	Thr
G	Glycine	Gly	U	Thréonine	
H	Histidine	His	V	Valine	Val
I	Isoleucine	Ile	W	Tryptophane	Trp
J	Isoleucine		X	Valine	
K	Lysine	Lys	Y	Tyrosine	Tyr
L	Leucine	Leu	Z	Tyrosine	
M	Méthionine	Met	?	Selenocystéine	Sec
N	Asparagine	Asp	?	Pyrrolysine	Pyr

Figure 1.19 – Le code génétique.

CHAPITRE 2

Complexité et algorithmique

2.1 Différences entre problème et algorithme

On appelle *théorie de la complexité** l'étude formelle de l'efficacité des algorithmes ainsi que de la difficulté inhérente aux problèmes. Un *problème* est formalisé de la manière suivante: étant donné un ensemble de données $\mathcal{D}$ en entrée, un problème est une question posée sur $\mathcal{D}$ et dont la réponse peut éventuellement demander des calculs. Étant donné un problème $\mathcal{P}$, on note $R_\mathcal{P}$ l'ensemble des réponses (également appelées solutions) possibles à la question posée par $\mathcal{P}$. On appelle *instance* de $\mathcal{P}$, tout élément $d \in \mathcal{D}$.

Figure 2.1 – Muhammad Al-Khwarizmi

On définit généralement deux catégories de problèmes en fonction du type de réponses retournées: *les problèmes de décision* et *les problèmes d'optimisation*. Un problème de décision est un problème $\mathcal{P}$ pour lequel $R_\mathcal{P} = \{oui, non\}$ (on parle également de *problème de décision binaire*). Un problème d'optimisation est un problème $\mathcal{P}$ pour lequel la question consiste à rechercher parmi les éléments de $R_\mathcal{P}$ une solution maximisant une fonction f donnée.

La théorie de la complexité a été définie exclusivement pour les problèmes de décision. Pour autant, elle peut facilement s'étendre aux problèmes d'optimisation. En effet, à tout problème d'optimisation $\mathcal{P}_1$ sur un ensemble de données $\mathcal{D}$, on peut associer un problème de décision P_2 sur ce même ensemble de données. Pour ce faire, il suffit, par exemple, de définir la question de P_2 comme suit: "Existe-t-il une instance $d \in \mathcal{D}$ telle que $f(d) \geq b$?", avec $b \in R_\mathcal{P}$. Si on sait résoudre le problème d'optimisation alors on sait également résoudre le problème de décision correspondant. Réciproquement, on peut résoudre un problème d'optimisation si on sait résoudre le problème de décision correspondant. Il suffit, alors de faire varier la valeur de b jusqu'à ce que la réponse de $\mathcal{P}_2$ soit non. À titre d'exemple, rechercher à minimiser une valeur v revient à traiter le problème de décision consistant à comparer $f(v)$ au plus petit seuil possible b. La valeur optimale est déterminée en traitant dans l'ordre décroissant plusieurs valeurs de b. L'exemple qui suit illustre la correspondance entre la version décisionnelle et la version d'optimisation d'un problème.

Exemple 2.1.

LE VOYAGEUR DE COMMERCE
DONNÉES: Un ensemble de villes et les distances séparant les villes deux à deux.
QUESTION: Un voyageur de commerce doit passer exactement une et une seule fois dans toutes les villes. Soit $\mathcal{T}$ l'ensemble des trajets respectant cette condition. Notons $long(t)$ la longueur du trajet $t \in \mathcal{T}$.
VERSION OPTIMISATION: Quel est le trajet $t \in \mathcal{T}$ tel que $long(t)$ soit le plus petit possible ?
VERSION DÉCISION: Existe-il un trajet $t \in \mathcal{T}$ tel que $long(t)$ ne dépasse pas b km ?

La résolution d'un problème passe par la définition d'un *algorithme**. Un algorithme est une méthode de résolution de problème énoncée sous la forme d'une série d'opérations à effectuer. Un algorithme, terme latinisé du nom du mathématicien persan *Al-Khwarizmi* (780-850) (*cf.* Figure 2.1), est une suite finie séquentielle de règles que l'on applique à un nombre fini de données et permettant de résoudre des problèmes. L'algorithme du démarrage en monocycle est donné à titre d'exemple en Figure 2.2. L'algorithme le plus ancien est l'algorithme du mathématicien grec *Euclide* (-325,-265), qui permet de trouver le *Plus Grand Commun Diviseur* de deux nombres. C'est, comme nous le verrons dans la prochaine section, la complexité des algorithmes qui définit la complexité des problèmes.

Démarrer en monocycle

Entrée: Un monocycle
Principe:

1. Se placer derrière le monocycle
2. Placer une pédale plus basse que l'autre et plus près de soi
3. Placer la selle dans l'entre-jambe
4. Appuyer sur la pédale la plus basse
5. Monter sur le monocycle (la roue tourne alors d'un quart de tour)

Figure 2.2 – Algorithme du "démarrage en monocycle"

2.2 Classification des algorithmes

Il existe deux types d'algorithmes: les algorithmes *déterministes* et les algorithmes *non-déterministes*. Étant donné un ensemble de données $\mathcal{D}$ fixé en entrée, un algorithme déterministe exécutera toujours la même suite d'opérations, tandis qu'un algorithme non-déterministe pourra exécuter plusieurs suites d'opérations différentes pour un même ensemble de données car un tel algorithme possède des instructions de choix aléatoires parmi n valeurs.

Lors de l'étude de l'efficacité d'un algorithme, on s'intéresse généralement à deux critères:

- *la durée d'exécution*. Un algorithme est dit plus efficace qu'un autre si pour le même jeu de données, il s'exécute en un laps de temps plus court;
- *l'espace mémoire de stockage utilisé*. Un algorithme est dit plus efficace qu'un autre si pour résoudre le même problème, il utilise moins d'espace mémoire.

Ces deux critères sont souvent antinomiques car l'optimisation du temps d'exécution va, en général, à l'encontre de l'optimisation de l'espace occupé. Analyser la complexité d'un algorithme est équivalent à étudier l'incidence de la taille des instances du problème sur la durée d'exécution du programme ou l'espace mémoire utilisé. Chacun de ces critères peut être appliqué à l'ensemble des notions de complexité définies ci-après. Nous définirons donc ces notions en toute généralité, *i.e.* sans spécifier un de ces critères en particulier.

Définition 2.1 (Complexité d'un algorithme). La *complexité d'un algorithme* est la mesure du nombre d'opérations atomiques (*i.e.* indivisibles) qu'il effectue sur les entrées du problème. Elle est par conséquent exprimée en fonction de la taille des entrées du problème.

Il existe trois types de complexité:

1. *complexité au mieux*: le plus petit nombre d'opérations atomiques qu'aura à exécuter l'algorithme pour toute instance du problème considéré. Cette complexité est par conséquent une complexité minimale qui peut être atteinte en toute généralité mais qui ne reflète pas le comportement usuel de l'algorithme;
2. *complexité au pire*: le plus grand nombre d'opérations atomiques qu'aura à exécuter l'algorithme pour toute instance du problème considéré. Cette complexité est par conséquent une complexité maximale qui peut être atteinte en toute généralité mais qui ne reflète pas le comportement usuel de l'algorithme;
3. *complexité en moyenne*: l'espérance mathématique des complexités de l'algorithme pour toute instance du problème considéré. Cette complexité reflète le comportement en moyenne de l'algorithme.

Pourvu de la notion de complexité, nous définissons deux nouvelles notions: *l'efficacité* et *l'optimalité*.

Définition 2.2 (Efficacité). Étant donnés deux algorithmes A et B pour résoudre un problème, A est dit *plus efficace* que B si sa complexité est inférieure à celle de B.

Définition 2.3 (Optimalité). Un algorithme A, résolvant un problème P, est dit *optimal* si sa complexité est la complexité minimale parmi tous les algorithmes résolvant P.

On définit alors la notion de *complexité d'un problème* comme suit.

Définition 2.4 (Complexité d'un problème). La *complexité d'un problème* correspond à la complexité de l'algorithme connu[1] le plus efficace pour le résoudre.

Généralement, lors de l'étude de la complexité d'un algorithme, on recherche un ordre de grandeur de la complexité plutôt que la complexité exacte. Pour ce faire, étant donné un algorithme A, on étudie la rapidité de croissance de la complexité de A en fonction de la taille de ses instances. Soit $f(n)$ la fonction renvoyant la complexité d'un algorithme A pour toute instance d de A de taille n. Pour étudier le comportement asymptotique de la fonction f, on utilise les *notations de Landau* (voir Figure 2.3) – notations qui ont été mises en place pour faciliter la comparaison de fonctions numériques.

Définition 2.5 (Notations de Landau). Soient f et g deux fonctions.
On dit que g *domine f*, noté $f = \mathbf{O}(g)$, si $\exists n_0, \exists c \geq 0, \forall n \geq n_0, f(n) \leq c \times g(n)$

[1] On ne connaît pas forcément un algorithme optimal pour tout problème.

On dit que f *est dominé asymptotiquement par* g, noté $f = \mathbf{\Omega}(g)$, si $g = \mathbf{O}(f)$
On dit que f *est négligeable devant* g, noté $f = \mathbf{o}(g)$, si $\forall c > 0, \exists n_0, \forall n \geq n_0, f(n) \leq c \times g(n)$
On dit que f et g *sont du même ordre*, noté $f = \mathbf{\Theta}(g)$, si $\exists n_0, \exists c_1 > 0, \exists c_2 > 0, \forall n \geq n_0, c_1.g(n) \leq f(n) \leq c_2.g(n)$

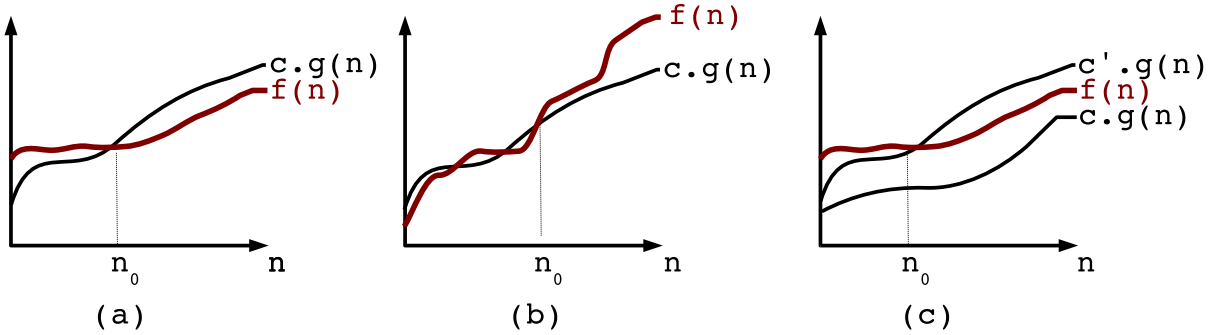


Figure 2.3 – Illustration des notations de Landau. n_0 est la valeur du paramètre n à partir de laquelle toutes les valeurs de (a) $g(n) \in \mathbf{O}(f(n))$, (b) $g(n) \in \mathbf{\Omega}(f(n))$ et (c) $g(n) \in \mathbf{\Theta}(f(n))$.

En algorithmique, les principaux types de complexité sont les suivants:

- la complexité constante, notée $\mathbf{O}(1)$. On rencontre cette complexité quand toutes les instructions sont exécutées une seule fois qu'elle que soit la taille de l'instance;
- la complexité logarithmique, notée $\mathbf{O}(log(n))$. La complexité croît légèrement avec n. Ce cas de figure se rencontre quand la taille de l'instance est divisée par une constante à chaque itération;
- la complexité linéaire, notée $\mathbf{O}(n)$. C'est typiquement le cas d'un algorithme avec une boucle à n itérations telle que le corps de la boucle effectue un travail de durée constante et indépendante de n;
- la complexité quasi-linéaire, notée $\mathbf{O}(n.log(n))$. Cette complexité se rencontre dans les algorithme où à chaque itération la taille du problème est divisée par une constante avec à chaque fois un parcours linéaire des données;
- la complexité quadratique, notée $\mathbf{O}(n^2)$.Typiquement c'est le cas d'algorithmes avec deux boucles imbriquées telles que le nombre d'itérations d'une des boucles est n, que le nombre d'itérations de l'autre boucle est au plus n;
- la complexité cubique, notée $\mathbf{O}(n^3)$. Le cas de trois boucles imbriquées;
- la complexité exponentielle, notée $\mathbf{O}(2^n)$.
- la complexité factorielle, notée $\mathbf{O}(n!)$.

La Figure 2.4 illustre la rapidité de croissance de l'ensemble de ces fonctions usuelles.

Définition 2.6 (Algorithme polynomial). Un algorithme polynomial est un algorithme s'exécutant en un temps polynomial, *i.e.* en $\mathbf{O}(n^c)$ pour une constante c.

La polynomialité d'un algorithme est synonyme "d'efficacité en pratique". L'intérêt majeur d'un algorithme polynomial par rapport à un algorithme exponentiel, illustrée en Figure 2.5, réside dans le fait que l'amélioration des performances technologiques a un effet multiplicatif sur la taille des instances que l'algorithme peut résoudre en un temps donné. Par exemple, si on note N_A (*resp.* N'_A) la taille de la plus grande instance que peut résoudre un algorithme polynomial A en une heure sur un ordinateur O_1

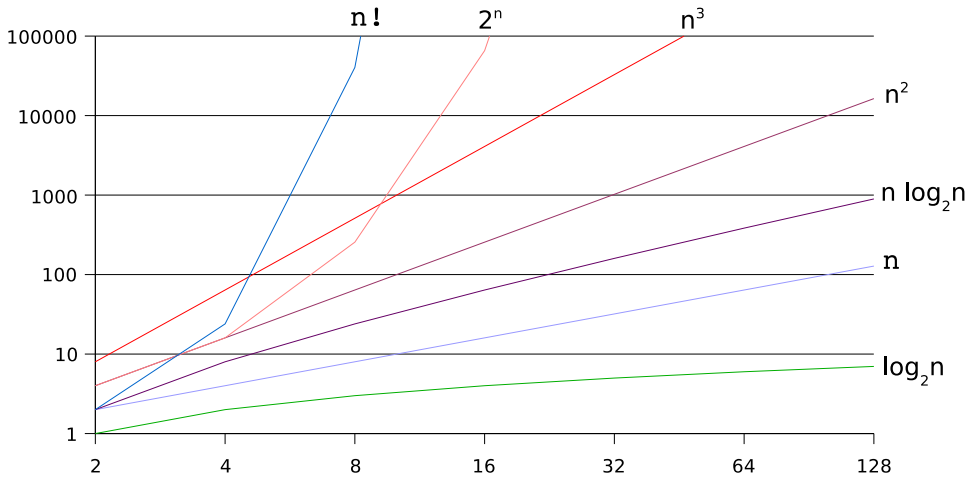


Figure 2.4 – Rapidité de croissance de certaines fonctions usuelles. Attention, par mesure de lisibilité, l'échelle de l'axe des ordonnées est logarithmique.

(*resp.* sur un ordinateur k fois plus performant que O_1) alors $N'_A = k' \cdot N_A$ avec $k' \leq k$. En revanche, des améliorations technologiques similaires ont un effet additif sur les performances d'un algorithme exponentiel. Par exemple, étant donné un algorithme exponentiel B, $N'_B = k' + N_B$ avec $k' \leq k$.

Effet des améliorations technologiques sur des algorithmes de différentes complexités.			
C \ TPGI	Avec un ordinateur actuel	Avec un ordinateur 100 fois plus rapide	Avec un ordinateur 1000 fois plus rapide
n	N_1	$100.N1$	$1000.N_1$
n^2	N_2	$10.N_2$	$31,6.N_2$
n^3	N_3	$4,46.N_3$	$10.N_3$
n^5	N_4	$2,5.N_4$	$3,98.N_4$
2^n	N_5	$N_5 + 6,64$	$N_5 + 9,97$
3^n	N_6	$N_6 + 4,19$	$N_6 + 6,29$

Figure 2.5 – Étant donné un problème $\mathcal{P}$, C représente la Complexité (en temps) d'un algorithme $\mathcal{A}$ résolvant $\mathcal{P}$ et TPGI représente la Taille de la Plus Grande Instance que peut résoudre $\mathcal{A}$ en une heure. Cette table est issue de [53].

2.3 Classification des problèmes

2.3.1 Classes P et NP

Rappelons qu'un problème de décision est un problème dont la réponse est "oui" ou "non". La classe **P** contient tous les problèmes de décision pouvant être résolus par un algorithme déterministe en un temps polynomial. La classe **NP** contient tous les problèmes de décision pour lesquels étant donnée une instance du problème, on peut déterminer, en temps polynomial, si la réponse à cette instance particulière

est "oui". Attention, **NP** est l'acronyme de NON-DÉTERMINISTE POLYNOMIAL, du fait d'une définition alternative (mais équivalente) basée sur la notion d'algorithme non-déterministe: la classe **NP** contient tous les problèmes de décision pouvant être résolus par un algorithme non-déterministe en un temps polynomial.

Afin de préciser les classes **P** et **NP**, considérons, dans un premier temps, le problème des *sept ponts de Königsberg*. Ce problème mathématique historique, qui fit la célébrité de la ville de Königsberg, est le suivant:

> LES SEPT PONTS DE KÖNIGSBERG
> DONNÉES*: Un plan de la ville de Königsberg (cf. Figure 2.6).*
> QUESTION*: Est-il possible de trouver un chemin permettant, à partir d'un point de départ au choix, de passer une et une seule fois par chaque pont, et de revenir à son point de départ, étant entendu qu'on ne peut traverser l'eau qu'en passant par les ponts.*

C'est Leonhard Euler (1707-1783) qui, le premier, donna à ce problème la première résolution mathématique formelle. En considérant la carte de Königsberg de l'époque, illustrée en Figure 2.6, Euler a modélisé le problème sous forme de *graphe** en représentant les ponts par des arêtes et les berges par des sommets. Soit G_K le graphe correspondant. En supprimant la contrainte de retour au point de départ, le problème correspond à la recherche d'une *chaîne eulérienne** dans G_K. Une chaîne eulérienne dans un graphe G est une suite de sommets reliés entre eux par des arêtes et passant par toutes les arêtes de G. Or, il n'existe une chaîne eulérienne dans un graphe G que si G possède 0 ou 2 sommets de *degré** impair. Dans G_K, 3 ou 5 arêtes partent de chaque sommet. Par conséquent, tout sommet de ce graphe est de degré impair, la réponse au problème est donc qu'il n'existe pas de tel chemin.

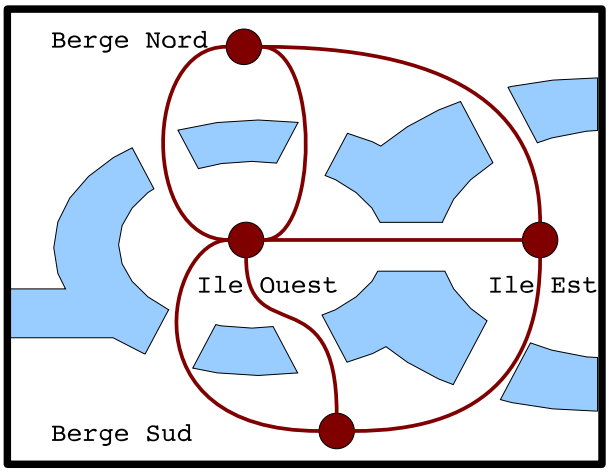


Figure 2.6 – La carte de Königsberg au temps d'Euler. La ville est construite sur deux îles reliées au continent par six ponts, et entre elles par un pont.

En toute généralité, étant donné un graphe à n sommets, il existe $n!$ possibilités de réponses pour ce problème. Malgré tout, Euler a démontré que le problème peut être résolu en $\mathbf{O}(n)$. C'est par conséquent un problème de la classe **P**.

Considérons maintenant la recherche d'un *chemin hamiltonien**. Un chemin hamiltonien dans un graphe G est une suite de sommets reliés entre eux par des arêtes et passant par tous les sommets de

G une et une seule fois. On ne connaît pas d'algorithme efficace (*i.e.* appartenant à **P**) pour déterminer l'existence d'un tel chemin. En revanche, étant donné un parcours, on peut vérifier en temps polynomial s'il est ou non hamiltonien. C'est par conséquent un problème de la classe **NP**.

2.3.2 Les problèmes NP-complets

Comme nous le verrons, un problème est **NP**-complet s'il est parmi les problèmes les plus difficiles de la classe **NP**. Nous débutons cette section en définissant la notion de *transformation polynomiale** d'un problème en un autre illustrée en Figure 2.7.

Définition 2.7 (Transformation polynomiale). Il existe une transformation polynomiale d'un problème de décision $\mathcal{P}_1$ en un problème de décision $\mathcal{P}_2$ si, étant donnée **toute** instance I_1 de $\mathcal{P}_1$, il est possible de construire **une** instance I_2 de $\mathcal{P}_2$ en un temps polynomial (par rapport à la taille de I_1), et tel que la réponse à I_1 est "oui" si et seulement si la réponse à I_2 est "oui" également.

Par la suite, nous utiliserons la notation $\mathcal{P}_1 \propto \mathcal{P}_2$ pour indiquer l'existence d'une transformation polynomiale de $\mathcal{P}_1$ vers $\mathcal{P}_2$. Une propriété importante d'une transformation polynomiale est la transitivité: si $\mathcal{P}_1 \propto \mathcal{P}_2$ et $\mathcal{P}_2 \propto \mathcal{P}_3$ alors $\mathcal{P}_1 \propto \mathcal{P}_3$.

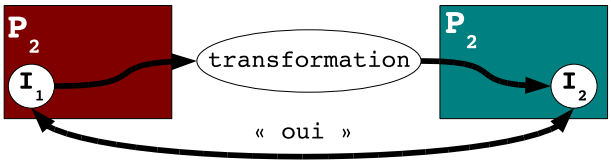


Figure 2.7 – Transformation polynomiale de $\mathcal{P}_1$ vers $\mathcal{P}_2$.

Définition 2.8 (Problèmes C-durs et C-complets). Un problème de décision $\mathcal{P}$ (*resp.* $\mathcal{P} \in$ **C**) est **C**-dur (*resp.* **C**-complet) si tous les problèmes appartenant à la classe **C** se transforment polynomialement en $\mathcal{P}$.

Définition 2.9 (Problèmes NP-durs [38]). Un problème de décision $\mathcal{P}$ est **NP**-dur si tous les problèmes appartenant à **NP** se transforment polynomialement en $\mathcal{P}$.

De prime abord, il est surprenant de penser que des problèmes **NP**-complets puissent exister, mais dans un théorème célèbre, Stephen Cook [38] a prouvé que le problème de satisfaisabilité d'une expression booléenne (le problème SAT) est **NP**-complet. Une conséquence directe de la définition de problème **NP**-complet est que s'il existe un algorithme polynomial pour résoudre **un** problème **NP**-complet alors il est possible de résoudre **tous** les problèmes de **NP** en temps polynomial. Un grand nombre de problèmes **NP**-complets ont été répertoriés dans le livre de Garey et Johnson [53] ainsi que dans le compendium de Crescenzi et Kann [41].

Prouver la **NP**-complétude d'un problème $\mathcal{P}$ nécessite de montrer que:

1. $\mathcal{P} \in$**NP**,
2. tous les problèmes appartenant à **NP** se transforment polynomialement en $\mathcal{P}$.

En pratique, la seconde partie de la preuve de **NP**-complétude d'un problème $\mathcal{P}$ consiste à démontrer qu'il existe une transformation polynomiale d'un problème **NP**-complet $\mathcal{P}_2$ en $\mathcal{P}$. Richard M. Karp en 1972 [68] a prouvé, à l'aide de transformations polynomiales successives illustrées en Figure 2.8, que les problèmes 3-SAT, 3-COLOR, 3-DIMENSIONAL MATCHING, VERTEX COVER, PLANAR-3-COLOR, EXACT-COVER, HAMILTONIAN CIRCUIT, CLIQUE et SUBSET-SUM sont **NP**-complets (pour plus de détails se référer à [68]).

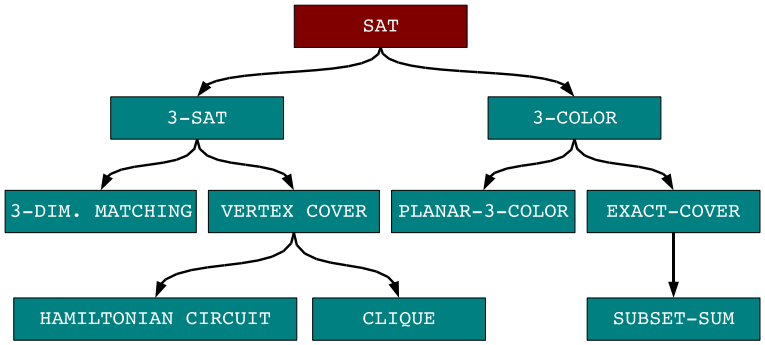


Figure 2.8 – Illustration des transformations polynomiales successives de R.M. Karp en 1972 [68].

2.3.3 P vs NP

Pour résumer, être dans **P**, c'est trouver une solution en temps polynomial, tandis qu'être dans **NP**, c'est prouver en temps polynomial qu'une proposition de réponse est ou non une solution au problème. Ainsi, tout problème qui est dans **P** se trouve dans **NP**. L'un des problèmes ouverts les plus fondamentaux et intéressants de notre époque en informatique théorique est sans doute la question réciproque: a-t-on **NP** $\subseteq$ **P** ? Il en découlerait que **P** = **NP**. Ce problème fait partie des plus grands problèmes ouverts, et sa résolution est primée un million de dollars par le Clay Mathematic Institute.

Ne sachant pas construire de machine non-déterministe, on ne peut que simuler un algorithme non-déterministe à l'aide d'un algorithme déterministe. Malheureusement, il a été démontré que toute simulation de la sorte s'effectue en temps exponentiel. Cela fait que, pour de grandes entrées, on ne peut pas résoudre de problèmes à l'aide d'algorithmes non-déterministes, quelle que soit la puissance de la machine.

Déterminer si **P** = **NP** revient à savoir s'il est possible de simuler un algorithme non-déterministe en temps polynomial. S'il s'avérait que **P** = **NP**, alors on pourrait résoudre tous les problèmes de **NP** en un temps polynomial sur une machine déterministe. Or, comme nous l'avons précisé, les problèmes **NP**-complets sont très fréquents et, pour l'instant, il n'existe pas d'algorithme polynomial en résolvant un. C'est pour cette raison qu'on conjecture cependant que les problèmes **NP**-complets ne sont pas résolvables en temps polynomial.

La Figure 2.9 illustre les relations entre les classes **P**, **NP** et **NP**-complet suivant la réponse à la question "**P** =**NP** ?".

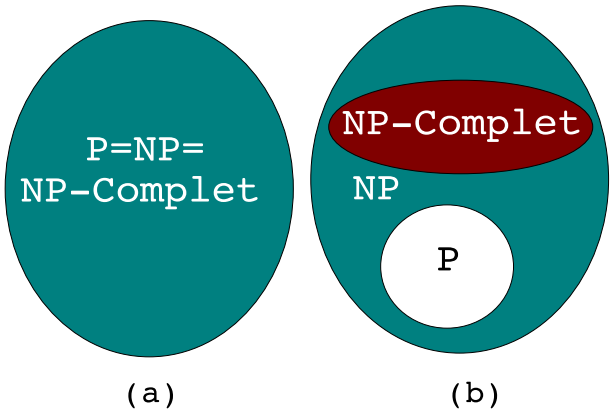


Figure 2.9 – Relations entre les classes **P**, **NP** et **NP**-complet. (a) si **P** =**NP**. (b) **P** $\neq$**NP**

2.4 Contourner la NP-complétude

Nous avons vu que les problèmes **NP**-complets sont les plus difficiles à traiter dans la mesure où trouver une solution polynomiale à l'un d'entre eux reviendrait à trouver une solution pour tous les problèmes de **NP**. D'une part, la classe des problèmes **NP**-complets contient de plus en plus de problèmes. D'autre part, ce sont bien souvent des problèmes très appliqués et qui ont un intérêt certain dans notre vie de tous les jours. Théoriquement intraitables, du moins tant que la conjecture **P** = **NP** n'a pas reçu de réponse positive, ces problèmes sont au cœur de la théorie de la complexité. Résoudre un problème **NP**-complet de manière exacte et en toute généralité, pour des instances de taille relativement importante, demande un temps de calcul irréaliste par rapport aux moyens informatiques dont nous disposons et même dont nous disposerons dans le futur. C'est pourquoi, afin de résoudre, malgré tout, un problème **NP**-complet pour toute instance non-triviale, une des approches suivantes est utilisée: *branch-and-bound*, *heuristique*, *approximation* et *complexité paramétrée*.

2.4.1 Branch-and-Bound

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

Dans les méthodes par séparation et évaluation, la séparation permet d'obtenir une méthode générique pour énumérer toutes les solutions tandis que l'évaluation évite l'énumération systématique de toutes les solutions.

Séparation La phase de séparation consiste à diviser le problème, noté $\mathcal{P}$, en un certain nombre de sous-problèmes qui ont chacun leur ensemble de solutions, de telle sorte que tous ces ensembles forment un *recouvrement** de l'ensemble des solutions de $\mathcal{P}$. Ainsi, en résolvant tous les sous-problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Ce principe de séparation peut être appliqué de manière récursive à chacun des sous-ensembles de solutions obtenus,

et ceci tant qu'il y a des ensembles contenant plusieurs solutions. Les ensembles de solutions ainsi construits (et leurs sous-problèmes associés) ont une hiérarchie naturelle en arbre, souvent appelée arbre de recherche ou *arbre de décision**.

Évaluation Tout noeud de l'arbre de recherche correspond à un sous-problème. L'évaluation d'un nœud n de l'arbre de recherche a pour but de déterminer l'optimum de l'ensemble des solutions pour le sous-problème associé à n ou, au contraire, de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème (typiquement, qu'il n'y a pas de solution optimale). Lorsqu'un tel nœud est identifié dans l'arbre de recherche, il est donc inutile d'effectuer la séparation de son espace de solutions.

L'optimum du sous-problème correspondant à un noeud peut être déterminé lorsque le sous-problème devient « suffisamment simple » (*i.e.* l'ensemble des instances du sous-problème est de faible cardinalité). Par exemple, lorsque l'ensemble des solutions est un singleton, le problème est effectivement simple: l'optimum est l'unique élément de cet ensemble.

Pour déterminer qu'un ensemble de solutions ne contient pas de solution optimale, la méthode la plus générale consiste à conserver en mémoire la meilleure solution S trouvée jusqu'alors. S'il n'existe pas de solution dans le sous-problème meilleure que S, alors on peut affirmer que le sous-problème ne contient pas l'optimum.

2.4.2 Heuristique

Pour certains problèmes, les algorithmes ont une complexité beaucoup trop grande pour obtenir un résultat en temps raisonnable, même si l'on pouvait utiliser une puissance de calcul phénoménale. On est donc amené à rechercher une solution la plus proche possible d'une solution optimale en procédant par essais successifs. Puisque toutes les combinaisons ne peuvent être essayées, certains choix doivent être faits. Ces choix, généralement très dépendants du problème traité, constituent ce que l'on appelle une *heuristique*. Le but d'une heuristique est donc de ne pas essayer toutes les combinaisons possibles avant de trouver celle qui répond au problème, afin de trouver une solution approchée convenable (qui peut être exacte dans certains cas) dans un temps raisonnable.

Le problème majeur d'une heuristique réside dans l'impossibilité d'évaluer l'éloignement de la solution retournée par rapport à la solution optimale.

2.4.3 Approximation

Contrairement à une heuristique, qui trouve des solutions acceptables assez rapidement, un algorithme d'approximation fournit une solution dont la qualité et la complexité temporelle peuvent être évaluées. Afin de résoudre un problème d'optimisation **NP**-dur à l'aide d'un algorithme polynomial, il est nécessaire d'accepter que l'algorithme ne retourne pas dans tous les cas la solution optimale mais qu'il est possible de quantifier l'éloignement de la solution retournée de la solution optimale. Cet éloignement relatif à la solution optimale est quantifié à l'aide de la notion de *ratio de performance*.

Définition 2.10 (Ratio de performance). Étant donné un problème d'optimisation $\mathcal{P}$, pour toute instance I de $\mathcal{P}$, soient $Algo(I)$ la solution renvoyée par un algorithme A et $Opt_{\mathcal{P}}(I)$ la solution optimale. Le ratio de performance $R_p(I)$ est défini comme suit:

$R_p(I) = max(\frac{Algo(I)}{Opt_{\mathcal{P}}(I)}, \frac{Opt_{\mathcal{P}}(I)}{Algo(I)})$

D'après la Définition 2.10, que $\mathcal{P}$ soit un problème de minimisation ou de maximisation, la valeur du ratio de performance $R_p(I)$ est toujours supérieure ou égale à 1 et égale à 1 si l'algorithme d'approximation renvoie la solution optimale. Le ratio de performance quantifie l'écart à l'optimal de la solution approchée.

Définition 2.11 (Algorithme d'α-approximation). Étant donné un problème d'optimisation $\mathcal{P}$ et un algorithme d'approximation A pour $\mathcal{P}$, A est un algorithme d'α-approximation pour $\mathcal{P}$ si, pour toute instance I de $\mathcal{P}$, le ratio de performance est borné par α: *i.e.* $R_p(I) \leq \alpha$.

D'après cette définition, un algorithme d'1-approximation est un algorithme exact. De plus, le ratio de performance est toujours supérieur ou égal à 1 et tend vers 1 lorsque la solution renvoyée par l'algorithme d'approximation se rapproche de la solution optimale. Lors de la recherche d'un algorithme d'α-approximation pour un problème **NP**-dur $\mathcal{P}$, l'objectif est de minimiser α afin de se rapprocher le plus possible de la solution optimale. Par la suite, un algorithme donnant une *approximation à un facteur α près* désignera un algorithme d'α-approximation. Un problème d'optimisation est alors *approximable à un facteur α près* en temps polynomial s'il existe un algorithme polynomial qui est une approximation à un facteur α près pour le problème.

Définition 2.12 (Classe NPO). La classe **NPO** contient tous les problèmes d'optimisation dont la version décisionnelle est dans **NP**.

L'approximation des problèmes **NP**-durs variant d'un problème à un autre, il est courant de classer les problèmes **NP**-durs suivant leur appartenance ou non aux classes de problèmes suivantes: **APX**, **PTAS** et **FPTAS**.

Définition 2.13 (Classe APX). La classe **APX** (la classe des problèmes approximables à ratio constant) est une sous-classe des problèmes appartenant à **NPO**. Cette classe contient tous les problèmes $\mathcal{P}$ de **NPO** admettant un algorithme ayant un facteur d'approximation constant, *i.e.* tous les problèmes pour lesquels il existe un algorithme d'α-approximation A pour $\mathcal{P}$, pour une constante $\alpha > 1$.

Définition 2.14 (Classe PTAS). La classe **PTAS** (Polynomial-Time Approximation Scheme) est une sous-classe des problèmes appartenant à **NPO**. Cette classe contient tous les problèmes $\mathcal{P}$ de **NPO** pour lesquels il existe un algorithme polynomial A qui est une approximation à un facteur $(1+\epsilon)$ près du problème, pour toute constante $\epsilon > 0$.

D'après la définition de la classe **PTAS**, le temps de calcul de l'algorithme A peut dépendre exponentiellement de $\frac{1}{\epsilon}$. Par exemple, le temps de calcul peut être de la forme: $\mathbf{O}(n^{\mathbf{O}(\frac{1}{\epsilon})})$ ou $\mathbf{O}(2^{\frac{1}{\epsilon}^{\mathbf{O}(1)}} n^{\mathbf{O}(1)})$.

Définition 2.15 (Classe FPTAS). La classe **FPTAS** (Fully Polynomial-Time Approximation Scheme) est une sous-classe des problèmes appartenant à **PTAS**. Cette classe contient tous les problèmes $\mathcal{P}$ de **PTAS** pour lesquels le temps de calcul de l'algorithme A est polynomial en $\frac{1}{\epsilon}$.

Notons que par définition, si $\mathbf{P} \neq \mathbf{NP}$ alors $\mathbf{FPTAS} \subseteq \mathbf{PTAS} \subseteq \mathbf{APX} \subseteq \mathbf{NPO}$.

Qualité d'approximation des problèmes de NPO. Papadimitriou *et al.* [81] ont proposé d'utiliser un formalisme logique pour étudier les qualités d'approximation des problèmes de **NPO**. Nous ne donnerons dans ce manuscrit que la définition de la classe **MaxSNP**. Pour de plus amples informations se référer à [81].

La transformation polynomiale présentée en Section 2.3.2 ne tient pas compte de la notion d'approximation et est donc inadaptée pour préserver cette dernière. C'est pourquoi l'étude des algorithmes d'approximation nécessite une transformation préservant l'approximation – c'est le rôle de la L-réduction illustrée en Figure 2.10. Étant donné un problème d'optimisation $\mathcal{P}$, pour toute instance I de $\mathcal{P}$, soient $App_{\mathcal{P}}(I)$ la solution approchée renvoyée par un algorithme d'approximation A et $Opt_{\mathcal{P}}(I)$ la solution optimale.

Définition 2.16 (L-réduction). Une L-réduction d'un problème d'optimisation $\mathcal{P}_1$ vers un problème d'optimisation $\mathcal{P}_2$ est une paire de fonctions (f, g), toutes deux calculables en temps polynomial, et telles que:

- si I_1 est une instance de $\mathcal{P}_1$ alors $f(I_1)$ est une instance de $\mathcal{P}_2$ telle que $Opt_{\mathcal{P}_2}(f(I_1)) \leq \alpha.Opt_{\mathcal{P}_1}(I_1)$ où α est une constante strictement positive;
- $|Opt_{\mathcal{P}_1}(I_1) - g(App_{\mathcal{P}_2}(f(I_1)))| \leq \beta.|Opt_{\mathcal{P}_2}(f(I_1)) - App_{\mathcal{P}_2}(f(I_1))|$ où β est une constante strictement positive.

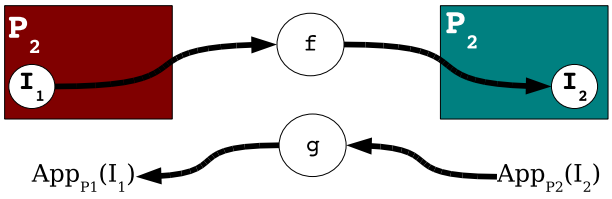


Figure 2.10 – L-réduction de $\mathcal{P}_1$ vers $\mathcal{P}_2$.

Nous reprenons ci-après les définitions proposées dans [80]. Un *type similaire fini* $\mathcal{T}$ est la donnée d'un k-uplet $(n_1, \ldots, n_k) \in \mathrm{N}^k$ pour k fini. Une *structure finie* $S = (U, R)$ sur un type similaire fini $S = (n_1, \ldots, n_k)$ est la donnée d'un ensemble fini U appelé univers et d'un ensemble $R = (r_1, \ldots, r_k)$ de k relations sur U vérifiant pour tout $i : r_i \subseteq U^{n_i}$. S est qualifiée de $\mathcal{S}$-structure finie.

Définition 2.17 (Classe MaxSNP$_0$). La classe **MaxSNP**$_0$ est une sous-classe des problèmes appartenant à **NPO**. Cette classe contient tous les problèmes $\mathcal{P}$ de **NPO** tels qu'il existe deux types similaires finis I et S, une formule ϕ du premier ordre sans quantificateur et deux constantes k et ℓ, tels que les optima de $\mathcal{P}$ peuvent être exprimés sous la forme : $\beta_{\mathcal{P}}(I) = \max_S |x \in U^k : \exists y \in U^\ell, \phi(I, S, x, y)|$ où I est une $\mathcal{I}$-structure finie d'univers U et S une $\mathcal{S}$-structure finie de même univers U. La structure I décrit les instances du problème $\mathcal{P}$, la structure S les solutions sur I, la formule ϕ les conditions de réalisabilité des solutions du problème.

Définition 2.18 (Classe MaxSNP). La classe **MaxSNP** contient tous les problèmes $\mathcal{P}$ de **NPO** pour lesquels il existe une L-réduction vers un problème appartenant à **MaxSNP**$_0$.

2.4.4 Complexité paramétrée

Contrairement aux heuristiques et aux algorithmes d'approximation, qui trouvent des solutions qui peuvent ne pas être satisfaisantes en pratique, la complexité paramétrée propose de restreindre l'explosion combinatoire des problèmes **NP**-durs à une partie de l'entrée: *le paramètre*. La complexité paramétrée fut introduite par Downey et Fellows [47, 46, 1] pour contourner la **NP**-complétude.

Problèmes à exponentialité contrôlée. De nombreux problèmes sont pourvus de paramètres décrivant des propriétés particulières à un ensemble d'instances. Chaque paramètre peut être utilisé pour "découper" un problème $\mathcal{P}$ en autant de sous-problèmes que de valeurs pour ce paramètre. Nous appelons la classe **FPT** (Fixed-Parameter Tractable) la classe des problèmes à exponentialité contrôlée.

Définition 2.19 (Problème paramétré). Un problème paramétré est un ensemble de couples $\mathcal{P}_1 \subseteq \Sigma^* \times \Gamma^*$ définis à partir de deux alphabets finis Σ et Γ. Pour chaque couple $(x, k) \in \mathcal{P}_1$, x représente l'instance et k le paramètre.

Définition 2.20 (Classe FPT). Un problème paramétré $\mathcal{P}_1$ appartient à la classe **FPT** s'il existe un algorithme A, pour résoudre $\mathcal{P}_1$, de complexité en temps $\mathbf{O}(f(k)|x|^c)$ où f est une fonction quelconque et c une constante.

Problèmes à exponentialité incontrôlable. Toutefois, certains problèmes semblent ne pas appartenir à la classe **FPT** (ils sont dits "Fixed-Parameter Intractable"). Afin de définir les problèmes à exponentialité incontrôlable, nous définissons la notion de *FPT-transformation*.

Définition 2.21 (FPT-transformation). Soient $\mathcal{P}_1 \subseteq \Sigma^* \times \Gamma^*$ et $\mathcal{P}_2 \subseteq (\Sigma')^* \times (\Gamma')^*$ deux problèmes paramétrés. Une *FPT-transformation* de $\mathcal{P}_1$ en $\mathcal{P}_2$ est une fonction $\mathcal{F} : \Sigma^* \times \Gamma^* \mapsto (\Sigma')^* \times (\Gamma')^*$ telle que:

1. $\forall (x, k) \in \Sigma^* \times \Gamma^*$, (x, k) est une solution de $\mathcal{P}_1$ si et seulement si $\mathcal{F}(x, k)$ est une solution de $\mathcal{P}_2$;
2. $\mathcal{F}$ est calculable par un algorithme de complexité en temps $\mathbf{O}(f(k)|x|^c)$ où f est une fonction quelconque et c une constante;
3. il existe une fonction $g : \mathbb{N} \mapsto \mathbb{N}$ telle que $\forall (x, k) \in \Sigma^* \times \Gamma^*$, avec $\mathcal{F}(x, k) = (x', k')$, on a $k' \leq g(k)$.

Par la suite, nous utiliserons la notation $\mathcal{P}_1 \leq^{fpt} \mathcal{P}_2$ pour indiquer l'existence d'une FPT-transformation de $\mathcal{P}_1$ vers $\mathcal{P}_2$. Les FPT-transformations sont utilisées pour classer les problèmes paramétrés en classes de problèmes définies à l'aide de la complexité des *circuits de décisions*. Un *circuit de décision* produit un résultat (également appelé la *sortie*) à partir d'une entrée en appliquant une séquence d'opérations simples (appelées les *portes*).

Une porte est elle-même composée d'entrées et de sorties. Un exemple classique de porte est la porte "OU" qui produit un 1 en sortie si au moins une de ses entrées est à 1, et qui produit un 0 sinon. La sortie d'une porte peut bien entendu être utilisée comme entrée d'une autre porte. On dénombre deux types de portes: les portes *bornées* qui ont un nombre limité d'entrées et les portes *non-bornées* qui ont un nombre illimité d'entrées.

Un *chemin* dans un circuit C est une séquence de portes $S = p_1, p_2, \ldots p_{|S|}$ telle que:

1. l'entrée du circuit C est reliée à une entrée de la porte p_1;
2. $\forall 1 \leq i < |S|$ une sortie de la porte p_i est reliée à une entrée de la porte p_{i+1};
3. une sortie de la porte $p_{|S|}$ est reliée à la sortie du circuit.

La *profondeur* d'un circuit C, notée $p(C)$, correspond au nombre maximum de portes (bornées ou non) situées sur un chemin du circuit. La *trame* (de l'anglais *weft*) d'un circuit C, notée $t(C)$, correspond au nombre maximum de portes non-bornées situées sur un chemin du circuit.

Une *famille de circuits de décision* F est un ensemble de circuits de décision. Une famille de circuits de décision F a une profondeur (*resp.* une trame) bornée s'il existe une constante h telle que $\forall C \in F, p(C) \leq h$ (*resp.* $\forall C \in F, t(C) \leq h$).

Un circuit de décision C *accepte* un vecteur v en entrée si la sortie de C vaut 1 lorsque ses entrées sont données par v. Le *poids* d'un vecteur v est égal au nombre de 1 dans v.

On définit le problème paramétré de CIRCUIT DE DÉCISION comme suit:

PARAMETERIZED DECISION CIRCUIT (PDC$_F$)
DONNÉES*: Une famille de circuits de décision F et un entier k.*
QUESTION*: Existe-t-il un vecteur v de poids égal à k tel que $\forall C \in F$, C accepte v ?*

Définition 2.22 (Classe W[t]). Un problème paramétré $(\mathcal{P}, \mathcal{K})$ appartient à la classe **W[t]** si il existe une FPT-tranformation de $(\mathcal{P}, \mathcal{K})$ en PDC$_{F(p,t)}$ où $F(p,t)$ est la famille des circuits de décision ayant une profondeur p et une trame t bornées.

Définition 2.23 (Classe W[P]). Un problème paramétré $(\mathcal{P}, \mathcal{K})$ appartient à la classe **W[P]** si il existe une FPT-tranformation de $(\mathcal{P}, \mathcal{K})$ en PDC$_F$ où F est la famille de tous les circuits de décision.

Ces différentes classes s'organisent en une hiérarchie qui a été proposée dans [46]:
$\mathbf{FPT} \subseteq \mathbf{W}[\mathbf{1}] \subseteq \mathbf{W}[\mathbf{2}] \subseteq \ldots \subseteq \mathbf{W}[\mathbf{t}] \subseteq \ldots \mathbf{W}[\mathbf{P}]$

PARTIE II

Comparaison de structures de molécules d'ARN

CHAPITRE 3

Présentation générale

La comparaison de structures de molécules d'ARN est fondamentale, entre autres, pour:

- l'identification de structures hautement conservées au cours de l'évolution (non détectables dans la séquence primaire, qui est souvent faiblement conservée). Cette conservation peut indiquer une fonction commune aux molécules d'ARN étudiées [49, 61, 66, 90];
- la classification de molécules d'ARN de diverses espèces (phylogénie) [15, 29, 95];
- la prédiction du repliement d'une molécule d'ARN par référence à des structures déjà connues [64, 100];
- l'identification d'une structure consensus, et par conséquent d'un rôle commun aux molécules [31, 96].

C'est pourquoi l'étude des molécules d'ARN fait partie des principaux domaines de recherches en bio-informatique. Dans ce manuscrit, nous allons étudier diverses notions d'homologies portant sur la structure des molécules d'ARN. Nous débuterons cette partie par une présentation de deux des nombreux formalismes de représentation de la structure des molécules d'ARN. Les deux formalismes auxquels nous nous intéressons dans cette partie sont (1) les séquences arc-annotées et (2) les 2-intervalles. Nous définirons également un ensemble de problèmes associés à ces formalismes. Puis, nous présenterons nos résultats.

Une séquence arc-annotée consiste en un couple (S, P) où S est une séquence de caractères et P un ensemble d'arcs connectant des couples de caractères de S. Comme nous le verrons par la suite, cette représentation permet aisément de modéliser la structure d'une molécule d'ARN en représentant la structure primaire à l'aide de S et les liaisons hydrogènes à l'aide des arcs de P. Dans ce chapitre, nous présenterons trois problèmes d'homologie entre molécules d'ARN utilisant les séquences arc-annotées: (1) le calcul de la distance d'édition entre deux molécules d'ARN (le problème EDIT), (2) la recherche de la plus grande structure conservée entre deux molécules d'ARN (le problème LAPCS) et (3) la recherche d'une structure conservée dans une molécule d'ARN (le problème APS).

Formellement, un 2-intervalle est l'union disjointe de deux intervalles sur une ligne. Comme nous le verrons par la suite, les 2-intervalles permettent une représentation plus macroscopique de la structure d'une molécule d'ARN. Nous présenterons les deux problèmes d'homologie entre molécules d'ARN introduits par Stéphane Vialette dans [93, 94], à savoir: (1) la recherche de la plus grande sous-famille respectant une structure donnée et (2) la recherche de motifs structuraux. Par souci de lisibilité, ces deux problèmes ne seront plus précisément définis qu'en Section 3.2.

3.1 Les séquences arc-annotées

Après avoir présenté en toute généralité les séquences arc-annotées introduites par Patricia Anne Evans dans [48], nous montrerons comment on peut les utiliser pour représenter les structures de molécules d'ARN.

3.1.1 Notations et modélisation de molécules d'ARN

Notations. Étant donné un alphabet Σ, nous utiliserons S pour représenter une séquence de $|S|$ caractères appartenant à Σ. Soit $S[i]$ le $i^{ème}$ caractère de S, alors S peut s'écrire sous la forme:
$S = S[1]\ S[2] \dots S[|S|]$.

Définition 3.1 (Séquence arc-annotée). Étant donné un alphabet Σ, une *séquence arc-annotée* est un couple (S, P) où S est une séquence de $|S|$ caractères appartenant à Σ et P est un ensemble fini d'arcs. Tout arc appartenant à P est représenté par une paire de positions (i, j) telle que $i, j \in [1, |S|]$ et $i < j$. Par la suite, un caractère *libre* (*resp. marqué*) désigne un caractère de S non incident (*resp.* incident) à un arc de P.

Evans [48] a proposé quatre restrictions sur P qui ont été très largement reprises lors de l'étude des séquences arc-annotées:

1. il n'existe pas de caractères incidents à plus d'un arc (*i.e.* si $(i, j) \in P$ alors $\forall k \in [1, |S|]$ tel que $i \neq k$ et $j \neq k$, $\{(i, k), (k, i), (k, j), (j, k)\} \bigcap P = \emptyset$);
2. il n'existe pas d'arcs se croisant (*i.e.* $\forall (i, j), (k, l) \in P$ tels que $i < k$, soit $j \leq k$, soit $l \leq j$);
3. il n'y a pas d'arc contenu dans un autre (*i.e.* $\forall (i, j), (k, l) \in P$ tels que $i < k$, soit $j \leq k$, soit $j \leq l$);
4. il n'y a pas d'arc du tout (*i.e.* $P = \emptyset$).

Ces restrictions ont permis de définir cinq niveaux de complexité pour les problèmes sur les séquences arc-annotées (illustrés en Figure 3.1):

- UNLIMITED - pas de restrictions;
- CROSSING - restriction 1;
- NESTED - restrictions 1 et 2;
- CHAIN - restrictions 1, 2 et 3;
- PLAIN - restriction 4.

Étant données deux niveaux de complexité n_1 et n_2, on note $n_1 \subset n_2$ si toute séquence arc-annotée respectant le niveau de complexité n_1 respecte également le niveau de complexité n_2.

$\mathcal{P}$(UNLIMITED, UNLIMITED)								
$\cup$								
$\mathcal{P}$(UNLIMITED, CROSSING)	$\supset$	$\mathcal{P}$(CROSSING, CROSSING)						
$\cup$		$\cup$						
$\mathcal{P}$(UNLIMITED, NESTED)	$\supset$	$\mathcal{P}$(CROSSING, NESTED)	$\supset$	$\mathcal{P}$(NESTED, NESTED)				
$\cup$		$\cup$		$\cup$				
$\mathcal{P}$(UNLIMITED, CHAIN)	$\supset$	$\mathcal{P}$(CROSSING, CHAIN)	$\supset$	$\mathcal{P}$(NESTED, CHAIN)	$\supset$	$\mathcal{P}$(CHAIN, CHAIN)		
$\cup$		$\cup$		$\cup$		$\cup$		
$\mathcal{P}$(UNLIMITED, PLAIN)	$\supset$	$\mathcal{P}$(CROSSING, PLAIN)	$\supset$	$\mathcal{P}$(NESTED, PLAIN)	$\supset$	$\mathcal{P}$(CHAIN, PLAIN)	$\supset$	$\mathcal{P}$(PLAIN, PLAIN)

Table 3.1 – Hiérarchie entre les différents sous-problèmes de $\mathcal{P}$ en fonction des niveaux de complexité.

Propriété 3.1 (Hiérarchie des niveaux de complexité). PLAIN $\subset$ CHAIN $\subset$ NESTED $\subset$ CROSSING $\subset$ UNLIMITED.

Le bien-fondé de la Propriété 3.1 découle de la définition des niveaux de complexité. Étant donnés deux séquences arc-annotées (S, P), (T, Q) et un problème $\mathcal{P}$ d'homologie entre (S, P) et (T, Q), on note $\mathcal{P}(n_1, n_2)$ le sous-problème de $\mathcal{P}$ où P respecte le niveau de complexité n_1 et Q respecte le niveau

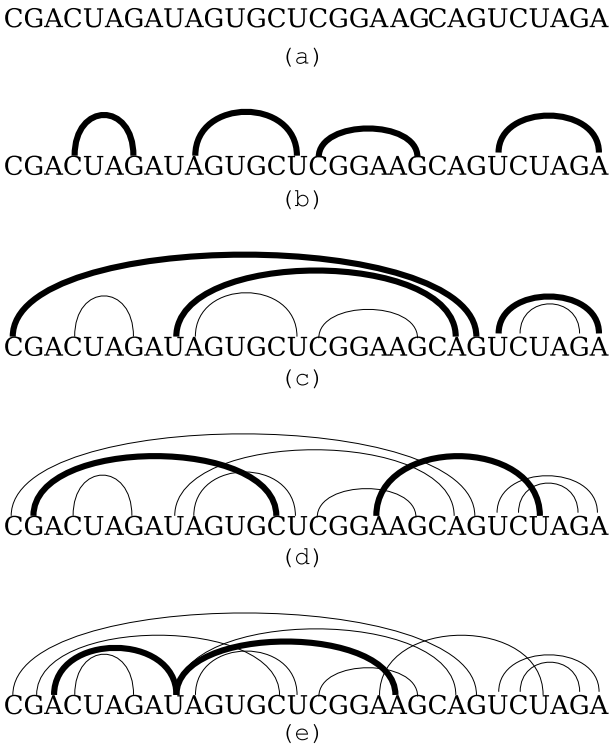


Figure 3.1 – Exemple de représentation des séquences arc-annotées en fonction des niveaux de complexité: (a) PLAIN, (b) CHAIN, (c) NESTED, (d) CROSSING et (e) UNLIMITED. Les arcs en gras sont responsables du changement de niveau de complexité.

de complexité n_2. Sans perte de généralité, si n_1 et n_2 sont deux niveaux de complexité différents alors on considère que $n_2 \subset n_1$ (*i.e.* (T, Q) est la séquence avec le niveau de complexité le plus "faible").

Soient $\mathcal{P}(n_1, n_2)$ et $\mathcal{P}(n_3, n_4)$ deux sous-problèmes de $\mathcal{P}$. On note $\mathcal{P}(n_1, n_2) \subset \mathcal{P}(n_3, n_4)$ si $n_1 \subset n_3$ et $n_2 \subset n_4$. Cette définition et la Propriété 3.1 permettent de définir également une hiérarchie entre les différents sous-problèmes de $\mathcal{P}$, comme illustré dans la Table 3.1.

Pourvus de cette hiérarchie entre les différents sous-problèmes de $\mathcal{P}$, nous indiquons la propriété sur la propagation de la complexité du problème $\mathcal{P}$ suivante.

Propriété 3.2. *Soient $\mathcal{P}(n_1, n_2)$ et $\mathcal{P}(n_3, n_4)$ deux sous-problèmes de $\mathcal{P}$ tels que*

1. *n_1, n_2, n_3 et n_4 appartiennent à $\{$PLAIN, CHAIN, NESTED, CROSSING, UNLIMITED$\}$;*
2. *$\mathcal{P}(n_1, n_2) \subset \mathcal{P}(n_3, n_4)$.*

Si le problème $\mathcal{P}(n_1, n_2)$ est **NP***-complet (resp. le problème $\mathcal{P}(n_3, n_4)$ est polynomial) alors le problème $\mathcal{P}(n_3, n_4)$ (resp. $\mathcal{P}(n_1, n_2)$) l'est également.*

Modélisation de molécules d'ARN. Rappelons que l'on dénombre généralement trois niveaux de structure pour une molécule d'ARN. Hormis la structure dite *primaire* de l'ARN, également appelée *séquence*, il existe deux autres niveaux de structure: la structure *secondaire* et la structure *tertiaire*.

La structure primaire d'une molécule d'ARN peut être clairement modélisée à l'aide d'une séquence arc-annotée définie sur l'alphabet $\Sigma = \{A, C, G, U\}$ et respectant le niveau de complexité PLAIN.

La structure secondaire d'une molécule d'ARN désigne le repliement planaire de la structure primaire. La contrainte de planarité des structures secondaires d'ARN induit le lemme suivant.

Lemme 3.1. *Toute structure secondaire sans pseudo-noeud peut être modélisée à l'aide d'une séquence arc-annotée définie sur l'alphabet* $\Sigma = \{A, C, G, U\}$ *et respectant le niveau de complexité* NESTED.

Contrairement à la structure secondaire qui peut toujours être représentée en deux dimensions, la structure tertiaire d'une molécule d'ARN fait explicitement référence à un repliement dans un espace à trois dimensions. Cette particularité induit le lemme suivant.

Lemme 3.2. *Toute structure tertiaire peut être modélisée à l'aide d'une séquence arc-annotée définie sur l'alphabet* $\Sigma = \{A, C, G, U\}$ *et respectant le niveau de complexité* CROSSING.

Nous définissons, ci-après, trois paramètres sur une séquence arc-annotée (S, P): la *hauteur*, la *largeur de coupe* et la *largeur de bande*.

Définition 3.2 (Hauteur). Étant donnée une séquence arc-annotée (S, P), la *hauteur* de (S, P) est égale à $max\{|P'|\}$ où $P' \subseteq P$ est tel que $\forall (i, j) \in P'$, $\nexists (k, l) \in P'$ tel que $i < k < j < l$ ou $i < j < k < l$.

Définition 3.3 (Largeur de coupe - Cutwidth). Étant donnée une séquence arc-annotée (S, P), la *largeur de coupe* de (S, P) est égale au nombre maximal d'arcs se croisant deux à deux.

Définition 3.4 (Largeur de bande - Bandwidth). Étant donnée une séquence arc-annotée (S, P), la *largeur de bande* de (S, P) est égale à $max_{(i,j)\in P}\{|j - i|\}$.

3.1.2 Quelques problèmes utilisant les séquences arc-annotées

Dans cette section, nous allons présenter trois problèmes utilisant la modélisation des structures de molécules d'ARN à l'aide de séquences arc-annotées: (1) le calcul de la distance d'édition entre deux molécules d'ARN (le problème EDIT), (2) la recherche de la plus grande structure conservée entre deux molécules d'ARN (le problème LAPCS) et (3) la recherche d'une structure conservée dans une molécule d'ARN (le problème APS).

3.1.2.1 Le problème du calcul de la distance d'édition

À l'origine, la notion de distance d'édition a été introduite dans le contexte de la comparaison de chaînes de caractères [59]. Comme nous le verrons par la suite, cette notion peut facilement s'étendre aux séquences arc-annotées et permettre ainsi de comparer deux molécules d'ARN. La distance d'édition entre deux chaînes de caractères mesure le coût de transformation d'une chaîne vers l'autre. Cette transformation est rendue possible au moyen de trois opérations dites d'édition.

Définition 3.5 (Opérations d'édition). Étant donnée une chaîne S de caractères appartenant à l'alphabet Σ, les trois *opérations d'édition* sont les suivantes:

1. l'*insertion* d'un caractère dans la chaîne S;
2. la *suppression* d'un caractère de la chaîne S;
3. la *substitution* d'un caractère de la chaîne S par un caractère de Σ.

Afin d'évaluer la distance d'édition, on attribue un coût à chacune de ces opérations: w_i pour une insertion, w_d pour une suppression et w_m pour une substitution (le d de w_d et le m de w_m proviennent respectivement de l'anglais *deletion* et *matching*).

Définition 3.6 (Transformation). Soient S et T deux chaînes de caractères. Nous définissons une *transformation* de S vers T comme étant une suite de substitutions, d'insertions et de suppressions produisant T à partir de S.

Le *coût d'une transformation* est égal à la somme des coûts des opérations composant la transformation. Nous définissons la *distance d'édition* entre deux séquences S et T comme suit.

Définition 3.7 (Distance d'édition). Soient S et T deux chaînes de caractères. La *distance d'édition* entre S et T est égale au coût minimum d'une transformation de S vers T.

Cette distance est également appelée *distance de Levenshtein* [73]. Dans [59], Gusfield propose un algorithme de *programmation dynamique** pour calculer la distance de Levenshtein entre deux chaînes de caractères S et T de complexité en temps $\mathbf{O}(nm)$ avec $n = |S|$ et $m = |T|$.

Définition 3.8 (Alignement). Soient S et T deux chaînes de caractères de l'alphabet Σ. Un *alignement* de S et T est une paire de chaînes de caractères (S', T'), de taille identique, construites à partir de (S, T) avec l'alphabet $\Sigma \cup \{-\}$ telles que $\forall 1 \leq i \leq |S'|$:

- soit $S'[i] \neq$"$-$" et $T'[i] \neq$"$-$": les caractères $S'[i]$ et $T'[i]$ sont alignés;
- soit $S'[i] =$"$-$" et $T'[i] \neq$"$-$": le caractère $T'[i]$ est inséré;
- soit $S'[i] \neq$"$-$" et $T'[i] =$"$-$": le caractère $S'[i]$ est supprimé.

Le symbole $-$ est appelé *gap*.

Étant données deux chaînes de caractères, il existe plusieurs alignements possibles. Par définition, la suppression des *gaps* de S' (*resp.* T') produit la chaîne de caractères d'origine S (*resp.* T). De plus, dans un alignement, deux *gaps* ne peuvent être alignés. Par exemple, un alignement possible des chaînes de caractères "THESARD" et "RETARD" est:

T H E – S A R – D
R – E T – A R D –

Un alignement de deux chaînes de caractères S et T est la représentation d'une transformation de S vers T: deux caractères alignés correspondent à une substitution, un caractère de S' (*resp.* T') supprimé (*resp.* inséré) à une suppression (*resp.* insertion). Il est tout aussi aisé de passer d'une transformation à un alignement.

L'extension de la notion de distance d'édition aux séquences arc-annotées est essentiellement l'oeuvre de Kaizhong Zangh *et al.* [99] et Guo-Hin Lin *et al.* [75]. Il existe d'autres extensions [5] mais qui ont l'inconvénient de traiter séparément, du point de vue des opérations d'édition, la séquence et les arcs. Dans ce manuscrit, nous ne traiterons que des deux premières extensions précédemment citées.

Extension à la Zhang *et al.* [99]. Étant donné que la différence entre une séquence et une séquence arc-annotée réside dans la présence d'arcs, Zhang *et al.*, dans [99], proposent tout naturellement d'appliquer les opérations d'édition aux arcs. Ainsi, dans la proposition de Zhang *et al.*, le nombre des opérations d'édition passe de trois à six: trois opérations pour les caractères et trois opérations pour les arcs.

Définition 3.9 (Opérations d'édition). Étant donnée une séquence arc-annotée (S, P) où S est une séquence de caractères appartenant à l'alphabet Σ, les six opérations d'édition sont les suivantes:

1. l'*insertion* d'un caractère dans la séquence S;

2. la *suppression* d'un caractère de la séquence S;
3. la *substitution* d'un caractère de la séquence S par un caractère de Σ;
4. l'*insertion* d'un arc (i, j) dans P – les caractères i et j sont également insérés dans la séquence S;
5. la *suppression* d'un arc (i, j) de P – les caractères i et j sont également supprimés de la séquence S;
6. la *substitution* d'un arc (i, j) de P par un arc (k, l) – le caractère i (*resp.* j) de la séquence S est remplacé par le caractère k (*resp.* l) où $k, l \in \Sigma$.

Notons que dans cette extension, les opérations d'édition d'un caractère libre ne peuvent pas être appliquées à un caractère marqué. En d'autres termes, il n'y a pas d'opération permettant de passer directement d'un arc à un caractère et vice-versa.

Afin d'évaluer la distance d'édition, on attribue un coût à chacune de ces opérations. On conserve les coûts définis pour les opérations d'édition de caractères, à savoir: w_d pour une insertion ou une suppression[1] et w_m pour une substitution. Le coût d'une insertion ou d'une suppression d'arc est w_r, celui d'une substitution d'arc est w_{am}.

La notion de transformation est étendue comme suit.

Définition 3.10 (Transformation). Soient (S, P) et (T, Q) deux séquences arc-annotées. Une *transformation* de (S, P) vers (T, Q) est une suite de substitutions, d'insertions et de suppressions de caractères et d'arcs produisant (T, Q) à partir de (S, P).

Pourvus de cette nouvelle définition d'une transformation, la notion de distance d'édition reste inchangée. L'idée innovatrice de cette extension est la prise en compte simultanée des caractères et des arcs: une opération sur un arc a un impact sur les deux caractères marqués par ce dernier.

Par la suite, nous définissons le problème Z-EDIT comme étant le problème consistant à calculer la distance d'édition à la Zhang *et al.* entre deux séquences arc-annotées (S, P) et (T, Q). Comme l'illustre la Table 3.2, la complexité de chacun des sous-problèmes Z-EDIT a été déterminée. Il faut noter que les niveaux de complexité UNLIMITED et CHAIN n'ont pas été considérés par Zhang *et al.* pour le problème Z-EDIT. Malgré tout, par la Propriété 3.2, il est facile de déduire la complexité du problème Z-EDIT pour ces deux niveaux de complexité.

Z-EDIT			
	CROSSING	NESTED	PLAIN
CROSSING	**NP**-complet [99]	$\mathbf{O}(m^2 n \log n)$ [99]	
NESTED		$\mathbf{O}((n-\|P\|).(m-\|Q\|).min\{n-2\|P\|, k_{(S,P)}\}.min\{n-2\|Q\|, k_{(T,Q)}\})$ [98]	
PLAIN			$\mathbf{O}(nm)$ [59]

Table 3.2 – Complexité du problème Z-EDIT où $n = |S|$, $m = |T|$ et $k_{(S,P)}$ (*resp.* $k_{(T,Q)}$) correspond à la hauteur de la séquence (S, P) (*resp.* (T, Q)).

Dans [98], Zhang *et al.* ont proposé un algorithme calculant la distance d'édition entre deux arborescences ordonnées étiquetées (nous ne détaillerons pas dans ce manuscrit ces notions, pour plus de détails sur les arborescences se référer à [98]). La complexité du sous-problème Z-EDIT(NESTED,NESTED) découle directement de cet algorithme. En effet, une modélisation naïve des séquences arc-annotées (de

[1] Dans cette extension, les opérations d'insertion et de suppression ont un coût identique, *i.e.* $w_i = w_d$.

niveaux de complexité au plus NESTED) sous forme d'arbres rend équivalent les deux problèmes. L'algorithme de [98] calculant la distance d'édition entre deux arborescences ordonnées étiquetées T_1, T_2 a une complexité en temps en $\mathbf{O}(|T_1|.|T_2|.min\{leaves_1, depth_1\}.min\{leaves_2, depth_2\})$ où $|T_1|$ (*resp.* $|T_2|$) correspond au nombre de noeuds et de feuilles de l'arbre T_1 (*resp.* T_2), $leaves_1$ (*resp.* $leaves_2$) correspond au nombre de feuilles de l'arbre T_1 (*resp.* T_2) et $depth_1$ (*resp.* $depth_2$) à la hauteur de l'arbre T_1 (*resp.* T_2). Étant donnée une modélisation naïve d'une séquence arc-annotée (S, P) par un arbre T dans laquelle chaque arc de P est représenté par un noeud et chaque caractère libre par une feuille de l'arbre T, on peut montrer que dans cette modélisation la taille de T (*i.e.* $|T|$) est égale à $|S| - |P|$, le nombre de feuilles de T est égal à $n - 2|P|$ et que la hauteur de T est égale à la hauteur de la séquence (S, P). La complexité du sous-problème Z-EDIT(NESTED,NESTED) en est déduite. Les détails de cette preuve sont omis dans ce manuscrit. Pour plus de détails sur la représentation sous forme d'arbres des structures de molécules d'ARN, le lecteur peut se référer à [2, 76, 85].

Extension à la Lin *et al.* [75]. L'extension à la Lin *et al.* [75] est sensiblement la même que celle de Zhang *et al.*, excepté qu'elle prend en compte trois nouvelles opérations d'édition opérant sur les caractères marqués.

Définition 3.11 (Opérations d'édition). Étant donnée une séquence arc-annotée (S, P) où S est une séquence de caractères appartenant à l'alphabet Σ, les neuf opérations d'édition sont les suivantes[2]:

1. l'*insertion* d'un caractère dans la séquence S;
2. la *suppression* d'un caractère de la séquence S;
3. la *substitution* d'un caractère de la séquence S par un caractère de Σ;
4. l'*insertion* d'un arc (i, j) dans P – les caractères i et j sont également insérés dans la séquence S;
5. la *suppression* d'un arc (i, j) de P – les caractères i et j sont également supprimés de la séquence S;
6. la *substitution* d'un arc (i, j) de P par un arc (k, l) – le caractère i (*resp.* j) de la séquence S est remplacé par le caractère k (*resp.* l) où $k, l \in \Sigma$;
7. le *marquage* de deux caractères libres i et j de la séquence S – les caractères libres i et j deviennent marqués et l'arc (i, j) est ajouté à P;
8. la *libération* d'un arc (i, j) de P – les caractères marqués i et j deviennent libres et l'arc (i, j) est ôté de P;
9. l'*altération* d'un arc (i, j) de P – un des caractères marqués parmi i et j devient libre, l'autre est supprimé et l'arc (i, j) est ôté de P.

Afin d'évaluer la distance d'édition, on attribue un coût à chacune de ces opérations. On conserve les coûts définis pour les opérations d'édition d'ores et déjà définies pour le problème Z-EDIT. Les coûts pour les trois nouvelles opérations d'édition sont les suivants: w_b pour une libération d'arc ou le marquage de deux caractères libres et w_a pour une altération d'arc.

Remarquons que si le coût de l'alignement des quatre caractères impliqués dans une substitution d'arc (*resp.* une suppression d'arc) est égal au coût de cette opération, *i.e.* w_{am} (*resp.* w_r), ce n'est pas toujours le cas pour les caractères impliqués par une libération, une altération d'arc ou le marquage de caractères. En effet, le coût d'une libération, d'une altération d'arc ou d'un marquage de caractères

[2] Dans [75], les opérations d'insertion d'un arc et de marquage de deux caractères ne sont pas définies explicitement mais sont prises en compte.

n'inclut pas le coût des substitutions de caractères qui peuvent exister entre les caractères alignés. En conséquence, le coût de l'alignement des quatre caractères concernés par une libération (*resp.* altération) d'arc ou un marquage de caractères est w_b ou $w_b + w_m$ ou $w_b + 2w_m$ (*resp.* w_a ou $w_a + w_m$). L'ensemble des opérations d'édition est illustré en Figure 3.2.

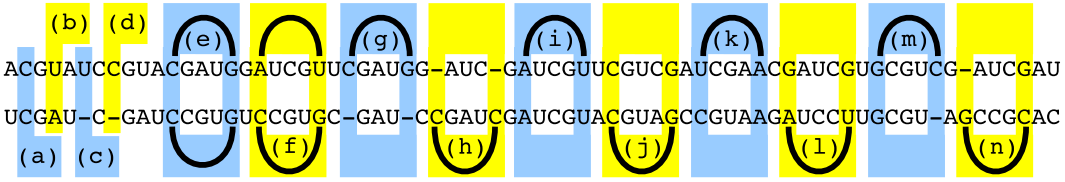


Figure 3.2 – Opérations autorisées dans la résolution du problème L-EDIT: (a) appariement, (b) substitution, (c) suppression, (d) insertion d'un caractère, (e) appariement, (f) substitution, (g) suppression, (h) insertion, (i) libération d'un arc, (j) marquage de deux caractères, (k) libération d'un arc accompagnée d'une substitution de caractère, (l) marquage de deux caractères accompagnée d'une substitution de caractère, (m) altération d'un arc, (n) altération d'un arc accompagnée d'une substitution de caractère.

Les notions de transformation et distance d'édition restent inchangées mais prennent en compte ce nouvel ensemble d'opérations d'édition. Par la suite, nous définissons le problème L-EDIT comme étant le problème consistant à calculer la distance d'édition à la Lin *et al.* entre deux séquences arc-annotées (S, P) et (T, Q). Comme l'illustre la Table 3.3, la complexité de tous les sous-problèmes, excepté un, a été déterminée. Le problème de l'existence d'un algorithme polynomial pour le sous-problème L-EDIT(NESTED,NESTED) est le dernier cas ouvert du problème. Comme nous le prouverons dans le Chapitre 4, le problème L-EDIT(NESTED, NESTED) est **NP**-complet.

L-EDIT				
	UNLIMITED	CROSSING	NESTED	PLAIN
UNLIMITED	**MaxSNP**-dur [75]			
CROSSING		**MaxSNP**-dur [75]		
NESTED			?	**O**(nm^3) [75]
PLAIN				**O**(nm) [84]

Table 3.3 – Complexité du problème L-EDIT avec $n = |S|$, $m = |T|$.

Lin *et al.* ont étudié les complexités classiques et d'approximation du problème. Ils ont prouvé dans [75] que le problème EDIT(CROSSING, NESTED) est **MaxSNP**-dur. D'après la Propriété 3.2, sachant que tout problème **MaxSNP**-dur est également **NP**-dur, ce sous-problème est également **NP**-dur. Lin *et al.* ont, de ce fait, proposé un algorithme de α-approximation avec $\alpha = max\{\frac{2w_a}{w_b+w_r}, \frac{w_b+w_r}{2w_a}\}$ pour ce sous-problème.

3.1.2.2 Le problème LAPCS

Le problème LAPCS [48], acronyme de LONGEST ARC-PRESERVING COMMON SUBSEQUENCE, consiste à rechercher la plus longue sous-structure commune à deux séquences arc-annotées. Ce problème revient donc à inférer la plus grande structure commune à deux molécules d'ARN. L'objectif est d'exhiber une homologie, du point de vue de la structure, entre deux molécules d'ARN.

Afin de définir formellement le problème LAPCS, nous introduisons les notions de *correspondance* et *sous-séquence commune* arc-préservantes entre deux séquences arc-annotées.

Définition 3.12 (Correspondance arc-préservante). Étant données deux séquences arc-annotées (S, P) et (T, Q) définies sur l'alphabet Σ, une *correspondance arc-préservante* $\mathcal{C}$ est un ensemble de paires de positions (s_i, t_i) de S et T tel que $\forall (s_i, t_i), (s_j, t_j) \in \mathcal{C}$, la correspondance $\mathcal{C}$:

- **est univoque:** si $s_i = s_j$ alors $t_i = t_j$;
- **préserve l'ordre:** si $s_i < s_j$ alors $t_i < t_j$;
- **est arc-préservante:** $(s_i, s_j) \in P$ si et seulement si $(t_i, t_j) \in Q$;
- **préserve les caractères:** $S[s_i] = T[t_i]$ et $S[s_j] = T[t_j]$.

Définition 3.13 (Sous-séquence commune arc-préservante). Étant donnée une correspondance arc-préservante $\mathcal{C} = \{(s_1, t_1), (s_2, t_2), \ldots (s_{|\mathcal{C}|}, t_{|\mathcal{C}|})\}$ entre deux séquences arc-annotées (S, P) et (T, Q) définies sur l'alphabet Σ, une séquence arc-annotée (U, R) est une *sous-séquence commune arc-préservante* de (S, P) et (T, Q) si:

- $\forall (s_i, t_i) \in \mathcal{C}$, $U[i] = S[s_i]$;
- $\forall (s_i, t_i), (s_j, t_j) \in \mathcal{C}$, $(i, j) \in R$ si et seulement si $(s_i, s_j) \in P$.

Le problème d'optimisation LAPCS se définit à l'aide de ces deux notions comme suit:

LONGEST ARC-PRESERVING COMMON SUBSEQUENCE (LAPCS)
DONNÉES*: Deux séquences arc-annotées (S, P) et (T, Q) définies sur un alphabet Σ.*
QUESTION*: Trouver la sous-séquence commune arc-préservante de (S, P) et (T, Q) de longueur (en terme de séquence) maximale.*

Dans la suite de cette section, soient (S, P) et (T, Q) deux séquences arc-annotées. La Table 3.4 présente les complexités des sous-problèmes de LAPCS en fonction des niveaux de complexité: UNLIMITED, CROSSING, NESTED, CHAIN et PLAIN.

LAPCS					
	UNLIMITED	CROSSING	NESTED	CHAIN	PLAIN
UNLIMITED	**NP**-complet [48]				
CROSSING		**NP**-complet [48]			
NESTED			**NP**-complet [74]	$\mathbf{O}(nm^3)$ [65]	
CHAIN				$\mathbf{O}(nm)$ [48]	
PLAIN					$\mathbf{O}(nm)$[63]

Table 3.4 – Complexités des sous-problèmes de LAPCS en fonction des différents niveaux de complexité.

Hirschberg [63] a proposé un algorithme de complexité en temps en $\mathbf{O}(|S_1|.|S_2|)$ pour rechercher la plus longue sous-séquence commune à deux séquences S_1 et S_2 sur un alphabet donné. La complexité du sous-problème LAPCS(PLAIN,PLAIN) en découle.

Afin de contourner la **NP**-complétude de certains des sous-problèmes, Evans a proposé quatre algorithmes de complexité paramétrée:

1. un algorithme en $\mathbf{O}(9^c n.m)$ pour LAPCS(CROSSING,CROSSING) où c est la largeur de coupe des séquences arc-annotées;
2. un algorithme en $\mathbf{O}(c^2 4^c nm)$ pour LAPCS(NESTED,NESTED);
3. un algorithme en $\mathbf{O}(9^b nm)$ pour LAPCS(CROSSING,CROSSING) où b est la largeur de bande des séquences arc-annotées;
4. un algorithme en $\mathbf{O}(b^2 4^b nm)$ pour LAPCS(NESTED,NESTED).

De plus, Evans a poursuivi l'étude de complexité paramétrée du problème LAPCS en proposant deux preuves de **W[1]**-complétude. La Table 3.5 présente les complexités paramétrées des sous-problèmes de LAPCS en fonction des différents niveaux de complexité.

LAPCS					
	UNLIMITED	CROSSING	NESTED	CHAIN	PLAIN
UNLIMITED	**W[1]**-complet pour L				
CROSSING		**W[1]**-complet pour L; $\mathbf{O}(9^x nm)$	$\mathbf{O}(9^x nm)$		
NESTED			$\mathbf{O}(x^2 4^x nm)$	-	
CHAIN				-	
PLAIN					-

Table 3.5 – Complexités paramétrées, issues de [48], des sous-problèmes de LAPCS en fonction des différents niveaux de complexité. L représente la longueur de la séquence désirée et $x \in \{b, c\}$.

En 2000 et 2001, Jiang *et al.* et Lin *et al.* [65, 74] ont poursuivi l'étude de Evans en complétant la table de complexité du problème et en proposant une étude sur l'approximation du problème. Les résultats de cette dernière étude sont illustrés dans la Table 3.6.

LAPCS					
	UNLIMITED	CROSSING	NESTED	CHAIN	PLAIN
UNLIMITED	Non-approximable avec un ratio n^ϵ tel que $\epsilon \in (0, \frac{1}{4})$				
CROSSING		**MaxSNP**-dur ; 2-approximable			
NESTED			2-approximable	-	
CHAIN				-	
PLAIN					-

Table 3.6 – Complexités d'approximation, issues de [65], des sous-problèmes de LAPCS.

3.1.2.3 Le problème APS

Le problème APS, acronyme de ARC-PRESERVING SUBSEQUENCE, consiste à trouver une occurrence d'une sous-structure (appelée le motif) dans une séquence arc-annotée. Ce problème revient à détecter la présence d'un motif structurel dans une molécule d'ARN. On le rencontre couramment lors de la recherche d'un motif structurel d'une molécule d'ARN dans une base de données de molécules d'ARN [58]. De plus, du point de vue théorique, le problème APS peut être vu comme une version restreinte du problème LAPCS, et par conséquent, a des applications dans la comparaison de structures de molécules d'ARN [48, 99].

Le problème de décision APS se définit facilement à l'aide de la notion de sous-séquence commune arc-préservante:

ARC-PRESERVING SUBSEQUENCE (APS)
DONNÉES*: Deux séquences arc-annotées* (S, P) *et* (T, Q) *définies sur un alphabet* Σ *telles que* $|T| \leq |S|$.
QUESTION*: Existe-t-il une sous-séquence commune arc-préservante de* (S, P) *et* (T, Q) *de longueur* $|T|$*?*

On peut formuler également ce problème indépendamment de la notion de sous-séquence commune arc-préservante comme suit.

ARC-PRESERVING SUBSEQUENCE (APS)
DONNÉES*: Deux séquences arc-annotées* (S, P) *et* (T, Q) *définies sur un alphabet* Σ *telles que* $|T| \leq |S|$.
QUESTION*:* (T, Q) *peut-elle être obtenue à partir de* (S, P) *en supprimant certaines de ses bases ainsi que les éventuels arcs incidents à ces dernières?*

Dans la suite de cette section, soient (S, P) et (T, Q) deux séquences arc-annotées telles que $|S| \geq |T|$. Le problème APS a été intensivement étudié par le passé [48, 57, 58]. Comme l'illustre la Table 3.7, la complexité de tous les sous-problèmes, excepté un, a été déterminée. La seule complexité jusqu'alors inconnue est celle du sous-problème APS(CROSSING, PLAIN). Comme nous le prouverons dans le Chapitre 5, le sous-problème APS(CROSSING, PLAIN) est **NP**-complet.

APS					
	UNLIMITED	CROSSING	NESTED	CHAIN	PLAIN
UNLIMITED	**NP**-complet [48]				
CROSSING		**NP**-complet [48]	**NP**-complet [58]		?
NESTED			$\mathbf{O}(nm)$ [57]		
CHAIN				$\mathbf{O}(nm)$ [57]	$\mathbf{O}(n+m)$ [57]

Table 3.7 – Complexité du problème APS.

Contrairement au problème LAPCS, il n'existe pas d'études de la complexité paramétrée et de l'approximation du problème APS. En effet, la notion d'approximation ne concerne que les problèmes d'optimisation et les résultats de complexité paramétrée du problème LAPCS peuvent être transposés au problème APS.

3.2 Les 2-intervalles

Le problème de l'élaboration d'une représentation générale de motifs structuraux, *i.e.* de *descripteurs macroscopiques* de séquences arc-annotées, a été considéré par Stéphane Vialette dans [93, 94]. L'approche adoptée a été de définir des *descripteurs géométriques* des arcs en faisant abstraction de la nature

des caractères de la séquence. Pour ce faire, Vialette a utilisé une généralisation naturelle des intervalles, nommée 2-*intervalle*.

Dans cette section, après avoir introduit les notations relatives aux 2-intervalles, nous présenterons les deux problèmes d'homologie entre molécules d'ARN, introduits par Vialette [93, 94], à savoir: (1) la recherche de la plus grande sous-famille respectant une structure donnée et (2) la recherche de motifs structuraux.

3.2.1 Notations et modélisation de molécules d'ARN

Notations. Afin de définir la notion de 2-intervalle, nous débutons cette section par la présentation de la notion d'*intervalle*.

Définition 3.14 (Intervalle). Un *intervalle* $I = [x : y]$ est l'ensemble des points sur la droite réelle compris entre les deux points extrémités x et y.

Étant donnés deux intervalles $I = [x : y]$ et $J = [x' : y']$, nous définissons deux relations de comparabilité illustrées en Figure 3.3:

1. la **précédence** (notée $<$): $I < J$ si et seulement si $y < x'$;
2. l'**inclusion** (notée $\subseteq$): $I \subseteq J$ si et seulement si $x' < x$ et $y < y'$.

Deux intervalles incomparables par précédence ou inclusion sont dits *chevauchants*.

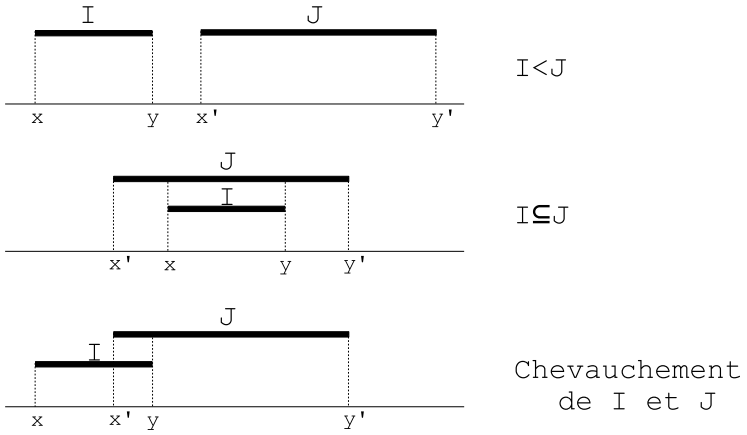


Figure 3.3 – Relations de comparabilité entre intervalles.

Définition 3.15 (Intervalles disjoints). Deux intervalles I et J sont *disjoints* si il n'existe pas de point de la droite réelle appartenant à la fois à I et à J – *i.e.* $I < J$ ou $J < I$. On notera alors $I \bigcap J = \emptyset$.

Définition 3.16 (2-intervalle). Un *2-intervalle* est l'union disjointe de deux intervalles. Nous noterons un 2-intervalle par $D = (I, J)$ où I et J sont des intervalles tels que $I < J$.

Définition 3.17 (2-intervalles comparables). Deux 2-intervalles $D_1 = (I_1, J_1)$ et $D_2 = (I_2, J_2)$ sont *comparables* si I_1, I_2, J_1, J_2 sont disjoints deux à deux (*i.e.* les intervalles impliqués ne se chevauchent pas).

Étant donnés deux 2-intervalles comparables $D_1 = (I_1, J_1)$ et $D_2 = (I_2, J_2)$, il est facile de vérifier que D_1 et D_2 sont dans l'une des situations illustrées en Figure 3.4 et définies formellement par l'une des trois relations suivantes:

- $D_1 < D_2$ si $I_1 < J_1 < I_2 < J_2$;
- $D_1 \sqsubset D_2$ si $I_2 < I_1 < J_1 < J_2$;
- $D_1 \bowtie D_2$ si $I_1 < I_2 < J_1 < J_2$.

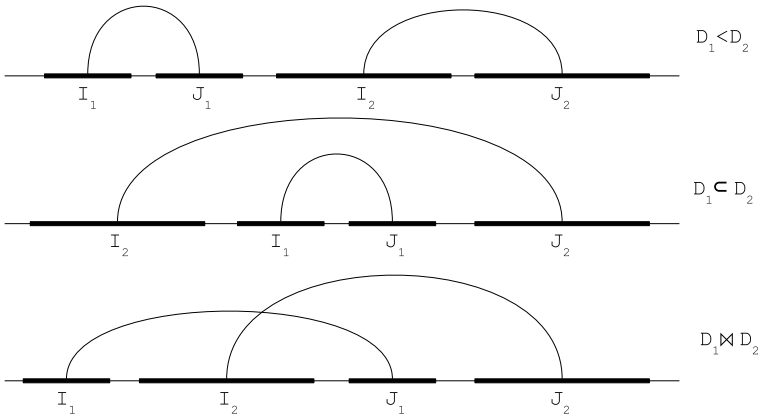


Figure 3.4 – Relations de comparabilité entre 2-intervalles.

Notons que les relations $<$ et $\sqsubset$ sur les 2-intervalles sont transitives: *i.e.* si $D_1 < D_2$ (*resp.* $D_1 \sqsubset D_2$) et $D_2 < D_3$ (*resp.* $D_2 \sqsubset D_3$) alors $D_1 < D_3$ (*resp.* $D_1 \sqsubset D_3$).

Deux 2-intervalles D_1 et D_2 sont *τ-comparables* pour un $\tau \in \{<, \sqsubset, \bowtie\}$ si $D_1 \tau D_2$ ou $D_2 \tau D_1$. Pourvus de cette notion, nous définissons celle de *modèle de comparaison*.

Définition 3.18 (Modèle de comparaison). Soient $\mathcal{D}$ un ensemble de 2-intervalles et $R \subseteq \{<, \sqsubset, \bowtie\}$ un ensemble non vide. L'ensemble $\mathcal{D}$ est *R-comparable* si pour tout couple de 2-intervalles distincts (D_1, D_2) de $\mathcal{D}$, D_1 et D_2 sont τ-comparables pour un $\tau \in R$. Tout sous-ensemble non vide R est appelé *modèle de comparaison*.

D'après cette définition, si un ensemble de 2-intervalles $\mathcal{D}$ est R-comparable alors $\mathcal{D}$ est un ensemble de 2-intervalles disjoints.

Définition 3.19 (Support). Le *support* d'un ensemble de 2-intervalles $\mathcal{D}$, noté $\mathsf{Support}(\mathcal{D})$, est l'ensemble de tous les intervalles impliqués dans $\mathcal{D}$.

Dans [94], Vialette a proposé deux restrictions sur le support:

1. tous les intervalles du support sont de taille identique;
2. tous les intervalles du support sont disjoints deux à deux, *i.e.* si deux intervalles $I, I' \in \mathsf{Support}(\mathcal{D})$ se chevauchent, alors $I = I'$.

À partir de ces restrictions sur le support, nous pouvons définir un ensemble de restrictions sur la complexité structurelle d'un ensemble de 2-intervalles, et donc sur la complexité même des problèmes utilisant des 2-intervalles. Les deux restrictions sur le support induisent les trois niveaux de complexité suivants illustrés en Figure 3.5:

- UNLIMITED: aucune restriction;
- UNITARY: restriction 1;

- DISJOINT: restrictions 1 et 2.

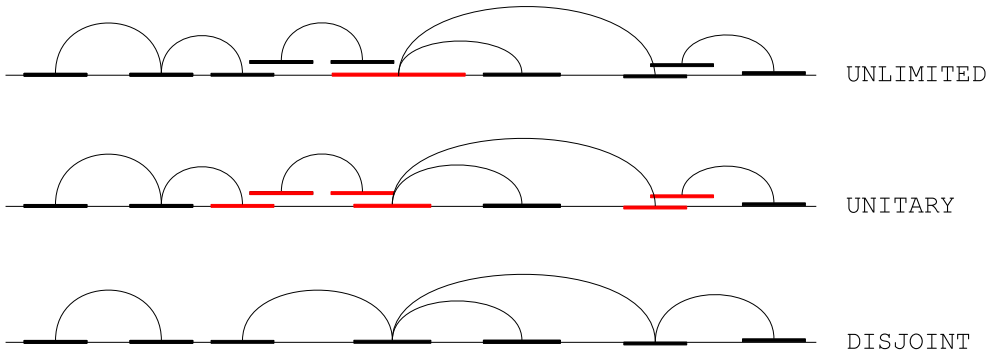


Figure 3.5 – Exemple d'ensemble de 2-intervalles respectant les différents niveaux de complexité de support: UNLIMITED, UNITARY et DISJOINT.

Étant donnés deux niveaux de complexité n_1 et n_2, on note $n_1 \subset n_2$ si tout ensemble de 2-intervalles respectant le niveau de complexité n_1 respecte également le niveau de complexité n_2. À l'instar des séquences arc-annotées, nous définissons une unique hiérarchie de niveaux de complexité pour les 2-intervalles définie à l'aide de la notion de support.

Propriété 3.3 (Hiérarchie des niveaux de complexité). DISJOINT $\subset$ UNITARY $\subset$ UNLIMITED.

Le bien-fondé de la Propriété 3.3 découle des définitions des niveaux de restriction de support. Étant donnés un ensemble de 2-intervalles $\mathcal{D}$, un problème $\mathcal{P}$ prenant en entrée $\mathcal{D}$ et un modèle de comparaison $R \subseteq \{<, \sqsubset, \bowtie\}$, on note $\mathcal{P}(n_1, R)$ le sous-problème de $\mathcal{P}$ où $\mathcal{D}$ respecte le niveau de complexité de support n_1 et le résultat de $\mathcal{P}(n_1, R)$ est un sous-ensemble de $\mathcal{D}$ R-comparable.

Soient $\mathcal{P}(n_1, R)$ et $\mathcal{P}(n_2, R)$ deux sous-problèmes de $\mathcal{P}$, on note $\mathcal{P}(n_1, R) \subset \mathcal{P}(n_2, R)$ si $n_1 \subset n_2$. Cette définition et la Propriété 3.3 permettent de définir également une hiérarchie entre les différents sous-problèmes de $\mathcal{P}$ comme illustrée dans la Table 3.8.

$\mathcal{P}(\text{UNLIMITED}, R)$	$\supset$	$\mathcal{P}(\text{UNITARY}, R)$	$\supset$	$\mathcal{P}(\text{DISJOINT}, R)$

Table 3.8 – Hiérarchie entre les différents sous-problèmes de $\mathcal{P}$ en fonction des niveaux de complexité du support, $\forall R \subseteq \{<, \sqsubset, \bowtie\}$.

Pourvus de cette hiérarchie entre les différents sous-problèmes de $\mathcal{P}$, nous en déduisons une propriété sur la propagation de la complexité du problème $\mathcal{P}$.

Propriété 3.4. *Soient $\mathcal{P}(n_1, R)$ et $\mathcal{P}(n_2, R)$ deux sous-problèmes de $\mathcal{P}$ tels que*

1. *n_1 et n_2 appartiennent à $\{\text{DISJOINT}, \text{UNITARY}, \text{UNLIMITED}\}$;*
2. *$R \subseteq \{<, \sqsubset, \bowtie\}$;*
3. *$\mathcal{P}(n_1, R) \subset \mathcal{P}(n_2, R)$.*

Si le problème $\mathcal{P}(n_1, R)$ est **NP***-complet (resp. le problème $\mathcal{P}(n_2, R)$ est polynomial) alors le problème $\mathcal{P}(n_2, R)$ (resp. $\mathcal{P}(n_1, R)$) l'est également.*

Modélisation de molécules d'ARN. Partant du constat que pour de nombreuses molécules d'ARN, la structure est mieux conservée durant l'évolution que la séquence de nucléotides [32], Vialette a étudié, dans [93, 94], le problème de l'élaboration d'une représentation générale de motifs structuraux, *i.e.* de *descripteurs macroscopiques* des structures secondaires d'ARN. L'approche adoptée a été de définir des *descripteurs géométriques* d'hélices à l'aide des 2-intervalles. Les propriétés géométriques des 2-intervalles peuvent permettre une meilleure compréhension de la complexité du problème de recherche de motifs structurés dans des séquences d'ARN.

Intuitivement, chaque 2-intervalle représente une hélice, *i.e.* une suite continue de liaisons entre nucléotides complémentaires. Plus précisément, étant donné un 2-intervalle $D = (I, J)$ représentant une hélice, I (*resp.* J) représente la première (*resp.* seconde) suite continue de nucléotides, en parcourant le brin d'ARN dans le sens $5' - 3'$, constituant l'hélice, comme illustré en Figure 3.6. Ainsi un ensemble de 2-intervalles représente une structure d'ARN en faisant abstraction de la nature des nucléotides (les longueurs des régions simples brins et des hélices peuvent être modélisées dans cette représentation en faisant varier l'espace entre deux intervalles et la taille de ces derniers). La contrainte de planarité des structures secondaires d'ARN induit le lemme suivant.

Lemme 3.3. *Toute structure secondaire sans pseudo-noeud peut être modélisée à l'aide d'un ensemble de 2-intervalles* $\{<, \sqsubset\}$*-comparable respectant le niveau de complexité de support* DISJOINT.

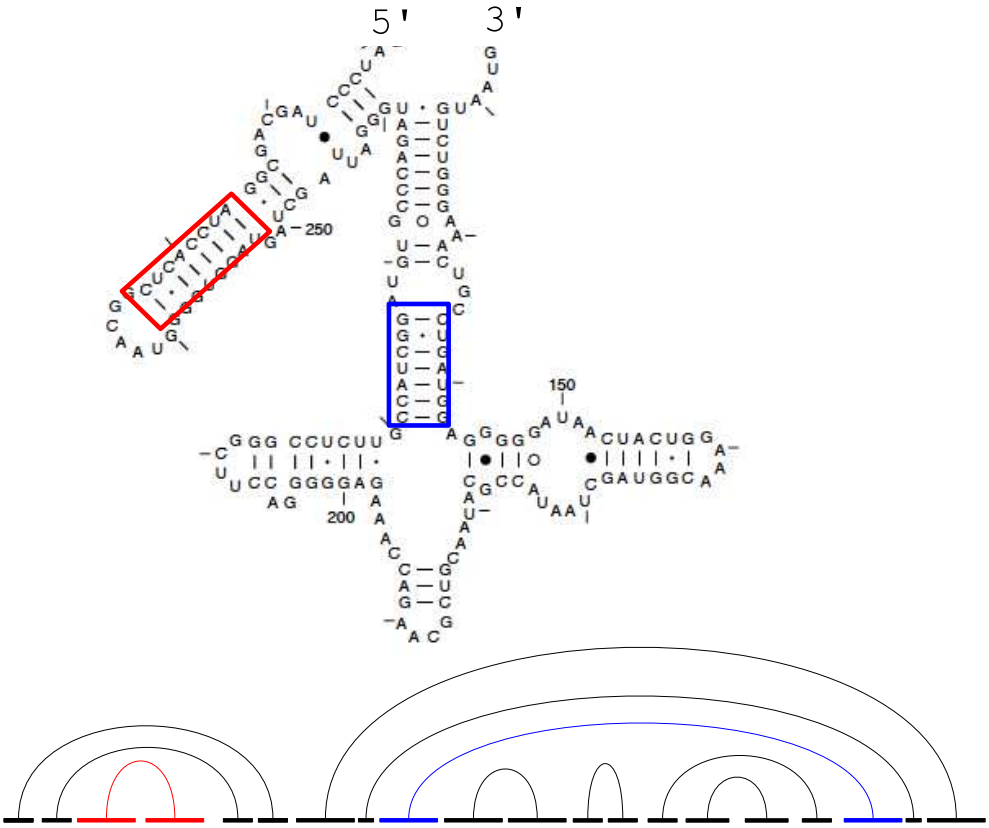


Figure 3.6 – Exemple de représentation de la structure de l'ARN ribosomal 16s d'Escherichia Coli [60] au moyen de 2-intervalles (vue partielle). Par souci de lisibilité, les longueurs des régions simples brins n'ont pas été respectées.

3.2.2 Quelques problèmes utilisant les 2-intervalles

Dans cette section, nous allons présenter les deux problèmes, utilisant la modélisation des structures de molécules d'ARN à l'aide de 2-intervalles, introduits par Vialette [93, 94]: (1) la recherche du plus grand sous-ensemble de 2-intervalles R-comparable, où R est un modèle de comparaison donné dans l'instance (le problème 2-IP) et (2) la recherche d'un *motif* (notion définie en Section 3.2.2.2) dans un ensemble de 2-intervalles pour différents modèles de comparaison (le problème PMO-2).

3.2.2.1 Le problème 2-IP

Le problème 2-IP, acronyme de 2-INTERVAL PATTERN, consiste à rechercher le plus grand sous-ensemble de 2-intervalles R-comparable pour un modèle R donné. Dans le contexte de l'étude de la structure des molécules d'ARN, si on modélise les structures de molécules d'ARN à l'aide de 2-intervalles alors ce problème revient à rechercher une sous-structure sans tenir compte de la nature des nucléotides dans une molécule d'ARN. Comme nous l'avons précisé précédemment, ce problème est rencontré couramment lors de la recherche d'une sous-structure d'une molécule d'ARN dans une base de données de molécules d'ARN [58]. L'intérêt du problème 2-IP réside dans l'abstraction de la nature des nucléotides.

2-INTERVAL PATTERN (2-IP)
DONNÉES*: Un ensemble de 2-intervalles $\mathcal{D}$, un modèle de comparaison $R \subseteq \{<, \sqsubset, \bowtie\}$ et un entier positif k.*
QUESTION*: Existe-il un sous-ensemble $\mathcal{D}' \subseteq \mathcal{D}$ de cardinalité supérieure ou égale à k, tel que $\mathcal{D}'$ soit R-comparable ?*

Comme l'illustre la Table 3.9, la complexité de tous les sous-problèmes, exceptés quatre, a été déterminée. Il subsiste donc quatre problèmes ouverts: les sous-problèmes 2-IP($n_1, \{<, \bowtie\}$) pour tout $n_1 \in \{\text{DISJOINT}, \text{UNITARY}, \text{UNLIMITED}\}$ et 2-IP(DISJOINT,$\{\sqsubset, \bowtie\}$). Dans le Chapitre 6, nous en résoudrons trois.

2-IP			
	SUPPORT		
MODÈLE	UNLIMITED	UNITARY	DISJOINT
$\{<, \sqsubset, \bowtie\}$	**NP**-complet [94]		$\mathbf{O}(n\sqrt{n})$[78]
$\{\sqsubset, \bowtie\}$	**NP**-complet [94]		?
$\{<, \sqsubset\}$	$\mathbf{O}(n^2)$ [94]		
$\{<, \bowtie\}$	?		
$\{<\}$	$\mathbf{O}(n \log n)$ [94]		
$\{\sqsubset\}$	$\mathbf{O}(n^2)$ [94]		
$\{\bowtie\}$	$\mathbf{O}(n^2 \log n)$ [94]		

Table 3.9 – Complexité du problème 2-IP avec $n = |\mathcal{D}|$.

Dans [94], Vialette a initié une étude sur l'approximation du problème qui a été par la suite poursuivie dans [7] et [42]. La Table 3.10 présente l'ensemble de ces résultats.

2-IP			
	SUPPORT		
MODÈLE	UNLIMITED	UNITARY	DISJOINT
$\{<, \sqsubset, \bowtie\}$	**APX**-dur [7] ; 4-approximable[94]	**APX**-dur [7] ; 3-approximable [42]	polynomial
$\{\sqsubset, \bowtie\}$	**APX**-dur [7] ; 4-approximable [42]	**APX**-dur [7] ; 3-approximable [42]	polynomial
$\{<, \sqsubset\}$	polynomial		
$\{<, \bowtie\}$	6-approximable [42]	3-approximable [42]	2-approximable [42]
$\{<\}$	polynomial		
$\{\sqsubset\}$	polynomial		
$\{\bowtie\}$	polynomial		

Table 3.10 – Complexité de l'approximation du problème 2-IP.

3.2.2.2 Le problème PMO-2

Le problème PMO-2, acronyme de PATTERN MATCHING OVER SET OF 2-INTERVALS, consiste à rechercher un *motif* dans un ensemble de 2-intervalles pour différents modèles de comparaison. Contrairement au problème 2-IP, le motif recherché est connu et est fourni dans l'instance. Avant de présenter formellement le problème, nous définissons la notion de motif de 2-intervalles.

Définition 3.20 (Motif de 2-intervalles). Un *motif de 2-intervalles* est un mot composé de caractères provenant d'un alphabet Σ tel que chaque caractère présent dans le motif apparaît exactement deux fois.

Par exemple, le mot $abaccb$ est un motif de 2-intervalles. Tout motif de 2-intervalles peut être associé à un ensemble de 2-intervalles respectant un modèle de comparaison précis. En effet, il est possible d'affecter un 2-intervalle pour chaque couple de caractères identiques composant un motif de 2-intervalles: chaque caractère correspond à un intervalle et deux caractères identiques sont associés pour former un 2-intervalle. Par exemple, le motif $abaccb$ représente un ensemble de 2-intervalles $\{<, \sqsubset, \bowtie\}$-comparable tandis que le motif $abbacc$ représente un ensemble de 2-intervalles $\{<, \sqsubset\}$-comparable. Le problème PMO-2 se définit comme suit.

PATTERN MATCHING OVER SET OF 2-INTERVALS (PMO-2)
DONNÉES*: Un ensemble de 2-intervalles $\mathcal{D}$, un motif p respectant un modèle de comparaison $R \subseteq \{<, \sqsubset, \bowtie\}$.*
QUESTION*: Existe-il une occurrence de p dans $\mathcal{D}$?*

Comme l'illustre la Table 3.11, la complexité de tous les sous-problèmes de PMO-2 a été déterminée.

Dans ce chapitre, nous avons abordé deux des nombreux formalismes permettant de représenter les structures des molécules d'ARN: les séquences arc-annotées et les 2-intervalles. Pour chacun de ces formalismes nous avons présenté un ensemble de problèmes associés. Globalement, on peut constater que, dans le cas des séquences arc-annotées, dès lors qu'une des deux séquences en entrée respecte le

Problème PMO-2			
	SUPPORT		
MODÈLE	UNLIMITED	UNITARY	DISJOINT
$\{<, \sqsubset, \bowtie\}$	**NP**-complet [94]		
$\{\sqsubset, \bowtie\}$	**NP**-complet [94]		
$\{<, \sqsubset\}$	$\mathbf{O}(mn^3 \log n)$ [94]		
$\{<, \bowtie\}$	$\mathbf{O}(n^6 m^2)$[56]		
$\{<\}$	$\mathbf{O}(n \log n)$ [94]		
$\{\sqsubset\}$	$\mathbf{O}(n^2)$ [94]		
$\{\bowtie\}$	$\mathbf{O}(n^2 \log n)$ [94]		

Table 3.11 – Complexité du problème PMO-2 avec $n = |\mathcal{D}|$ et m est la taille du motif.

niveaux CROSSING les trois problèmes que sont L-EDIT, LAPCS et APS sont **NP**-complets. En ce qui concerne les 2-intervalles, si le modèle de comparaison souhaité contient au moins les relations $\sqsubset$ et $\bowtie$ alors les deux problèmes présentés sont **NP**-complets.

Comme indiqué dans les tables de complexité de ce chapitre, il reste des cas ouverts pour les problèmes L-EDIT, APS et 2-IP. Dans les chapitres 4, 5 et 6, nous déterminerons la complexité de la majorité des cas ouverts en proposant deux types de résultats: des algorithmes polyomiaux et des preuves de **NP**-complétude. Finalement, dans le Chapitre 7, nous présenterons des résultats qui différent un peu du reste de cette partie puisqu'ils concernent le design d'ARN. Plus précisément, nous nous intéresserons au design de molécules d'ARN codant pour des sélénoprotéines (*i.e.* des protéines contenant le $21^{ème}$ acide aminé – la sélénocystéine). Ce problème, connu sous le nom de problème MRSO, a été formalisé dans [3, 4] par Backofen *et al.*

CHAPITRE 4

Le problème L-EDIT

4.1 Introduction

Dans ce chapitre, nous présentons des résultats publiés dans [23] et obtenus en collaboration avec Guillaume Fertin, Irena Rusu et Christine Sinoquet.

Nous rappelons que le problème L-EDIT(T_1,T_2) a pour objet le calcul du nombre minimum d'opérations d'édition nécessaires pour transformer une molécule d'ARN ayant une structure de type T_1 en une molécule d'ARN ayant une structure de type T_2 où T_1,T_2 prennent des valeurs dans {PLAIN, CHAIN,NESTED, CROSSING, UNLIMITED}. Lin *et al.* [75] ont proposé de tenir compte simultanément des structures secondaires et tertiaires dans la comparaison de molécules d'ARN en considérant les liens hydrogènes et leurs bases incidentes conjointement dans le calcul de la similarité.

Ils ont proposé dans [75] des algorithmes polynomiaux exacts et des algorithmes d'approximation pour certaines classes de problèmes L-EDIT(T_1,T_2). Ils ont également donné quelques preuves de **NP**-complétude pour d'autre classes. Pour autant, l'existence d'un algorithme polynomial exact pour le sous-problème L-EDIT(NESTED, NESTED) était, jusqu'à présent, le dernier cas ouvert. Dans ce chapitre, nous démontrons que le sous-problème L-EDIT(NESTED, NESTED) est un problème **NP**-complet– complétant ainsi l'étude du problème L-EDIT introduit par Lin *et al.* [75] comme l'illustre la Table 4.1.

L-EDIT				
	UNLIMITED	CROSSING	NESTED	PLAIN
UNLIMITED	**MaxSNP**-dur [75]			
CROSSING		**MaxSNP**-dur [75]		
NESTED			**NP-complet [23]**	$\mathbf{O}(nm^3)$ [75]
PLAIN				$\mathbf{O}(nm)$ [84]

Table 4.1 – Complexité du problème L-EDIT avec $n = |S|$, $m = |T|$.

Nous allons débuter ce chapitre par la présentation d'un ensemble de notions de théorie des graphes qui nous seront utiles pour prouver la **NP**-complétude du problème. Un *graphe* $G = (V, E)$ est constitué d'un ensemble fini de sommets V et d'un ensemble E de couples de sommets de V. Un élément de E est appelé *arête*. Une arête est dite *incidente* aux sommets qu'elle relie. Le *degré* d'un sommet est égal au nombre d'arêtes incidentes à ce sommet. On appelle *chaîne* entre deux sommets u et v de G, une suite de sommets $v_1, v_2, \ldots, v_k$ telle que $u = v_1$, $v = v_k$ et $\forall 1 \leq i \leq k$, $(v_i, v_{i+1}) \in E$.

Définition 4.1 (Graphe cubique connexe planaire sans isthme). Un graphe $G = (V, E)$ est dit *cubique connexe planaire sans isthme* si:

- tout sommet de V est de degré trois (cubique);

- il est possible de représenter G dans le plan sans croisement d'arêtes (planaire);
- il existe une chaîne reliant tout couple de sommets de V (connexe);
- la suppression d'une arête quelconque de E ne supprime pas la connexité de G (sans isthme).

Définition 4.2 (Plongement en deux pages). Un *plongement en deux pages* d'un graphe G est un couple formé:

- d'un ordonnancement des sommets de G suivant une ligne D;
- d'une affectation de chacune des arêtes de G à un des deux demi-plans délimités par D tels que les arêtes assignées à un même demi-plan puissent être dessinées sans croisement.

Chaque demi-plan est appelé *page*.

Définition 4.3 (Ensemble indépendant). Un *ensemble indépendant* de sommets d'un graphe $G = (V, E)$ est un ensemble $V' \subseteq V$ de sommets de V tel qu'il n'existe pas d'arête dans E reliant deux sommets de V'.

4.2 NP-complétude du sous-problème L-EDIT(NESTED, NESTED)

Comme annoncé précédemment, le résultat principal de ce chapitre est le suivant:

Théorème 4.1. *Le problème* L-EDIT(NESTED, NESTED) *est* **NP***-complet.*

Nous allons considérer le problème de décision suivant:

MIS-3P
DONNÉES*: Un graphe cubique connexe planaire sans isthme G et un entier k.*
QUESTION*: Peut-on trouver un ensemble indépendant de sommets de G de cardinalité supérieure ou égale à k?*

On rapelle ci-dessous la définition formelle du problème L-EDIT(NESTED, NESTED). Rappelons également que le score d'un alignement est égal à la somme des coûts des opérations de transformation induites par l'alignement.

L-EDIT(NESTED, NESTED)
DONNÉES*: Deux séquences arc-annotées (S, P) et (T, Q) respectant le niveaux* NESTED*, un ensemble de coûts pour les opérations sur les bases (w_m, w_d) et sur les arcs (w_{am}, w_b, w_a, w_r), un entier s'.*
QUESTION*: Peut-on trouver un alignement des deux séquences (S, P) et (T, Q) tel que le score induit soit inférieur ou égal à s'?*

Remarquons que le problème L-EDIT(NESTED, NESTED) appartient à la classe **NP**. En effet, étant donné un alignement, il est possible de vérifier en temps polynomial si le score induit par l'alignement est inférieur ou égal à s'. Afin de prouver la **NP**-complétude du sous-problème L-EDIT(NESTED, NESTED), nous proposons une transformation polynomiale à partir du problème **NP**-complet MIS-3P [14]. Nous détaillons cette transformation ci-après.

Soit $G = (V, E)$ un graphe cubique connexe planaire sans isthme avec $V = \{v_1, v_2, \dots v_n\}$ (un exemple est donné dans la Figure 4.1(a)). La construction de l'instance du problème L-EDIT(NESTED, NESTED) s'effectue en deux temps.

Dans un premier temps, nous construisons un *plongement en deux pages* de G tel que sur chaque page, tout sommet est de degré non nul (*cf.* Figure 4.1(b)). Nous notons *B-plongement* tout plongement de ce type. Bernhart *et al.* [13] ont démontré qu'il existe toujours un B-plongement d'un graphe cubique connexe planaire sans isthme et qu'il peut être trouvé en temps polynomial. Par la suite, nous désignerons la première (*resp.* seconde) page par $page_1$ (*resp.* $page_2$) et nous supposerons que les sommets v_i, $\forall 1 \leq i \leq n$, du plongement sont ordonnés dans l'ordre croissant.

Dans un second temps, nous construisons une séquence S (*resp.* T) à partir de $page_1$ (*resp.* $page_2$) comme suit:

$$S = S_c\ S_{v_1}\ S_c\ S_{v_2}\ S_c\ S_{v_3} \dots\ S_c\ S_{v_n}\ S_c$$
$$T = T_c\ T_{v_1}\ T_c\ T_{v_2}\ T_c\ T_{v_3} \dots\ T_c\ T_{v_n}\ T_c$$

Nous détaillons ci-après les éléments composant S et T. Pour tout $v_i \in V$, si v_i est de degré deux dans $page_1$ alors $S_{v_i} = UAUAGG$ sinon $S_{v_i} = GGUAUA$. De façon similaire, pour tout $v_i \in V$, si v_i est de degré deux dans $page_2$ alors $T_{v_i} = UAUAGG$ sinon $T_{v_i} = GGUAUA$.

Notons que G étant cubique, pour tout sommet de G, si $S_{v_i} = UAUAGG$ (*resp.* $S_{v_i} = GGUAUA$) alors $T_{v_i} = GGUAUA$ (*resp.* $T_{v_i} = UAUAGG$). Chaque segment S_c ou T_c est construit à partir d'un nombre q determiné de bases C, avec $q > \frac{3nw_r}{w_d}$ (la valeur de q sera justifiée dans la preuve du Lemme 4.2).

Nous obtenons les deux séquences S et T de longueur $6n + (n+1)q$: chacun des n segments S_{v_i} est composé de 6 caractères et chacun des $n+1$ segments S_c est composé de q caractères.

Afin d'obtenir les deux séquences arc-annotées (S, P) et (T, Q) respectant le niveau NESTED et composant l'instance du problème L-EDIT(NESTED, NESTED), nous définissons ci-après les ensembles d'arcs P et Q à partir de la configuration des arêtes de chaque page. L'instance finale du sous-problème L-EDIT(NESTED, NESTED) obtenue est illustrée sur un exemple en Figure 4.1.

Nous commençons par détailler le mécanisme permettant d'obtenir l'ensemble d'arcs P. Pour chaque arête (v_i, v_j) de $page_1$, on crée deux arcs $a = (k, l)$ et $a' = (k+1, l-1)$ où k et l sont respectivement des positions dans les segments S_{v_i} et S_{v_j}. Plus précisément, l'arc a relie un U du segment S_{v_i} à un A du segment S_{v_j}, tandis que l'arc a' relie un A du segment S_{v_i} à un U du segment S_{v_j}.

Intuitivement, si le sommet v_i est de degré deux alors tous les U et les A du segment S_{v_i} sont marqués par un arc de P. En revanche, si le sommet v_i est de degré un alors seulement un des deux U et un des deux A sont marqués par un arc de P. Afin de simplifier la preuve, lorsque le sommet v_i est de degré un, nous imposons que l'arc a marque la base U la plus à droite du segment S_{v_i} et, par conséquent, que l'arc a' marque la base A la plus à droite du segment S_{v_i}.

Par construction, on peut affirmer qu'étant donnée une arête (v_i, v_j) de $page_1$, les arcs a et a' correspondants dans P ne se croisent pas (ce qui est une condition imposée par le respect du niveau NESTED). Il nous reste à prouver qu'il est possible de reproduire la configuration sans croisement des arêtes de $page_1$. Pour ce faire, étant données deux arêtes (v_i, v_j) et (v_i, v_k) de $page_1$, il convient d'étudier les trois cas possibles suivants:

1. $i < j < k$ (le cas où $i < k < j$ est équivalent);
2. $k < j < i$ (le cas où $j < k < i$ est équivalent);

3. $j < i < k$ (le cas où $k < i < j$ est équivalent).

Dans le cas (1), le respect du niveaux NESTED est obtenu en reliant les arcs représentant l'arête (v_i, v_j) aux caractères U et A les plus à droite du segment S_{v_i}. Les arcs représentant l'arête (v_i, v_k) sont, par conséquent, reliés aux caractères U et A les plus à gauche du segment S_{v_i}.

Dans le cas (2), le respect du niveaux NESTED est obtenu en reliant les arcs représentant l'arête (v_i, v_j) aux caractères U et A les plus à gauche du segment S_{v_i}. Les arcs représentant l'arête (v_i, v_k) sont, par conséquent, reliés aux caractères U et A les plus à droite du segment S_{v_i}.

Dans le cas (3), le respect du niveaux NESTED est obtenu en reliant les arcs représentant l'arête (v_i, v_j) aux caractères U et A les plus à gauche du segment S_{v_i}. Les arcs représentant l'arête (v_i, v_k) sont, par conséquent, reliés aux caractères U et A les plus à droite du segment S_{v_i}.

Ce mécanisme nous permet d'affirmer que les arcs de P ne se croisent pas deux à deux et que par conséquent l'ensemble d'arcs P respecte le niveau NESTED. Le même mécanisme peut être appliqué aux arêtes de $page_2$ pour obtenir l'ensemble Q. La Figure 4.1 procure un exemple de ce type de construction. Nous appelons, par la suite, *UA-construction* toute construction de ce type.

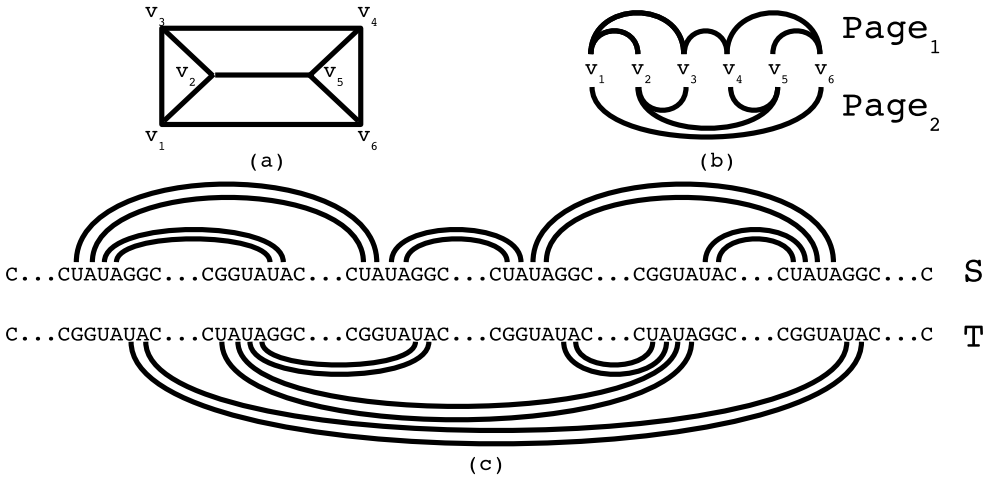


Figure 4.1 – Illustration d'une *UA-construction*. (a) Un graphe G cubique connexe planaire sans isthme à six sommets. (b) Un B-plongement du graphe G. (c) L'instance du problème L-EDIT(NESTED, NESTED) résultant de la *UA-construction*.

Dans [13], Bernhart *et al.* ont montré qu'un B-plongement d'un graphe cubique connexe planaire sans isthme peut être réalisé en temps polynomial. De plus, la taille des séquences construites est linéaire par rapport au nombre de sommets de G. Il en découle qu'une *UA-construction* peut être effectuée en temps polynomial.

De façon à compléter l'instance du problème L-EDIT(NESTED,NESTED), nous définissons formellement le paramètre $s' = 3n(w_b + \frac{4w_d}{3}) - p(6w_b + 2w_d - 6w_a)$ (la valeur $p > 0$ sera explicitée par la suite). Nous supposerons, par la suite, que toute instance du problème L-EDIT(NESTED,NESTED) que nous construisons vérifie les inégalités suivantes:

$$w_a > w_b > w_d > 0 \tag{4.1}$$

$$w_r > w_a + w_d \tag{4.2}$$

$$w_b + \frac{w_d}{3} > w_a \tag{4.3}$$

$$w_m > 2w_r \tag{4.4}$$

Nous débutons la preuve de la **NP**-complétude du problème L-EDIT(NESTED,NESTED) par quelques propriétés (Lemmes 4.1 à 4.5) sur les alignements optimaux des séquences S et T. Puis, ces résultats seront réutilisés dans le Lemme 4.6 pour conclure la preuve. Nous considérons dans tous ces lemmes que les conditions imposées par les inégalités (4.1) à (4.4) ci-avant sont vérifiées.

Par souci de lisibilité, nous renommons chacune des opérations d'édition suivantes (définies page 37) par le terme anglais correspondant et défini dans [75], à savoir:

- *arc-removing* désigne l'insertion ou la suppression d'un arc;
- *arc-mismatch* désigne la substitution d'un arc (i, j) par un arc (k, l) (si $S[i] = T[k]$ et $S[j] = T[l]$, on nomme cette opération *arc-match*);
- *arc-breaking* désigne le marquage de deux caractères libres ou la libération d'un arc;
- *arc-altering* désigne l'altération d'un arc.

Nous rappelons qu'une base est *libre* si elle n'est pas incidente à un arc; *marquée* par un arc dans le cas contraire. Par la suite, étant donné un alignement $\mathcal{A}$, $Score(\mathcal{A})$ représente la somme des coûts des opérations d'édition composant l'alignement.

Lemme 4.1. *Dans un alignement de S et T de score minimum, il n'y a pas de substitution de base.*

Preuve. Le principe de la preuve consiste à montrer que, sous les conditions imposées par les propriétés (4.1) à (4.4), partant d'un alignement $\mathcal{A}_1$ contenant une substitution de base, il est possible de construire un alignement $\mathcal{A}_2$ qui ne contient pas de substitution de base et tel que $Score(\mathcal{A}_2) < Score(\mathcal{A}_1)$. Rappelons que la substitution d'une base peut intervenir dans trois configurations différentes:

1. substitution entre deux bases libres: alors $\mathcal{A}_2$ est obtenu à partir de $\mathcal{A}_1$ en remplaçant la substitution de base par une insertion et une suppression de base.
   Dans ce cas, $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = 2w_d - w_m$

2. substitution entre une base libre et une base marquée par un arc a_1. Il y a deux sous-cas:
   - a_1 induit un *arc-breaking* dans $\mathcal{A}_1$. Alors $\mathcal{A}_2$ est obtenu à partir de $\mathcal{A}_1$ en remplaçant la substitution de base par une insertion ou une suppression de base, et l'*arc-breaking* par un *arc-altering* en alignant chaque base impliquée dans la substitution à un *gap*.
     Dans ce cas, $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = w_a + w_d - (w_b + w_m)$
   - a_1 induit un *arc-altering* dans $\mathcal{A}_1$. Alors $\mathcal{A}_2$ est obtenu à partir de $\mathcal{A}_1$ en remplacant la substitution de base par une insertion ou une suppression de base, et l'*arc-altering* par un *arc-removing* en alignant chaque base impliquée dans la substitution à un *gap*.
     Dans ce cas, $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = w_r + w_d - (w_a + w_m)$

3. substitution entre une base marquée par un arc a_1 et une base marquée par un arc a_2. Il y a trois sous-cas:
   - a_1 et a_2 induisent deux *arc-breaking* dans $\mathcal{A}_1$. Alors $\mathcal{A}_2$ est obtenu à partir de $\mathcal{A}_1$ en remplaçant les deux *arc-breaking* par deux *arc-altering* en alignant chaque base impliquée dans la substitution à un *gap*.
     Dans ce cas, $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = 2w_a - (2w_b + w_m)$
   - a_1 et a_2 induisent deux *arc-altering* dans $\mathcal{A}_1$. Alors $\mathcal{A}_2$ est obtenu à partir de $\mathcal{A}_1$ en remplaçant les deux *arc-altering* par deux *arc-removing* en alignant chaque base impliquée dans la substitution à un *gap*.
     Dans ce cas, $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = 2w_r - (2w_a + w_m)$

- a_1 induit un *arc-altering*, a_2 induit un *arc-breaking* dans $\mathcal{A}_1$. Alors $\mathcal{A}_2$ est obtenu à partir de $\mathcal{A}_1$ en remplaçant (i) l'*arc-breaking* par un *arc-altering* et (ii) l'*arc-altering* par un *arc-removing*, en alignant chaque base impliquée dans la substitution à un *gap*.
  Dans ce cas, $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = w_r + w_a - (w_b + w_m + w_a) = w_r - (w_b + w_m)$

Étant donné que $w_m > 2w_r$ et $w_r > w_a > w_b > w_d$ (voir inégalités (4.1), (4.2) et (4.4)), on en déduit que $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) < 0$ dans l'ensemble des cas. Donc, pour tout alignement $\mathcal{A}_1$ donné comprenant au moins une substitution de base, il est possible de trouver un alignement $\mathcal{A}_2$ sans substitution tel que $Score(\mathcal{A}_2) < Score(\mathcal{A}_1)$. Ce qui prouve le lemme. □

Définition 4.4 (Alignement canonique). Un *alignement canonique* de deux séquences S et T obtenu à partir d'une UA-construction est un alignement où le $i^{ème}$ segment S_c de S est aligné base à base au $i^{ème}$ segment T_c de T, pour tout $1 \leq i \leq n+1$.

Remarquons que, par construction, ni un *arc-match*, ni un *arc-mismatch* ne peut être présent dans un alignement canonique de S et T.

Lemme 4.2. *Tout alignement de S et T de score minimum est également canonique.*

Preuve. Soit $\mathcal{A}_1$ un alignement non canonique. Nous allons montrer que $\mathcal{A}_1$ n'est pas un alignement de score minimum. D'après le Lemme 4.1, nous pouvons supposer que $\mathcal{A}_1$ ne contient pas de substitution de base. De ce fait, $\mathcal{A}_1$ peut être non canonique dans deux cas de figure:

Cas 1.
Il existe un *alignement croisé* dans $\mathcal{A}_1$. Nous appelons alignement croisé, un alignement où au moins une base de S_{v_k} est alignée soit avec une base de T_{v_m}, soit avec un *gap* situé entre deux bases de T_{v_m} et tel que $k \neq m$, comme l'illustre la Figure 4.2.

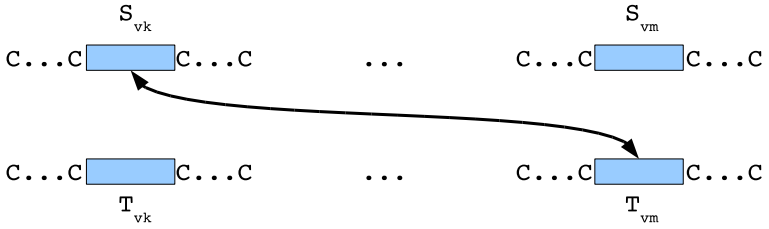


Figure 4.2 – Alignement croisé.

Soit $\mathcal{A}_2$ un alignement canonique sans substitution de base. Selon les conditions imposées par les inégalités (4.1) et (4.2), on a $w_r > w_a + w_d > 2w_d$. Par conséquent, le coût associé à toute opération sur une base libre peut être majoré par $\frac{w_r}{2}$

De plus, d'après les inégalités (4.1) et (4.2), on a $\frac{w_b}{2} < \frac{w_r}{2}$ et $\frac{w_a}{2} < \frac{w_r}{2}$. Donc, en répartissant équitablement le coût de l'opération sur un arc aux deux bases qu'il marque (par exemple, le coût w_b d'un *arc-breaking* peut être vu comme la composition d'un coût $\frac{w_b}{2}$ pour chacune des bases marquées), le coût associé à toute opération sur une base marquée par un arc peut être majoré par $\frac{w_r}{2}$ également.

Dans l'alignement $\mathcal{A}_2$, le score de l'alignement des segments S_c est nul. Par conséquent, $Score(\mathcal{A}_2)$ ne dépend que du score de l'alignement des segments S_{v_i} et T_{v_i} pour $1 \leq i \leq n$. Or, chacun de ces segments est composé de six bases. Étant donné que nous venons de prouver que le coût associé à toute opération sur une base libre ou marquée peut être majoré par $\frac{w_r}{2}$, on a $Score(\mathcal{A}_2) < 6nw_r$.

Sans perte de généralité, supposons qu'il existe un alignement croisé dans $\mathcal{A}_1$ tel que $k < m$. Dans un tel alignement, le croisement impose qu'au moins q ($q = |S_c|$) bases C de T situées à gauche de T_{v_m} soient insérées et qu'autant de bases C de S situées à droite de S_{v_k} soient supprimées. Nous obtenons donc $Score(\mathcal{A}_1) \geq 2qw_d$. Nous en déduisons que $Score(\mathcal{A}_2) < 6nw_r < 2qw_d \leq Score(\mathcal{A}_1)$ étant donné que $q > \frac{3nw_r}{w_d}$. L'alignement $\mathcal{A}_1$ n'est donc pas de score minimum dans le Cas 1.

Cas 2. Il n'existe pas d'alignement croisé dans $\mathcal{A}_1$. Dans ce cas, pour tout k, toute base de S_{v_k} est alignée soit à une base de T_{v_k}, soit à un *gap* situé entre deux bases de T_{v_k}. Soit ξ la somme des coûts d'alignement des segments S_{v_k} et T_{v_k} pour tout $1 \leq k \leq n$. Alors, nous obtenons $Score(\mathcal{A}_1) = \xi + \mathcal{S}'$ où $\mathcal{S}'$ est le score total de l'alignement base à base des segments S_c et T_c. L'hypothèse de départ (*i.e.* $\mathcal{A}_1$ est un alignement non canonique) impose qu'au moins une base C soit supprimée et une base C soit insérée dans $\mathcal{A}_1$. Par conséquent, nous obtenons $\mathcal{S}' \geq 2w_d$. Ainsi, $Score(\mathcal{A}_1) \geq \xi + 2w_d$.

Soit $\mathcal{A}_2$ un alignement canonique dans lequel, pour tout k, toute base de S_{v_k} est alignée identiquement à l'alignement $\mathcal{A}_1$, *i.e.* avec la même base de T_{v_k} ou un *gap*. Nous obtenons $Score(\mathcal{A}_2) = \xi < Score(\mathcal{A}_1)$, par conséquent $\mathcal{A}_1$ n'est pas un alignement de score minimum. □

Sachant que tout alignement de S et T de score minimum est canonique (*cf.* Lemme 4.2), le score d'un alignement de S et T de score minimum dépend uniquement des alignements locaux des segments S_{v_k} et T_{v_k} représentant les sommets de G. Étant donné qu'il n'y a pas de substitution de base dans un alignement canonique (*cf.* Lemme 4.1), alors par une analyse cas par cas nous pouvons dénombrer exactement dix-huit types d'alignements locaux comme l'illustre la Figure 4.3. On peut voir que tout autre alignement des segments S_{v_k} et T_{v_k}, pour $1 \leq k \leq n$, est équivalent, en terme de score, à un des dix-huit cas illustrés en Figure 4.3.

Définition 4.5 (Symétrique d'un type d'alignement). Le *symétrique d'un type d'alignement* t_i, noté t_{iSym}, est le type d'alignement obtenu à partir de t_i en inversant les deux segments (*i.e.* tel que le segment de S soit dorénavant sur T et vice-versa).

Notons que, d'après la définition des opérations sur les bases et les arcs, deux alignements locaux symétriques ont un score identique.

Lemme 4.3. *Un alignement $\mathcal{A}_2$ de S et T de score minimum contient uniquement des alignements locaux de types t_g, t_{ua}, t_{gSym} et t_{uaSym}.*

Preuve. Nous allons prouver que tout alignement de score minimum ne contient aucun alignement local de type $t \in \{t_1, t_{1Sym}, t_2, t_{2Sym}, \ldots t_{16}, t_{16Sym}\}$.

Soient $\mathcal{A}_1$ et $\mathcal{A}_2$ deux alignements canoniques ne différant que par l'alignement local de S_{v_i} et T_{v_i} pour un $1 \leq i \leq n$ donné. Plus précisément, nous considérerons que cet alignement local est de type t_{ua} ou t_g dans $\mathcal{A}_2$ et de tout autre type différent dans $\mathcal{A}_1$. La différence de score entre $\mathcal{A}_1$ et $\mathcal{A}_2$ ne peut donc provenir que de l'alignement local de S_{v_i} et T_{v_i}. Remarquons que cette différence est due localement à l'alignement d'un sous-ensemble de bases de S_{v_i} et T_{v_i}. Les alignements des bases de S_{v_i} et T_{v_i} communs à $\mathcal{A}_1$ et $\mathcal{A}_2$ ne seront par conséquent pas pris en compte lors du calcul de la différence de score entre $\mathcal{A}_1$ et $\mathcal{A}_2$. De plus, si un changement affecte une base marquée par un arc (disons un arc entre une base de S_{v_i} et une base de S_{v_j}) alors il est nécessaire de ne plus considérer uniquement la base affectée, mais de considérer les deux bases marquées par cet arc.

Le principe de la preuve qui suit est de démontrer qu'en considérant les conditions définies par les inégalités (4.1) à (4.4) et les alignements $\mathcal{A}_1$ et $\mathcal{A}_2$ définis précédemment, on a toujours: $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) < 0$, et donc que l'alignement $\mathcal{A}_1$ n'est pas un alignement de score minimum.

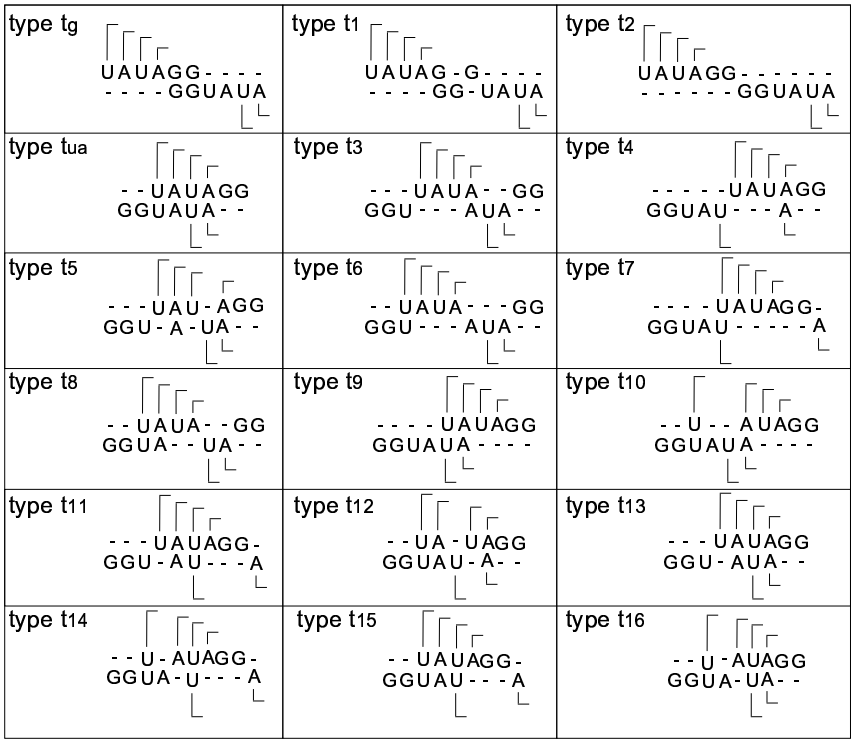


Figure 4.3 – Les dix-huit types d'alignements locaux des segments S_{v_k} et T_{v_k}.

Cas 1. Soit $\mathcal{A}_1$ un alignement canonique contenant un alignement local de S_{v_i} et T_{v_i} de type t_j pour $j \in \{3, \dots 16\}$. Soit $\mathcal{A}_2$ un alignement obtenu à partir de $\mathcal{A}_1$ en remplaçant l'alignement local de S_{v_i} et T_{v_i} par un alignement local de type t_{ua}. La Figure 4.4 illustre le Cas 1 pour $j = 3$.

Soit k_0 le nombre de bases de S_{v_i} et T_{v_i} supprimées ou insérées dans $\mathcal{A}_1$ et qui ne le sont plus dans $\mathcal{A}_2$. Soit k_1 le nombre de bases de S_{v_i} et T_{v_i} dans $\mathcal{A}_1$ induisant un *arc-removing* et n'induisant pas cet *arc-removing* mais un *arc-altering* dans $\mathcal{A}_2$. Soit k_2 le nombre de bases de S_{v_i} et T_{v_i} dans $\mathcal{A}_1$ induisant un *arc-altering* et n'induisant pas cet *arc-altering* mais un *arc-breaking* dans $\mathcal{A}_2$.

D'après la définition de k_0, k_1 et k_2, on a $k_0 + k_1 + k_2 > 0$, $k_0, k_1, k_2 \geq 0$ et $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = k_1(w_a - w_r) + k_2(w_b - w_a) + k_0(-w_d)$. D'après les conditions définies par les inégalités (4.1) et (4.2), $w_d > 0$ et $w_b < w_a < w_r$, on en déduit que $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) < 0$ dans le Cas 1.

Cas 2. Soit $\mathcal{A}_1$ un alignement canonique contenant un alignement local de S_{v_i} et T_{v_i} de type t_j pour $j \in \{1, 2\}$. Soit $\mathcal{A}_2$ un alignement obtenu à partir de $\mathcal{A}_1$ en remplaçant l'alignement local de S_{v_i} et T_{v_i} par un alignement local de type t_g.

Soit k_3 le nombre de bases de S_{v_i} et T_{v_i} supprimées ou insérées dans $\mathcal{A}_1$ et qui ne le sont plus dans $\mathcal{A}_2$. Nous obtenons $k_3 > 0$ et $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = k_3(-w_d)$. D'après les conditions définies par l'inégalité (4.1), $w_d > 0$. On en déduit que $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) < 0$ dans le Cas 2.

Par conséquent, tout alignement global canonique contenant un alignement local de S_{v_i} et T_{v_i} de type t_j pour $j \in \{1, 2\}$ (*resp.* $j \in \{3, \dots, 16\}$) a un score strictement supérieur au score d'un alignement identique où cet alignement local est de type t_g (*resp.* t_{ua}).

De plus, deux alignements locaux symétriques ayant un score identique, on peut en déduire que tout

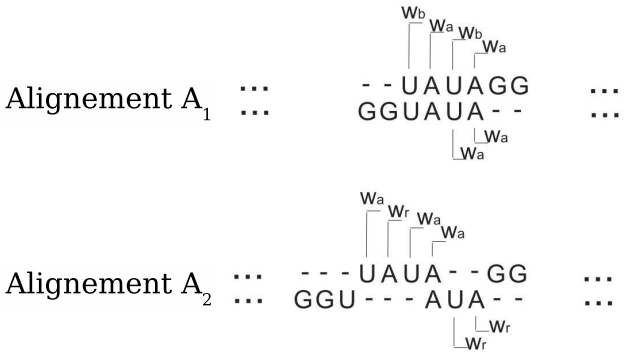


Figure 4.4 – Exemple d'un alignement canonique $\mathcal{A}_1$ contenant un alignement local de type t_3 et l'alignement $\mathcal{A}_2$ correspondant où l'alignement local de type t_{ua} a été substitué à celui de type t_3. Ici, $k_0 = 1, k_1 = 3, k_2 = 2$.

alignement global canonique contenant un alignement local de S_{v_i} et T_{v_i} de type t_{jSym} pour $j \in \{1, 2\}$ (*resp.* $j \in \{3, \ldots, 16\}$) a un score strictement supérieur au score d'un alignement identique où cet alignement local est de type t_{gSym} (*resp.* t_{uaSym}).

Par conséquent, on peut affirmer que tout alignement de S et T de score minimum est canonique (d'après le Lemme 4.2) et contient seulement des alignements locaux de types t_g, t_{ua}, t_{gSym} et t_{uaSym}. □

Lemme 4.4. *Soient $\mathcal{A}_1$ un alignement de S et T et S_{v_i}, S_{v_j}, T_{v_i} et T_{v_j} quatre segments tels que $1 \leq i, j \leq n$ et il existe un arc entre une base de S_{v_i} et une base de S_{v_j} ou un arc entre une base de T_{v_i} et une base de T_{v_j}.*

Si l'alignement $\mathcal{A}_1$ est de score minimum, $\mathcal{A}_1$ ne peut pas contenir simultanément:

- *un alignement local de S_{v_i} et T_{v_i} de type t_k et*
- *un alignement local de S_{v_j} et T_{v_j} de type $t_{k'}$ pour $k, k' \in \{g, gSym\}$.*

Preuve. Soit $\mathcal{A}_1$ un alignement de S et T de score minimum contenant un alignement local de S_{v_i} et T_{v_i} de type t_k pour $k \in \{g, gSym\}$ et un alignement local de S_{v_j} et T_{v_j} de type $t_{k'}$ pour $k' \in \{g, gSym\}$.

Soit $\mathcal{A}_2$ un alignement obtenu à partir de $\mathcal{A}_1$ en remplaçant l'alignement local de S_{v_i} et T_{v_i} par un alignement local de type $t_{k''}$ pour $k'' \in \{ua, uaSym\}$.

Nous allons montrer que $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) < 0$, et que par conséquent $\mathcal{A}_1$ n'est pas un alignement de coût minimum. Soit k_1 le nombre de bases de S_{v_i} dans $\mathcal{A}_1$ qui induisent un *arc-removing* et qui dans $\mathcal{A}_2$ n'induisent plus cet *arc-removing* mais un *arc-altering*. Soit k_2 le nombre de bases de S_{v_i} dans $\mathcal{A}_1$ qui induisent un *arc-altering* et qui dans $\mathcal{A}_2$ n'induisent plus cet *arc-altering* mais un *arc-breaking*.

Nous obtenons $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) = 4w_d - 2w_d + k_1(w_b - w_a) + k_2(w_a - w_r) = 2w_d + 2w_a - 2w_r + k_1(w_b - w_a) + (k_2 - 2)(w_a - w_r)$ où $k_1 + k_2 = 6$, $k_1 \geq 0$ et $k_2 \geq 2$.

D'après les conditions définies par les inégalités (4.1) et (4.2), $w_d > 0$ et $w_r > w_a > w_b$. On en déduit que $Score(\mathcal{A}_2) - Score(\mathcal{A}_1) < 0$. Par conséquent, l'alignement $\mathcal{A}_1$ n'est pas un alignement de score minimum. □

Définition 4.6 (Alignement t_g-stable). Soit $\mathcal{A}$ un alignement canonique de S et T contenant uniquement des alignements locaux de types t_g, t_{ua}, t_{gSym} et t_{uaSym}. Soient S_{v_i}, S_{v_j}, T_{v_i} et T_{v_j} quatre segments tels

que $1 \leq i, j \leq n$ et il existe un arc entre une base de S_{v_i} et une base de S_{v_j} ou un arc entre une base de T_{v_i} et une base de T_{v_j}.

L'alignement $\mathcal{A}$ est t_g-stable s'il ne contient pas simultanément:

- un alignement local de S_{v_i} et T_{v_i} de type t_k pour $k \in \{g, gSym\}$ et
- un alignement local de S_{v_j} et T_{v_j} de type $t_{k'}$ pour $k' \in \{g, gSym\}$.

Pour simplifier la preuve du Lemme 4.5, on appelle *lien* t_k-$t_{k'}$ tout arc reliant une base appartenant à un alignement local de type t_k à une base appartenant à un alignement local de type $t_{k'}$.

Lemme 4.5. *Le score d'un alignement canonique t_g-stable $\mathcal{A}$ est:*

$$Score(\mathcal{A}) = 3n(w_b + \tfrac{4w_d}{3}) - p(6w_b + 2w_d - 6w_a)$$

où p est le nombre d'alignements locaux de type t_k où $k \in \{g, gSym\}$ dans $\mathcal{A}$.

Preuve. G étant un graphe cubique à n sommets, G a $\frac{3n}{2}$ arêtes. Par construction, pour chaque arête de G, deux arcs sont construits dans $P \cup Q$. Donc, $|P \cup Q| = 3n$. Si p est le nombre d'alignements locaux de type t_k où $k \in \{g, gSym\}$ dans $\mathcal{A}$, alors il existe $6p$ liens t_k-$t_{k'}$ où $k \in \{g, gSym\}$ et $k' \in \{ua, uaSym\}$. Par conséquent il exite $3n - 6p$ liens t_k-$t_{k'}$ avec $k, k' \in \{ua, uaSym\}$.

Afin de calculer $Score(\mathcal{A})$, on répartit équitablement, pour chaque alignement local, le coût induit par les bases libres sur les bases marquées. Par exemple, le coût d'une base libre dans un alignement local de type t_{ua} est nul. En revanche, le coût d'une base marquée dans un alignement local de type t_{ua} est augmenté de $\frac{2w_d}{3}$ (*i.e.* le coût des quatre bases libres de l'alignement est réparti équitablement sur les six bases marquées: $\frac{4w_d}{6}$).

On peut alors évaluer le coût d'un lien t_k-$t_{k'}$ où $k, k' \in \{ua, uaSym\}$ comme suit: le coût de chaque base marquée est augmenté de $\frac{2w_d}{3}$. Par conséquent, le coût d'un lien t_k-$t_{k'}$ où $k, k' \in \{ua, uaSym\}$, est égal à $w_b + \frac{2w_d}{3} + \frac{2w_d}{3} = w_b + \frac{4w_d}{3}$.

De façon similaire, on peut évaluer le coût d'un lien t_k-$t_{k'}$ où $k \in \{g, gSym\}$ et $k' \in \{ua, uaSym\}$ comme suit: chaque base marquée de l'alignement local de type $t_{k'}$ est augmentée de $\frac{2w_d}{3}$. Le coût d'une base marquée dans un alignement local de type t_k est augmenté de $\frac{w_d}{3}$ (*i.e.* le coût des deux bases libres de l'alignement est réparti équitablement sur les six bases marquées: $\frac{2w_d}{6}$). Par conséquent, le coût d'un lien t_k-$t_{k'}$ où $k \in \{g, gSym\}$ et $k' \in \{ua, uaSym\}$ est égal à $w_a + \frac{w_d}{3} + \frac{2w_d}{3} = w_a + w_d$.

Nous obtenons:

$$\begin{aligned} Score(\mathcal{A}) &= (3n - 6p)(w_b + \tfrac{4w_d}{3}) + 6p(w_d + w_a) \\ &= 3n(w_b + \tfrac{4w_d}{3}) - p(6w_b + \tfrac{24w_d}{3} - 6w_d - 6w_a) \\ &= 3n(w_b + \tfrac{4w_d}{3}) - p(6w_b + 2w_d - 6w_a) \end{aligned}$$

□

Les Lemmes 4.1 à 4.5 fournissent l'ensemble des résultats intermédiaires dont nous avons besoin pour prouver la **NP**-complétude du problème L-EDIT(NESTED, NESTED).

Lemme 4.6. *Un graphe cubique connexe planaire sans isthme G admet un ensemble indépendant de sommets V' tel que $|V'| \geq k$ si et seulement si la distance d'édition entre les séquences (S, P) et (T, Q) obtenues à partir de G par une UA-construction est inférieure ou égale à s', où $s' = 3n(w_b + \frac{4w_d}{3}) - k(6w_b + 2w_d - 6w_a)$.*

Preuve. ($\Rightarrow$)
Soient $G = (V, E)$ un graphe cubique connexe planaire sans isthme et (S, P) et (T, Q) les séquences obtenues à partir de G par une UA-construction.

Soit $V' \subseteq V$ un ensemble indépendant de G tel que $|V'| \geq k$. Soit $\mathcal{A}$ un alignement canonique de (S, P) et (T, Q) tel que:

- pour tout sommet $v \in V'$, l'alignement local de S_v et T_v est de type t_k où $k \in \{g, gSym\}$;
- pour tout sommet $u \in V - V'$, l'alignement local de S_u et T_u est de type $t_{k'}$ où $k' \in \{ua, uaSym\}$.

Par conséquent, par définition, l'alignement est $t_{\mathcal{G}}$*-stable*. Par le Lemme 4.5:

$$\begin{aligned} Score(\mathcal{A}) = &\quad 3n(w_b + \tfrac{4w_d}{3}) - |V'|(6w_b + 2w_d - 6w_a) \\ \leq &\quad 3n(w_b + \tfrac{4w_d}{3}) - k(6w_b + 2w_d - 6w_a) = s' \end{aligned}$$

puisque $|V'| \geq k$ par hypothèse.

($\Leftarrow$)
Soient $G = (V, E)$ un graphe cubique connexe planaire sans isthme et (S, P) et (T, Q) les séquences obtenues à partir de G par une UA-construction. Soit $\mathcal{A}$ un alignement de (S, P) et (T, Q) de score minimum. D'après les Lemmes 4.3 et 4.4, $\mathcal{A}$ est un alignement $t_{\mathcal{G}}$*-stable*.

Soit V' un ensemble de sommets de G pour lesquels les alignements locaux des segments correspondants dans $\mathcal{A}$ sont de type t_k où $k \in \{g, gSym\}$. Étant donné que $\mathcal{A}$ est un alignement $t_{\mathcal{G}}$*-stable*, V' est un ensemble indépendant de G. De plus, par le Lemme 4.5 : $Score(\mathcal{A}) = 3n(w_b + \frac{4w_d}{3}) - |V'|(6w_b + 2w_d - 6w_a)$.

Or, $Score(\mathcal{A}) \leq 3n(w_b + \frac{4w_d}{3}) - k(6w_b + 2w_d - 6w_a)$ et d'après l'inégalité (4.3) $w_b + \frac{w_d}{3} > w_a$. Nous obtenons $k \leq |V'|$. Le Lemme 4.6 est prouvé. □

Nous avons démontré en début de section que le problème L-EDIT(NESTED, NESTED) appartient à la classe **NP**. D'autre part, nous avons proposé une transformation polynomiale à partir du problème **NP**-complet MIS-3P. Par conséquent, d'après le Lemme 4.6 le problème L-EDIT(NESTED, NESTED) est **NP**-complet. Le Théorème 4.1 est prouvé.

4.3 Conclusion et perspectives

Dans ce chapitre, nous avons prouvé que le problème L-EDIT(NESTED,NESTED) défini dans [75] est **NP**-complet. Cela a été fait en utilisant une réduction à partir du problème MIS-3P, via une technique de plongement en deux pages d'un graphe. Le Théorème 4.1 répond à une question ouverte posée dans [75], et complète l'étude de la complexité du problème L-EDIT(T_1,T_2) proposé par Lin *et al.* [75], où $T_1, T_2 \in \{$PLAIN,CHAIN,NESTED,CROSSING,UNLIMITED$\}$ (*cf.* Table 4.1).

La distance d'édition proposée par Lin *et al.* considère séparément les bases et les liens hydrogènes (les arcs), ce qui semble pertinent vis-à-vis des structures secondaires d'ARN. D'autre part, Lin *et al.* ont proposé de nouvelles opérations d'édition non-triviales agissant simultanément sur un arc et une base incidente à ce dernier. Malheureusement, l'utilisation de ces nouvelles opérations d'édition pour la comparaison de structures secondaires rend le problème L-EDIT(NESTED,NESTED) **NP**-complet. C'est pourquoi cette approche n'est pas utilisable en pratique pour la comparaison des ARN, bien qu'elle soit plus fine dans la gestion des arcs.

Bien que le problème L-EDIT(NESTED,NESTED) soit **NP**-complet, il existe un algorithme d'approximation avec un ratio d'approximation $\alpha = \max\{\frac{2w_a}{w_b+w_r}, \frac{w_b+w_r}{2w_a}\}$ proposé dans [75]. Malgré tout, le ratio d'approximation α dépend des valeurs des paramètres w_a, w_b and w_r. On est en droit de se demander si un algorithme d'approximation ayant un ratio constant existe.

Malgré la difficulté, dans sa généralité, du problème de calcul de la distance d'édition entre deux structures secondaires de molécules d'ARN, il existe des algorithmes exacts performants pour résoudre ce type de problème sur des classes d'instances plus spécifiques. Parmi toutes les approches proposées pour résoudre ce problème, celle de Denise *et al.* [62] a la particularité de prendre en compte en partie les opérations introduites par Lin *et al.* dans le calcul d'une distance d'édition entre deux structures secondaires de molécules d'ARN.

Dans [62], Denise *et al.* proposent de modéliser les structures secondaires par des arbres enracinés et de traduire les différentes opérations d'édition de Lin *et al.* dans ce nouveau contexte. La distance entre deux structures secondaires correspond alors à une distance entre deux structures arborescentes [70, 98]. La Figure 4.5 présente le calcul de la distance d'édition entre deux arbres représentant des structures secondaires de molécules d'ARN.

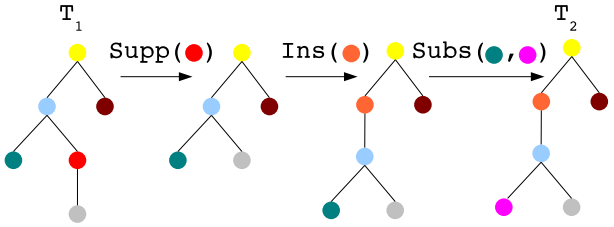


Figure 4.5 – Calcul de la distance d'édition entre deux arbres représentant des structures secondaires de molécules d'ARN. Dans cette exemple, le passage de T_1 à T_2 nécessite une suppression, une insertion et une substitution.

Dans [62], Denise *et al.* proposent de définir de nouvelles opérations d'édition d'arbres correspondant aux opérations d'*arc-breaking* et d'*arc-altering*. En effet, les autres opérations sont d'ores et déjà prises en compte dans le modèle de la distance d'édition entre deux arbres: l'insertion ou la suppression (*resp.* la substitution) d'un noeud interne correspond à un *arc-removing* (*resp.* un *arc-mismatch*). Les nouvelles opérations sont illustrées en Figure 4.6. Les auteurs ont montré que le problème est polynomial, et ont pour cela proposé un algorithme de programmation dynamique qui a une complexité en temps en $O(n^2m^2(m+n))$. La polynomialité de cet algorithme est rendue possible par le fait que l'on ne considère pas certains cas de figure pouvant intervenir dans le calcul de la distance d'édition à la Lin *et al.*

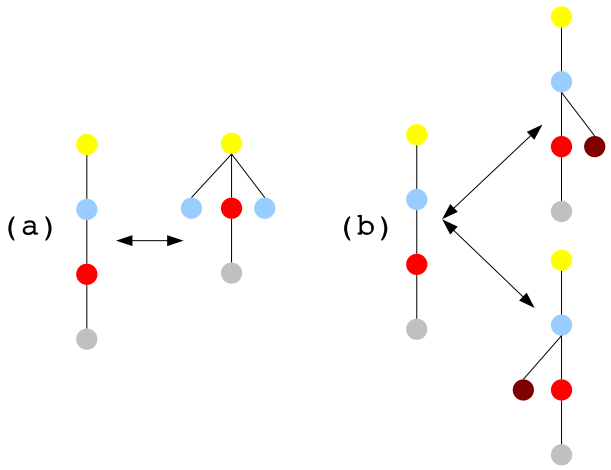


Figure 4.6 – Deux nouvelles opérations d'édition entre deux arbres représentant des structures secondaires de molécules d'ARN. (a) Une opération d'appariement/désappariement correspondant à un *arc-breaking*. (b) Une opération correspondant à un *arc-altering*.

CHAPITRE 5

Le problème ARC-PRESERVING SUBSEQUENCE

5.1 Introduction

Dans ce chapitre, nous développons des résultats publiés dans [21, 22] et obtenus en collaboration avec Guillaume Fertin, Romeo Rizzi et Stéphane Vialette. Rappelons que le problème APS, acronyme de ARC-PRESERVING SUBSEQUENCE, consiste à trouver une occurrence d'une sous-structure (appelée le motif) dans une séquence arc-annotée. Le problème de décision APS se définit comme suit.

ARC-PRESERVING SUBSEQUENCE (APS)
DONNÉES: *Deux séquences arc-annotées* (S, P) *et* (T, Q) *telles que* $|T| \leq |S|$.
QUESTION: (T, Q) *peut-elle être obtenue à partir de* (S, P) *en supprimant certaines de ses bases ainsi que les éventuels arcs incidents à ces dernières?*

Nous rappelons que ce problème est équivalent à savoir si (T, Q) est une sous-séquence commune arc-préservante de (S, P) et (T, Q). Comme nous l'avons précisé dans la Section 3.1.2.3, l'existence d'un algorithme polynomial pour le sous-problème APS(CROSSING, PLAIN) est le dernier cas ouvert pour le problème APS [57]. Dans ce chapitre, nous démontrons que le sous-problème APS(CROSSING, PLAIN) est un problème **NP**-complet – complétant ainsi l'étude du problème APS (*cf.* Table 5.1).

APS					
	UNLIMITED	CROSSING	NESTED	CHAIN	PLAIN
UNLIMITED	**NP**-complet [48]				
CROSSING		**NP**-complet [48]	**NP**-complet [58]		**NP-complet [21, 22]**
NESTED			$\mathbf{O}(nm)$ [57]		
CHAIN				$\mathbf{O}(nm)$ [57]	$\mathbf{O}(n+m)$ [57]

Table 5.1 – Complexité du problème APS.

Après avoir prouvé que le sous-problème APS(CROSSING, PLAIN) est **NP**-complet, nous proposerons un raffinement du problème APS afin de mieux définir ce qui rend le problème difficile. En effet, lors de l'analyse de la complexité d'un problème, il est intéressant de déterminer précisément la frontière

entre les cas polynomiaux et les cas **NP**-complets. Par conséquent, afin de mieux évaluer la complexité du problème APS, il est intéressant de subdiviser encore les sous-problèmes plus spécifiquement, en raffinant les niveaux de complexité classiques.

Pour ce faire, nous utiliserons dans la Section 5.3 les notions introduites par Vialette [93] dans le contexte des 2-intervalles (*cf.* Section 3.2). La conséquence directe est que le nombre de niveaux de complexité passe de cinq à huit, et la table de complexité correspondante doit être complétée. Les résultats existants, complétés de deux preuves de **NP**-complétude ainsi que deux algorithmes polynomiaux présentés en Section 5.3, nous permettront de compléter intégralement la nouvelle table de complexité, déterminant ainsi ce qui rend réellement le problème APS difficile.

5.2 NP-complétude du sous-problème APS(CROSSING, PLAIN)

Nous démontrons dans cette section que le sous-problème APS(CROSSING, PLAIN) est **NP**-complet. Ce résultat répond au dernier cas ouvert posé dans [57], et complète la table de complexité du problème APS en considérant les niveaux classiques de complexité, *i.e.* PLAIN, CHAIN, NESTED et CROSSING (*cf.* Table 5.1).

Pour ce faire, nous proposons une transformation polynomiale du problème **NP**-complet EXACT 3-CNF-SAT [53] vers le sous-problème APS(CROSSING, PLAIN).

> EXACT 3-CNF-SAT
> DONNÉES*: Un ensemble $\mathcal{V}_n$ de n variables et un ensemble $\mathcal{C}_q$ de q clauses (chacune composée de trois littéraux) sur $\mathcal{V}_n$.*
> QUESTION*: Existe-il un ensemble de valeurs pour $\mathcal{V}_n$ satisfaisant toutes les clauses de $\mathcal{C}_q$?*

Il est facile de vérifier que le sous-problème APS(CROSSING, PLAIN) est dans la classe **NP**. Le reste de cette section est consacré à prouver que ce sous-problème est également **NP**-dur. Soient $\mathcal{V}_n = \{x_1, x_2, ...x_n\}$ un ensemble fini de n variables et $\mathcal{C}_q = \{c_1, c_2, \ldots, c_q\}$ un ensemble de q clauses. Sans perte de généralité, nous supposons que les littéraux de chaque clause sont ordonnés de gauche à droite, *i.e.* si $c_i = (x_j \vee x_k \vee x_l)$ alors $j < k < l$. Nous commençons par définir formellement la construction des séquences S et T:

$$S = S^s_{x_1} A\, S^s_{\overline{x_1}}\, S^s_{x_2} A\, S^s_{\overline{x_2}} \ldots S^s_{x_n} A\, S^s_{\overline{x_n}}\, S_{c_1}\, S_{c_2} \ldots S_{c_q}\, S^e_{x_1}\, S^e_{x_2} \ldots S^e_{x_n}$$
$$T = T^s_{x_1}\, T^s_{x_2} \ldots T^s_{x_n}\, T_{c_1}\, T_{c_2} \ldots T_{c_q}\, T^e_{x_1}\, T^e_{x_2} \ldots T^e_{x_n}$$

Nous détaillons maintenant les sous-séquences composant S et T. Soient γ_m (*resp.* $\gamma_{\overline{m}}$) le nombre d'occurrences du littéral x_m (*resp.* $\overline{x_m}$) dans $\mathcal{C}_q$ et $k_m = \max(\gamma_m, \gamma_{\overline{m}})$. Pour chaque variable $x_m \in \mathcal{V}_n$, avec $1 \leq m \leq n$, nous construisons les mots:

- $S^s_{x_m} = AC^{k_m}$;
- $S^s_{\overline{x_m}} = C^{k_m}A$;
- $T^s_{x_m} = AC^{k_m}A$.

où C^{k_m} est un mot de k_m bases C consécutives. De plus, pour chaque clause c_i de $\mathcal{C}_q$, avec $1 \leq i \leq q$, nous construisons les mots $S_{c_i} = UGGGA$ et $T_{c_i} = UGA$. Finalement, pour chaque variable $x_m \in \mathcal{V}_n$, avec $1 \leq m \leq n$, nous construisons les mots $S^e_{x_m} = UUA$ et $T^e_{x_m} = UA$.

Une fois les deux séquences définies, nous définissons les structures d'arcs correspondantes (*cf.* Figure 5.1). Par la suite, on note $S[i]$ la $i^{ème}$ base de la séquence S et, pour tout $1 \leq m \leq n$, on

note $l_{\overline{m}} = |S^s_{\overline{x_m}}|$. Pour tout $1 \leq m \leq n$, nous construisons les deux arcs suivant: ($S^s_{x_m}[1]$,$S^e_{x_m}[1]$) et ($S^s_{\overline{x_m}}[l_{\overline{m}}]$,$S^e_{x_m}[2]$). Pour chaque clause c_i de $\mathcal{C}_q$, avec $1 \leq i \leq q$, et pour tout $1 \leq m \leq n$, si le $k^{ème}$ ($k \in \{1,2,3\}$) littéral de c_i est x_m (*resp.* $\overline{x_m}$) alors nous construisons un arc entre une base libre C quelconque de $S^s_{\overline{x_m}}$ (*resp.* $S^s_{x_m}$) et la $k^{ème}$ base G de S_{c_i} (notons que ceci est rendu possible du fait de la définition de $S^s_{\overline{x_m}}$, $S^s_{x_m}$ et S_{c_i}).

En tout, l'instance ainsi construite est composée de $3q+2n$ arcs. Nous appelons APS-CP-construction, toute construction de ce type. Par la suite, nous distinguerons les arcs entre une base A et une base U, que l'on nommera AU-arcs, des arcs entre une base C et une base G, que l'on nommera CG-arcs. Une illustration d'une APS-CP-construction est donnée en Figure 5.1. Clairement, notre construction peut être effectuée en temps polynomial. De plus, le résultat d'une telle construction est bien une instance du sous-problème APS(CROSSING, PLAIN), puisque $Q = \emptyset$ (aucun arc n'est ajouté à (T, Q)) et P est un ensemble d'arcs respectant le niveau CROSSING (puisqu'il n'existe pas deux arcs a_1, a_2 de P tels que a_1 précède a_2 ou a_2 précède a_1).

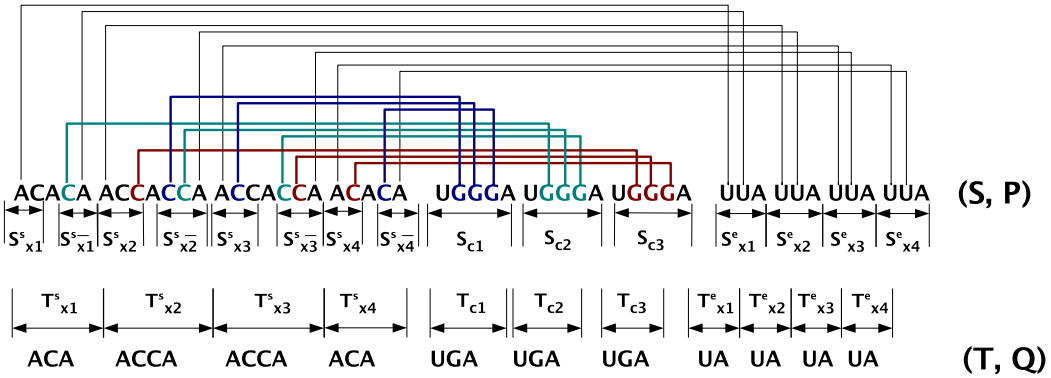


Figure 5.1 – Exemple d'une APS-CP-construction avec $\mathcal{C}_q = (x_2 \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4})$.

Nous commençons par prouver un lemme sur la canonicité d'une APS-CP-construction que nous utiliserons pour prouver la **NP**-complétude du sous-problème APS(CROSSING, PLAIN).

Lemme 5.1. *Soient (S, P) et (T, Q) deux séquences arc-annotées obtenues à partir d'une* APS-CP-*construction. Si (T, Q) peut être obtenue à partir de (S, P) en supprimant certaines des bases de S ainsi que les éventuels arcs incidents à ces bases, alors pour tout $1 \leq i \leq q$ et $1 \leq m \leq n$:*

1. *la séquence T_{c_i} est obtenue à partir de S_{c_i} en supprimant deux de ses trois bases G;*
2. *la séquence $T^e_{x_m}$ est obtenue à partir de $S^e_{x_m}$ en supprimant une de ses deux bases U;*
3. *la séquence $T^s_{x_m}$ est obtenue à partir de $S^s_{x_m} A S^s_{\overline{x_m}}$ en supprimant soit toutes les bases de $S^s_{x_m}$, soit toutes celles de $S^s_{\overline{x_m}}$.*

Preuve. Soient (S, P) et (T, Q) deux séquences arc-annotées obtenues à partir d'une APS-CP-construction.

(1) Par construction, la première base U apparaissant dans S (*resp.* T) est $S_{c_1}[1]$ (*resp.* $T_{c_1}[1]$). Par conséquent, la base $T_{c_1}[1]$ ne peut être obtenue qu'à partir de la base $S_{c_1}[1]$ ou d'une base U de S située après $S_{c_1}[1]$. De plus, le nombre de bases A apparaissant après la base $S_{c_1}[1]$ dans S est égal au nombre de bases A apparaissant après la base $T_{c_1}[1]$ dans T. Par conséquent, la $i^{ème}$ base A apparaissant après $S_{c_1}[1]$ dans S doit être appariée à la $i^{ème}$ base A apparaissant après $T_{c_1}[1]$: $\forall 1 \leq i \leq q$, $T_{c_i}[3]$ est appariée à $S_{c_i}[5]$.

De ce fait, la base $T_{c_q}[3]$ est appariée à la base $S_{c_q}[5]$. Étant donné qu'il y a autant de bases U entre les bases $S_{c_1}[1]$ et $S_{c_q}[5]$ qu'il y en a entre les bases $T_{c_1}[1]$ et $T_{c_q}[3]$, pour tout $1 \leq i \leq q$, la base $T_{c_i}[1]$ doit être appariée à la base $S_{c_i}[1]$. On en conclut donc que pour tout $1 \leq i \leq q$, la séquence T_{c_i} est obtenue en supprimant deux des trois bases G de S_{c_i}.

(2) En reprenant l'argument, développé dans (1), concernant les bases A apparaissant après les bases $S_{c_1}[1]$ et $T_{c_1}[1]$, on en conclut que si (T, Q) peut être obtenue à partir de (S, P), alors pour tout $1 \leq m \leq n$ la base $T^e_{x_m}[2]$ est appariée à la base $S^e_{x_m}[3]$. Par conséquent, pour tout $1 \leq m \leq n$, la séquence $T^e_{x_m}$ est obtenue à partir de $S^e_{x_m}$, et plus précisément $T^e_{x_m}[1]$ est appariée à $S^e_{x_m}[1]$ ou à $S^e_{x_m}[2]$.

(3) Par définition, étant donné que $Q = \emptyset$, pour tout arc de P, au moins une base marquée doit être supprimée. Nous venons de préciser que pour tout $1 \leq m \leq n$, $T^e_{x_m}[1]$ est appariée à $S^e_{x_m}[1]$ ou à $S^e_{x_m}[2]$. Donc, étant donné que, par construction, il y a un arc entre $S^e_{x_m}[1]$ et $S^s_{x_m}[1]$ (*resp.* $S^e_{x_m}[2]$ et $S^s_{\overline{x_m}}[l_{\overline{m}}]$), pour tout $1 \leq m \leq n$, soit $S^s_{x_m}[1]$, soit $S^s_{\overline{x_m}}[l_{\overline{m}}]$ doit être supprimée. Notons que tous ces arcs connectent une base A apparaissant avant la base $S_{c_1}[1]$ à une base U apparaissant après la base $S_{c_q}[5]$.

Par conséquent, pour tout $1 \leq m \leq n$ une base A apparaissant avant la base $S_{c_1}[1]$ dans S est supprimée. Au départ, il y avait $3n$ bases A apparaissant avant la base $S_{c_1}[1]$ dans S et $2n$ apparaissant avant la base $T_{c_1}[1]$ dans T. Donc, le nombre de bases A non supprimées dans S et apparaissant avant la base $S_{c_1}[1]$ est égal au nombre de bases A apparaissant avant la base $T_{c_1}[1]$ dans T. Étant donné que, pour tout $1 \leq m \leq n$, une base A de $S^s_{x_m}$ ou de $S^s_{\overline{x_m}}$ est supprimée, on en conclut que pour tout $1 \leq m \leq n$, la séquence $T^s_{x_m}$ est obtenue à partir de $S^s_{x_m} A S^s_{\overline{x_m}}$, soit en supprimant $S^s_{x_m}$, soit en supprimant $S^s_{\overline{x_m}}$. □

Il nous reste à prouver que notre construction est une transformation polynomiale du problème EXACT 3-CNF-SAT vers le sous-problème APS(CROSSING, PLAIN).

Lemme 5.2. *Soient I une instance du problème* EXACT 3-CNF-SAT *avec n variables et q clauses, et $I' = ((S, P); (T, Q))$ une instance du problème* APS(CROSSING,PLAIN) *obtenue à la suite d'une* APS-CP*-construction à partir de I. Il existe un ensemble de valeurs pour les n variables satisfaisant les q clauses de I si et seulement si (T, Q) est une sous-séquence commune arc-préservante de (S, P) et (T, Q).*

Preuve. ($\Rightarrow$) Supposons que nous avons un ensemble $\mathcal{V}$ de valeurs pour les n variables satisfaisant les q clauses de I. Par définition, pour chaque clause il existe au moins un littéral la satisfaisant. Par la suite, j_i représentera, pour tout $1 \leq i \leq q$, le plus petit indice des littéraux de c_i (*i.e.* 1, 2 ou 3) qui, par leurs valeurs, satisfont c_i. Soient (S, P) et (T, Q) deux séquences arc-annotées obtenues à la suite d'une APS-CP-construction à partir de I. Nous recherchons un ensemble $\mathcal{B}$ de bases à supprimer de S afin d'obtenir T. Pour chaque variable $x_m \in \mathcal{V}$ avec $1 \leq m \leq n$, nous définissons $\mathcal{B}$ comme suit:

- si $x_m = Vrai$ alors $\mathcal{B}$ contient chaque base de la séquence $S^s_{\overline{x_m}}$ et la base $S^e_{x_m}[1]$;
- si $x_m = Faux$ alors $\mathcal{B}$ contient chaque base de la séquence $S^s_{x_m}$ et la base $S^e_{x_m}[2]$;
- si $j_i = 1$ alors $\mathcal{B}$ contient les bases $S_{c_i}[3]$ et $S_{c_i}[4]$;
- si $j_i = 2$ alors $\mathcal{B}$ contient les bases $S_{c_i}[2]$ et $S_{c_i}[4]$;
- si $j_i = 3$ alors $\mathcal{B}$ contient les bases $S_{c_i}[2]$ et $S_{c_i}[3]$.

Étant donné qu'une variable ne peut avoir qu'une unique valeur ($Vrai$ ou $Faux$), pour tout $1 \leq m \leq n$ soit chaque base de $S^s_{\overline{x_m}}$ et la base $S^e_{x_m}[1]$, soit chaque base de $S^s_{x_m}$ et la base $S^e_{x_m}[2]$ appartient à $\mathcal{B}$. Par conséquent, $\mathcal{B}$ contient au moins une base de chaque AU-arc de P.

Pour tout $1 \leq i \leq q$, deux des trois bases G de S_{c_i} sont dans $\mathcal{B}$. Donc, $\mathcal{B}$ contient au moins une base de deux tiers des CG-arcs de P. De plus, $S_{c_i}[j_i + 1]$ est la base G n'appartenant pas à $\mathcal{B}$. Nous

supposons, par la suite, que le $j_i^{ème}$ littéral de la clause c_i est x_m, avec $1 \leq m \leq n$. Par construction, il existe un arc entre une base C de $S^s_{\overline{x_m}}$ et la base $S_{c_i}[j_i + 1]$ dans P.

Par définition, si $\mathcal{V}$ est un ensemble de valeurs pour les n variables satisfaisant les q clauses de I, $\mathcal{V}$ satisfait c_i et donc $x_m = Vrai$. Nous avons précisé, dans la définition de $\mathcal{B}$ que si $x_m = Vrai$ alors chaque base de $S^s_{\overline{x_m}}$ est dans $\mathcal{B}$. Donc, la base C de $S^s_{\overline{x_m}}$ marquée avec $S_{c_i}[j_i + 1]$ par un CG-arc de P appartient à $\mathcal{B}$. Un résultat similaire peut être obtenu si le $j_i^{ème}$ littéral de la clause c_i est $\overline{x_m}$. Par conséquent, $\mathcal{B}$ contient au moins une base de chaque CG-arc de P.

Si S' est la séquence obtenue à partir de S en supprimant toutes les bases de $\mathcal{B}$ ainsi que les éventuels arcs incidents à ces bases alors il n'y a pas d'arc dans S' (*i.e.* pas de AU-arcs, ni de CG-arcs). D'après la définition de $\mathcal{B}$, la séquence S' est donc obtenue à partir de S en supprimant, pour tout $1 \leq i \leq q$ et pour tout $1 \leq m \leq n$, toutes les bases de $S^s_{x_m}$ ou de $S^s_{\overline{x_m}}$, deux bases G de S_{c_i} et la base $S^e_{x_m}[1]$ ou $S^e_{x_m}[2]$. On peut vérifier que la séquence S' obtenue est similaire à T.

($\Leftarrow$) Soit I une instance du problème EXACT 3-CNF-SAT avec n variables et q clauses. Soit $I' = ((S, P); (T, Q))$ une instance du problème APS(CROSSING,PLAIN) obtenue à partir d'une APS-CP-construction de I telle que (T, Q) est une sous-séquence commune arc-préservante de (S, P) et (T, Q). Notons $\mathcal{B}$ l'ensemble des bases de S à supprimer pour obtenir T. Par le Lemme 5.1, pour tout $1 \leq m \leq n$, soit toutes les bases de $S^s_{\overline{x_m}}$, soit toutes les bases de $S^s_{x_m}$ appartiennent à $\mathcal{B}$. Pour tout $1 \leq m \leq n$, nous définissons l'ensemble $\mathcal{V}$ de valeurs pour les n variables comme suit:

- si toutes les bases de $S^s_{\overline{x_m}}$ appartiennent à $\mathcal{B}$ alors $x_m = Vrai$;
- si toutes les bases de $S^s_{x_m}$ appartiennent à $\mathcal{B}$ alors $x_m = Faux$.

Maintenant, nous allons prouver que pour tout $1 \leq i \leq q$ la clause c_i est satisfaite par $\mathcal{V}$. Par le Lemme 5.1, pour tout $1 \leq i \leq q$ il existe une base G de S_{c_i} (disons la $j_i + 1^{ème}$) qui n'appartient pas à $\mathcal{B}$. Par construction, il existe un CG-arc de P entre $S_{c_i}[j_i + 1]$ et une base C de $S^s_{\overline{x_m}}$ (*resp.* $S^s_{x_m}$) si le $j_i^{ème}$ littéral de c_i est x_m (*resp.* $\overline{x_m}$).

Supposons, sans perte de généralité, que le $j_i^{ème}$ littéral de c_i est x_m. Étant donné que $Q = \emptyset$, au moins une base marquée par tout arc de P appartient à $\mathcal{B}$. Donc, la base C de $S^s_{\overline{x_m}}$ marquée avec $S_{c_i}[j_i + 1]$ par un CG-arc de P appartient à $\mathcal{B}$ (étant donné que par hypothèse $S_{c_i}[j_i + 1] \notin \mathcal{B}$). Par conséquent, d'après le Lemme 5.1, toutes les bases de $S^s_{\overline{x_m}}$ appartiennent à $\mathcal{B}$. D'après la définition de l'ensemble $\mathcal{V}$, $x_m = Vrai$ et donc c_i est satisfaite. Une conclusion similaire peut être obtenue si le $j_i^{ème}$ littéral de c_i est $\overline{x_m}$. La preuve du lemme en découle. □

Nous venons donc de prouver le théorème suivant.

Théorème 5.1. *Le sous-problème* APS(CROSSING,PLAIN) *est* **NP**-*complet.*

5.3 Raffinement du problème APS

Dans cette section, nous proposons un raffinement du problème APS. Nous commençons par définir formellement notre approche et par expliquer en quoi un tel raffinement est intéressant tant du point de vue théorique que pratique. Nous terminons cette section en donnant des propriétés triviales sur le raffinement proposé qui nous seront utiles par la suite dans la Section 5.4.

Il faut noter que le niveau de complexité UNLIMITED, ne correspondant à aucune restriction, est, de par sa complexité dans les cas les plus simples, d'un intérêt limité pour notre étude. Par conséquent, nous n'y ferons plus référence par la suite.

5.3.1 De nouveaux niveaux de complexité

Dans la Section 5.2, nous avons prouvé que le sous-problème APS(CROSSING, PLAIN) est **NP**-complet. Ce résultat répond au dernier cas ouvert concernant la complexité du problème APS dans le cadre des niveaux de complexité classiques, *i.e.* PLAIN, CHAIN, NESTED et CROSSING (*cf.* Table 5.1). Malgré tout, il est intéressant de déterminer précisément la frontière entre les cas polynomiaux et les cas **NP**-complets. En effet, les théoriciens autant que les praticiens sont à même de demander plus d'informations sur les cas difficiles du problème APS de manière à mieux cerner ce qui rend le problème difficile.

Dans cette section, nous proposons de raffiner les niveaux classiques de complexité en les scindant en sous-niveaux plus précis décrivant la structure des arcs. Pour ce faire, nous adaptons les trois relations introduites par Vialette [93, 94] dans le contexte des 2-intervalles (*cf.* Section 3.2) aux séquences arc-annotées. Étant donnés deux arcs $a_1 = (i, j)$ et $a_2 = (k, l)$ de P, on note

1. $a_1 < a_2$ si $i < j < k < l$;
2. $a_1 \sqsubset a_2$ si $k < i < j < l$;
3. $a_1 \bowtie a_2$ si $i < k < j < l$.

Nous adaptons également la notion de comparabilité. Deux arcs a_1 et a_2 sont τ-comparables pour un $\tau \in \{<, \sqsubset, \bowtie\}$ si $a_1 \tau a_2$ ou $a_2 \tau a_1$. Soient $\mathcal{P}$ l'ensemble des arcs et R un sous-ensemble non-vide de $\{<, \sqsubset, \bowtie\}$. L'ensemble $\mathcal{P}$ est dit *R-comparable* si étant donné tout couple d'arcs distincts (a_1, a_2) de $\mathcal{P}$, a_1 et a_2 sont τ-comparables pour un $\tau \in R$. Une séquence arc-annotée (S, P) est dite *R-arc-annotée* pour un sous-ensemble non vide R de $\{<, \sqsubset, \bowtie\}$ si P est R-comparable. Nous notons $R = \emptyset$ l'absence d'arc (*i.e.* $P = \emptyset$). Ces définitions permettent de définir huit niveaux de complexité: $\emptyset$, $\{<\}$, $\{\sqsubset\}$, $\{\bowtie\}$, $\{<, \sqsubset\}$, $\{<, \bowtie\}$, $\{\sqsubset, \bowtie\}$ et $\{<, \sqsubset, \bowtie\}$.

Bien que certains de ces niveaux semblent peu réalistes pour la représentation de molécules d'ARN, ils seront utiles pour déterminer précisément ce qui rend le problème difficile. Une telle étude va permettre de mieux définir les difficultés inhérentes au problème, et ainsi pourra faciliter la conception d'heuristiques et d'algorithmes d'approximation ou de complexité paramétrée.

Il faut noter que notre modèle ne prend pas en compte les séquences arc-annotées composées d'un seul arc. Malgré tout, la complexité du sous-problème où il n'y a qu'un seul arc est identique à celle du sous-problème sans arc. D'autre part, nous pouvons exprimer chaque niveau de complexité classique comme étant une combinaison de ces nouvelles relations:

- PLAIN correspond à $R = \emptyset$;
- CHAIN correspond à $R = \{<\}$;
- NESTED correspond à $R = \{<, \sqsubset\}$;
- CROSSING correspond à $R = \{<, \sqsubset, \bowtie\}$.

Ce qui est intéressant, c'est que notre raffinement permet également d'exprimer de nouveaux niveaux de complexité, à savoir: $R = \{\sqsubset\}$, $R = \{\bowtie\}$, $R = \{<, \bowtie\}$ et $R = \{\sqsubset, \bowtie\}$. Dans le cadre du problème APS, on peut définir une nouvelle hiérarchie de niveaux de complexité comme suit.

Propriété 5.1 (Hiérarchie des niveaux de complexité). $\{\alpha\} \subset \{\alpha, \beta\} \subset \{\alpha, \beta, \gamma\}$, $\forall \alpha, \beta, \gamma \in \{<, \sqsubset, \bowtie\}$ *tels que* $\{\alpha, \beta, \gamma\} = \{<, \sqsubset, \bowtie\}$.

Il faut noter que cette hiérarchie ne concerne que le problème APS et découle directement de la hiérarchie des niveaux de complexité classiques (présentée en page 34) et de leurs expressions en tant que combinaisons des nouvelles relations introduites dans cette section.

5.3.2 Résultats triviaux

Remarquons que, comme dans la Table 5.1, nous n'avons à considérer que les sous-problèmes APS(n_1,n_2) où les niveaux de complexité n_1 et n_2 sont compatibles, *i.e.* $n_2 \subset n_1$. En effet, dans le cas contraire, on peut répondre négativement puisqu'il existe au moins deux arcs de (T, Q) satisfaisant une relation de n_2 qui n'existe pas dans n_1, et donc (T, Q) ne peut pas être obtenue à partir de (S, P) en supprimant certaines de ses bases ainsi que les éventuels arcs incidents. Ces cas incompatibles sont notés par des zones hachurées de la Table 5.2.

Un certain nombre de résultats concernant les niveaux de complexité classiques permettent de répondre d'ores et déjà à certains cas de la nouvelle table de complexité. Nous commençons par définir une propriété sur la propagation de la complexité du problème APS.

Propriété 5.2. *Soient* APS(n_1, n_2) *et* APS(n_3, n_4) *deux sous-problèmes de* APS *tels que*

1. n_1, n_2, n_3 *et* n_4 *appartiennent à* $\{R|R \subseteq \{<, \sqsubset, \bowtie\}\}$ *et*
2. $n_1 \subset n_3$ *et* $n_2 \subset n_4$.

Si le sous-problème APS(n_1, n_2) *est* **NP***-complet (resp. le sous-problème* APS(n_3, n_4) *est polynomial) alors le sous-problème* APS(n_3, n_4) *(resp.* APS(n_1, n_2)*) l'est également.*

Gramm *et al.* ont démontré que le sous-problème APS(NESTED, NESTED) peut être résolu par un algorithme de complexité en temps $\mathbf{O}(nm)$ [57]. Conformément à notre modèle, ce résultat induit que le sous-problème APS($\{<, \sqsubset\}$,$\{<, \sqsubset\}$) peut être résolu par un algorithme de complexité en temps $\mathbf{O}(nm)$. En combinant ce résultat et la Propriété 5.2 on peut en déduire le théorème suivant.

Théorème 5.2. *Le sous-problème* APS(n_1,n_2) *peut être résolu par un algorithme de complexité en temps* $\mathbf{O}(nm)$, $\forall n_1, n_2 \in \{R|R \subseteq \{<, \sqsubset\}\}$ *tels que* $n_2 \subset n_1$.

De manière similaire, en constatant que la preuve de **NP**-complétude du sous-problème APS(CROSSING, CROSSING) de Evans [48] utilise des séquences $\{<, \sqsubset, \bowtie\}$-arc-annotées, on peut affirmer que cette preuve démontre que le sous-problème APS($\{<, \sqsubset, \bowtie\}$, $\{<, \sqsubset, \bowtie\}$) est **NP**-complet.

En prouvant que le sous-problèmeAPS(CROSSING, CHAIN) est **NP**-complet [58], Guo a prouvé également que le sous-problème APS($\{<, \sqsubset, \bowtie\}$, $\{<\}$) est **NP**-complet. Il faut noter, que d'après la Propriété 5.2, ces résultats impliquent que les sous-problèmes APS($\{<, \sqsubset, \bowtie\}$, $\{<, \sqsubset\}$) et APS($\{<, \sqsubset, \bowtie\}$,$\{<, \bowtie\}$) sont également **NP**-complet.

La preuve de **NP**-complétude du sous-problème APS(CROSSING,PLAIN) de la Section 5.2 utilise une séquence $\{\sqsubset, \bowtie\}$-arc-annotée (S, P) et une séquence sans arc (T, Q). Par conséquent le Théorème 5.1 implique le corrollaire suivant.

Corollaire 5.1. *Le sous-problème* APS($\{\sqsubset, \bowtie\}, \emptyset$) *est* **NP***-complet.*

Dans [22] nous avons prouvé le théorème suivant.

Théorème 5.3. *Le sous-problème* APS($\{<, \bowtie\}, \emptyset$) *est* **NP***-complet.*

La preuve consiste, tout comme celle du sous-problème APS(CROSSING,PLAIN), en une réduction du problème EXACT 3-CNF-SAT. Cette preuve assez technique et longue (21 pages) est disponible dans [22]. D'après la Propriété 5.2, la complexité des sous-problèmes APS($\{\sqsubset, \bowtie\}, \emptyset$) et APS($\{<, \bowtie\}, \emptyset$) implique le théorème suivant.

Théorème 5.4. *Le sous-problème* APS(n_1,n_2) *est* **NP***-complet,* $\forall n_1 \in \{\{<, \bowtie\}, \{\sqsubset, \bowtie\}, \{<, \sqsubset, \bowtie\}\}$ *et* $\forall n_2 \in \{R|R \subseteq \{<, \sqsubset, \bowtie\}\}$ *tels que* $n_2 \subset n_1$.

La Table 5.2 synthétise l'ensemble de ces résultats. Il en résulte que les deux derniers problèmes ouverts sont APS($\{\bowtie\}$,$\{\bowtie\}$) et APS($\{\bowtie\}$,$\emptyset$). Dans la section suivante, nous fournissons deux algorithmes polynomiaux complétant l'étude de ces nouveaux niveaux de complexité.

APS								
	$\{<,\sqsubset,\bowtie\}$	$\{\sqsubset,\bowtie\}$	$\{<,\bowtie\}$	$\{\bowtie\}$	$\{<,\sqsubset\}$	$\{\sqsubset\}$	$\{<\}$	$\emptyset$
$\{<,\sqsubset,\bowtie\}$	**NP**-C [48]	**NP-C [21, 22]**	**NP**-C [58]	**NP-C [21, 22]**	**NP**-C [58]	**NP-C [21, 22]**	**NP**-C [58]	**NP-C [21, 22]**
$\{\sqsubset,\bowtie\}$		**NP-C [21, 22]**	////	**NP-C [21, 22]**	////	**NP-C [21, 22]**	////	**NP-C [21, 22]**
$\{<,\bowtie\}$			**NP-C [21, 22]**	**NP-C [21, 22]**	////	////	**NP-C [21, 22]**	**NP-C [21, 22]**
$\{\bowtie\}$				?	////	////	////	?
$\{<,\sqsubset\}$					$O(nm)$ [57]	$O(nm)$ [57]	$O(nm)$ [57]	$O(nm)$ [57]
$\{\sqsubset\}$						$O(nm)$ [57]	////	$O(nm)$ [57]
$\{<\}$							$O(nm)$ [57]	$O(n+m)$ [57]
$\emptyset$								$O(n+m)$ [57]

Table 5.2 – Complexité du problème APS résultant du raffinement et de la Propriété 5.2. ////: cas incompatible. Par manque de place, nous utilisons la notation **NP**-C plutôt que **NP**-complet dans cette table.

5.4 Polynomialité de certains sous-problèmes de APS

Nous prouvons dans cette section qu'il existe des algorithmes polynomiaux pour résoudre les sous-problèmes APS($\{\bowtie\}$,$\emptyset$) et APS($\{\bowtie\}$,$\{\bowtie\}$). En d'autres termes, nous prouvons que la relation de croisement $\bowtie$ n'implique pas, à elle-seule, la **NP**-complétude.

Nous avons besoin des notations suivantes. Soit $S = S[1]\,S[2]\ldots S[m]$ une séquence de longueur m. Pour tout $1 \leq i \leq j \leq m$, $S[i:j]$ représente la séquence $S[i]\,S[i+1]\ldots S[j]$. L'*inverse* d'une séquence S, notée S^R, est la séquence $S^R = S[m]\ldots S[2]\,S[1]$. On appelle *factorisation* de S toute décomposition $S = x_1\,x_2\ldots x_q$ où $x_1, x_2, \ldots x_q$ sont des sous-séquences (éventuellement vides) de S. Soient (S,P) une séquence $\{\bowtie\}$-arc-annotée et $a_1 = (i,j)$ un arc de P tel que $i < j$. Nous appelons *base avant* de a_1, la base $S[i]$ et *base arrière* de a_1, la base $S[j]$. Par la suite, LF_S représentera la position de la dernière base avant de (S,P) et FB_S représentera la position de la première base arrière de (S,P), *i.e.* $\mathsf{LF}_S = \max\{i|(i,j) \in P\}$ et $\mathsf{FB}_S = \min\{j|(i,j) \in P\}$. Par convention, $\mathsf{LF}_S = 0$ et $\mathsf{FB}_S = |S|+1$ si $P = \emptyset$. On peut noter que $\mathsf{LF}_S < \mathsf{FB}_S$. Enfin, le symbole "$\cdot$" sera utilisé par la suite pour représenter la concaténation de deux séquences.

Lemme 5.3. *Soient (S,P) et (T,Q) deux séquences $\{\bowtie\}$-arc-annotées de longueurs respectives n et m. Si (T,Q) est une sous-séquence commune arc-préservante de (S,P) et (T,Q), alors il existe une factorisation (éventuellement triviale) $T[\mathsf{LF}_T+1 : \mathsf{FB}_T-1] = x \cdot y$ telle que $T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$ est une sous-séquence commune arc-préservante de $S[1 : \mathsf{FB}_S-1] \cdot S[\mathsf{FB}_S : n]^R$ et $T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$.*

Preuve. Supposons que (T,Q) soit une sous-séquence commune arc-préservante de (S,P) et (T,Q). Étant donné que (S,P) et (T,Q) sont toutes deux des séquences $\{\bowtie\}$-arc-annotées, il existe deux factorisations $S[1 : \mathsf{LF}_S] = u \cdot w$ et $S[\mathsf{FB}_S : n] = z \cdot v$ telles que:

1. $T[1 : \mathsf{LF}_T]$ apparaît dans u;
2. $T[\mathsf{LF}_T+1 : \mathsf{FB}_T-1]$ apparaît dans $w \cdot S[\mathsf{LF}_S+1 : \mathsf{FB}_S-1] \cdot z$ et
3. $T[\mathsf{FB}_T : m]$ apparaît dans v.

Il en découle qu'il existe une factorisation $T[\mathsf{LF}_T+1 : \mathsf{FB}_T-1] = x \cdot y$ telle que x apparaît dans $w \cdot S[\mathsf{LF}_S+1 : \mathsf{FB}_S-1]$ et y apparaît dans z. Par conséquent $T' = T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$ est une sous-séquence commune arc-préservante de $S' = S[1 : \mathsf{FB}_S-1] \cdot S[\mathsf{FB}_S : n]^R$ et T' (*cf.* Figure 5.2). □

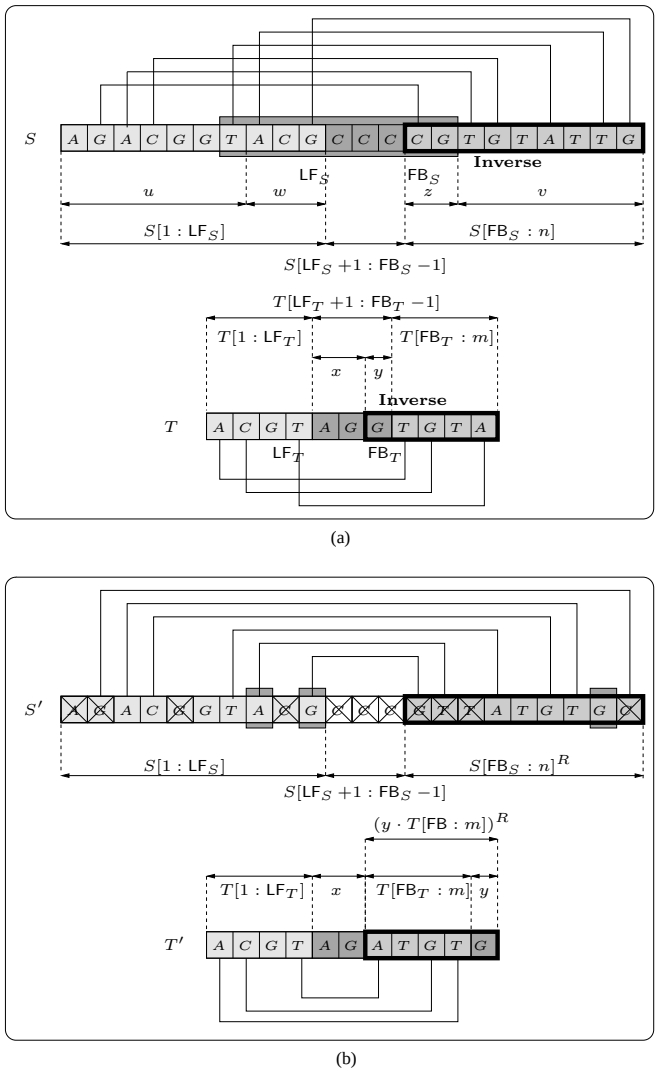


Figure 5.2 – Illustration du Lemme 5.3. (a) l'instance du sous-problème APS($\{\bowtie\}$,$\{\bowtie\}$). (b) l'instance du sous-problème APS($\{\sqsubset\}$,$\{\sqsubset\}$) correspondante.

Théorème 5.5. *Il existe un algorithme de complexité en temps* $\mathbf{O}(nm^2)$ *pour résoudre le sous-problème* APS($\{\bowtie\}$,$\{\bowtie\}$).

Preuve. Nous proposons l'Algorithme 1.

D'après le Lemme 5.3, on peut affirmer que l'Algorithme 1 résout bien le sous-problème APS($\{\bowtie\}$,$\{\bowtie\}$). Il nous reste à en prouver la complexité en temps. Il est clair que $S' = S[1 : \mathsf{FB}_S-1] \cdot S[\mathsf{FB}_S :$

Algorithme 1: Un algorithme de complexité en temps $\mathbf{O}(nm^2)$ pour résoudre le sous-problème APS($\{\bowtie\}$,$\{\bowtie\}$)

Données : Deux séquences $\{\bowtie\}$-arc-annotées (S, P) et (T, Q) de longueur respective n et m
Résultat : *Vrai* ssi (T, Q) est une sous-séquence commune arc-préservante de (S, P) et (T, Q)
début
1 $S' = S[1 : \mathsf{FB}_S - 1] \cdot S[\mathsf{FB}_S : n]^R$
2 **pour chaque** *factorisation* $T[\mathsf{LF}_T + 1 : \mathsf{FB}_T - 1]| = x \cdot y$ **faire**
3 $T' = T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$
4 **si** (T', Q) *est une sous-séquence commune arc-préservante de* (S', P) *et* (T', Q) **alors**
5 **retourner** Vrai
6 **retourner** Faux
fin

$n]^R$ est une séquence $\{\sqsubset\}$-arc-annotée. Il est important de remarquer que, pour toute factorisation $T[\mathsf{LF}_T + 1 : \mathsf{FB}_T - 1] = x \cdot y$, la séquence $T' = T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$ obtenue est une séquence $\{\sqsubset\}$-arc-annotée également.

Il y a au plus $m - 2|Q|$ itérations à faire avant de renvoyer éventuellement *Faux*. D'après la remarque précédente, le résultat de ligne 4 de l'Algorithme 1 est une instance du sous-problème APS($\{\sqsubset\}$,$\{\sqsubset\}$). Sachant que APS($\{\sqsubset\}$,$\{\sqsubset\}$) $\subset$ APS($\{<, \sqsubset\}$,$\{<, \sqsubset\}$), d'après la Propriété 5.2, ce problème peut être résolu à l'aide d'un algorithme de complexité en temps $\mathbf{O}(nm)$ [57]. Par conséquent, notre algorithme s'exécute dans sa totalité en $\mathbf{O}(nm(m - 2|Q|)) = \mathbf{O}(nm^2)$. □

Le preuve du Théorème 5.4 repose, clairement, sur l'optimalité de l'algorithme résolvant le sous-problème APS($\{\sqsubset\}$,$\{\sqsubset\}$): meilleure est la complexité du sous-problème APS($\{\sqsubset\}$,$\{\sqsubset\}$), meilleure sera celle du sous-problème APS($\{\bowtie\}$,$\{\bowtie\}$). Nous avons uniquement utilisé la propriété suivante: APS($\{\sqsubset\}$, $\{\sqsubset\}$) $\subset$ APS($\{<, \sqsubset\}$,$\{<, \sqsubset\}$). L'existence d'un algorithme de meilleure complexité pour le sous-problème APS($\{\sqsubset\}$,$\{\sqsubset\}$) reste une question ouverte.

D'après la Propriété 5.2, le Théorème 5.4 peut se généraliser comme suit.

Corollaire 5.2. *Il existe un algorithme de complexité en temps* $\mathbf{O}(nm^2)$ *pour résoudre le sous-problème* APS($\{\bowtie\}$,$\emptyset$).

5.5 Conclusion et perspectives

Dans ce chapitre, nous avons prouvé que le sous-problème APS(CROSSING,PLAIN) est **NP**-complet, répondant ainsi à une question ouverte posée dans [57] (voir Table 5.1). Ce résultat répond au dernier cas ouvert concernant le problème APS et considérant les niveaux classiques de complexité de structures, *i.e.* PLAIN, CHAIN, NESTED et CROSSING. De plus, nous avons affiné les quatre niveaux classiques de complexité de structures mentionnés précédemment afin d'explorer plus précisément la limite entre les cas polynomiaux et ceux **NP**-complets (voir Table 5.3).

Nous avons prouvé que les sous-problèmes APS($\{\sqsubset, \bowtie\}, \emptyset$) et APS($\{<, \bowtie\}, \emptyset$) sont, tous deux, **NP**-complets et nous avons proposé un algorithme polynomial pour résoudre les sous-problèmes APS($\{\bowtie\}$, $\emptyset$) et APS($\{\bowtie\}$,$\{\bowtie\}$). Par conséquent, le raffinement que nous avons proposé montre que le problème APS ne devient **NP**-complet que lorsque les séquences en entrée contiennent des arcs $\{\bowtie, \alpha\}$-

APS								
	$\{<,\sqsubset,\bowtie\}$	$\{\sqsubset,\bowtie\}$	$\{<,\bowtie\}$	$\{\bowtie\}$	$\{<,\sqsubset\}$	$\{\sqsubset\}$	$\{<\}$	$\emptyset$
$\{<,\sqsubset,\bowtie\}$	**NP**-C [48]	**NP-C [21, 22]**	**NP**-C [58]	**NP-C [21, 22]**	**NP**-C [58]	**NP-C [21, 22]**	**NP**-C [58]	**NP-C [21, 22]**
$\{\sqsubset,\bowtie\}$		**NP-C [21, 22]**	////	**NP-C [21, 22]**	////	**NP-C [21, 22]**	////	**NP-C [21, 22]**
$\{<,\bowtie\}$			**NP-C [22]**		////	////	**NP-C [22]**	
$\{\bowtie\}$				$O(nm^2)$ **[21, 22]**	////	////	////	$O(nm^2)$ **[21, 22]**
$\{<,\sqsubset\}$					$O(nm)$ [57]	$O(nm)$ [57]	$O(nm)$ [57]	$O(nm)$ [57]
$\{\sqsubset\}$						$O(nm)$ [57]	////	$O(nm)$ [57]
$\{<\}$							$O(nm)$ [57]	$O(n+m)$ [57]
$\emptyset$								$O(n+m)$ [57]

Table 5.3 – Complexité du problème APS résultant du raffinement. ////: cas incompatible.

comparables avec $\alpha \neq \emptyset$. Par conséquent, des arcs se croisant uniquement n'impliquent pas que le problème APS est difficile.

Il reste, à notre avis, nécessaire d'étudier la complexité paramétrée du problème APS. Nous pensons que les paramètres liés à la structure des arcs que sont :

- la *largeur* – *i.e.* la cardinalité maximale d'un sous-ensemble d'arcs $\{<\}$-comparable;
- la *profondeur* – *i.e.* la cardinalité maximale d'un sous-ensemble d'arcs $\{\bowtie\}$-comparable et
- la *hauteur* – *i.e.* la cardinalité maximale d'un sous-ensemble d'arcs $\{\sqsubset\}$-comparable

devraient nous permettre de définir des algorithmes de complexité paramétrée performants.

Finalement, comme nous l'avons précisé, l'existence d'un algorithme de meilleure complexité pour le sous-problème APS($\{\sqsubset\}$,$\{\sqsubset\}$) reste une question ouverte. Si un tel algorithme existe, il serait alors possible de parvenir également à une meilleure complexité pour les sous-problèmes APS($\{\bowtie\}$,$\{\bowtie\}$) et APS($\{\bowtie\}$,$\emptyset$).

CHAPITRE 6

La recherche de motifs de 2-intervalles

6.1 Introduction

Dans ce chapitre, nous développons des résultats publiés dans [24] et obtenus en collaboration avec Guillaume Fertin et Stéphane Vialette. Nous rappelons que le problème de recherche de motifs de 2-intervalles – noté 2-IP – se définit comme suit.

2-INTERVAL PATTERN (2-IP)
DONNÉES*: Un ensemble de 2-intervalles $\mathcal{D}$, un modèle de comparaison $R \subseteq \{<, \sqsubset, \bowtie\}$ et un entier positif k.*
QUESTION*: Existe-il un sous-ensemble $\mathcal{D}' \subseteq \mathcal{D}$ de cardinalité supérieure ou égale à k, tel que $\mathcal{D}'$ soit R-comparable ?*

Comme nous l'avons précisé dans la Section 3.2, les complexités des sous-problèmes 2-IP($n_1, \{<, \bowtie\}$) avec $n_1 \in \{\text{DISJOINT}, \text{UNITARY}, \text{UNLIMITED}\}$ et 2-IP(DISJOINT,$\{\sqsubset, \bowtie\}$) représentent les quatre derniers cas ouverts pour le problème 2-IP.

Dans ce chapitre, nous répondons à trois des quatre cas ouverts et nous améliorons la complexité d'un cinquième, comme illustré dans la Table 6.1. De plus, nous démontrons qu'il existe un algorithme de complexité paramétrée pour résoudre le sous-problème 2-IP(UNITARY,$\{<, \bowtie\}$) de complexité en temps $\mathbf{O}(n^2 \cdot \mathsf{FCrossing}(\mathcal{D}) \cdot 2^{\mathsf{FCrossing}(\mathcal{D})}(\log(n) + \mathsf{FCrossing}(\mathcal{D})))$ (le paramètre FCrossing est défini ci-après).

2-IP			
	SUPPORT		
MODÈLE	UNLIMITED	UNITARY	DISJOINT
$\{<, \sqsubset, \bowtie\}$	**NP**-complet [94]		$\mathbf{O}(n\sqrt{n})$[78]
$\{\sqsubset, \bowtie\}$	**NP**-complet [94]		$\mathbf{O}(n^2\sqrt{n})$ **[24]**
$\{<, \sqsubset\}$	$\mathbf{O}(n^2)$ [94]		
$\{<, \bowtie\}$	**NP-complet [24]**		?
$\{<\}$	$\mathbf{O}(n \log n)$ [94]		
$\{\sqsubset\}$	$\mathbf{O}(n \log n)$ **[24]**		
$\{\bowtie\}$	$\mathbf{O}(n^2 \log n)$ [94]		

Table 6.1 – Complexité du problème 2-IP avec $n = |\mathcal{D}|$.

Par la suite, étant donné un 2-intervalle $D = (I, J)$, on note $\mathsf{Gauche}(D) = I$ et $\mathsf{Droite}(D) = J$. Si $[x : y]$ et $[x' : y']$ sont deux intervalles tels que $[x : y] < [x' : y']$, nous écrirons par moments $D = ([x : y], [x' : y'])$ pour souligner la définition précise du 2-intervalle D. L'*intervalle couvrant* d'un 2-intervalle D, noté $\mathsf{ICouvrant}(D)$, est le plus petit intervalle couvrant à la fois $\mathsf{Gauche}(D)$ et $\mathsf{Droite}(D)$ (*i.e.* incluant l'ensemble des points appartenant à $\mathsf{Gauche}(D)$ et $\mathsf{Droite}(D)$). L'élément le *plus à gauche* (*resp. plus à droite*) d'un ensemble de 2-intervalles disjoints $\mathcal{D}$ désigne le 2-intervalle $D_i \in \mathcal{D}$ tel que $\mathsf{Gauche}(D_i) < \mathsf{Gauche}(D_j)$ (*resp.* $\mathsf{Droite}(D_j) < \mathsf{Droite}(D_i)$) pour tout $D_j \in \mathcal{D} - D_i$. Observons qu'il est possible qu'un unique 2-intervalle D_i soit à la fois l'élément le plus à gauche et le plus à droite de $\mathcal{D}$ (c'est en effet le cas si $|\mathcal{D}| = 1$ ou si $D_j \sqsubset D_i$ pour tout $D_j \in \mathcal{D} - D_i$).

Nous définissons, ci-après, deux paramètres sur un ensemble de 2-intervalles $\mathcal{D}$: la *profondeur* et le *nombre maximum de croisements à droite*.

Définition 6.1 (Profondeur – Depth). La *profondeur* d'un ensemble de 2-intervalles $\mathcal{D}$, notée $\mathsf{Depth}(\mathcal{D})$, est la cardinalité maximale d'un sous-ensemble $\{\bowtie\}$-comparable de $\mathcal{D}$.

Définition 6.2 (Nombre maximum de croisements à droite – $\mathsf{FCrossing}$). Le *nombre maximum de croisements à droite* d'un ensemble de 2-intervalles $\mathcal{D}$, noté $\mathsf{FCrossing}(\mathcal{D})$, est défini par $\mathsf{FCrossing}(\mathcal{D}) = \max_{D_i \in \mathcal{D}} |\{D_j : D_i \bowtie D_j\}|$.

Clairement, $\mathsf{Depth}(\mathcal{D}) - 1 \leq \mathsf{FCrossing}(\mathcal{D})$.

Nous présentons ci-après quelques notions de théorie des graphes issues de [55, 43].

Définition 6.3 (Sous-graphe). Un *sous-graphe* d'un graphe $G = (V, E)$ est un graphe $G' = (V', E')$ où $V' \subseteq V$ et pour tout $(i, j) \in E$, si $i \in V'$ et $j \in V'$ alors $(i, j) \in E'$.

Définition 6.4 (Graphe induit). Soient G un graphe et $V' \subseteq \mathbf{V}(G)$, le *graphe induit* $G[V']$ est un graphe tel que $\mathbf{V}(G[V']) = V'$ et $\mathbf{E}(G[V']) = \{(u, v) \in \mathbf{E}(G) | u, v \in V'\}$.
Étant donné $E' \subseteq \mathbf{E}(G)$, le *graphe induit par les arêtes* $G[E']$ est un graphe tel que $\mathbf{E}(G[E']) = E'$ et $\mathbf{V}(G[E']) = \{v \in e | e \in E'\}$.

Définition 6.5 (Graphe complet). Un graphe $G = (V, E)$ est *complet* s'il existe une arête dans E reliant tout couple de sommets de V.

Définition 6.6 (Graphe d'intersection). Étant donné un ensemble $\mathcal{F}$, le *graphe d'intersection* de $\mathcal{F}$, noté $G = \Omega(\mathcal{F})$, est obtenu en (1) associant à chaque sommet de G un sous-ensemble de $\mathcal{F}$ et en (2) ajoutant une arête entre deux sommets si l'intersection des sous-ensembles correspondants est non vide.

Définition 6.7 (Graphe d'intervalle). Un *graphe d'intervalle* est un graphe d'intersection d'un ensemble fini d'intervalles.

Définition 6.8 (Graphe de trapézoïde). Un *graphe de trapézoïde* est un graphe d'intersection d'un ensemble fini de trapèzes entre deux lignes parallèles.

Définition 6.9 (Graphe de comparabilité). Un graphe $G = (V, E)$ est *de comparabilité* si on peut orienter ses arêtes de façon transitive: *i.e.* pour tout arc $(u, v) \in E$, s'il existe un arc $(v, w) \in E$ alors il existe un arc $(u, w) \in E$.

Définition 6.10 (Couplage). Un *couplage* dans un graphe $G = (V, E)$ est un sous-ensemble d'arêtes $E' \subseteq E$ tel que le graphe induit par E' est de degré maximum 1.

Définition 6.11 (Clique). Une *clique* dans un graphe G est un sous-graphe complet de G.

6.2 Amélioration de la complexité du sous-problème 2-IP(UNLIMITED, $\{\sqsubset\}$)

Le sous-problème 2-IP(UNLIMITED,$\{\sqsubset\}$) a été considéré dans [93, 94]. Ce sous-problème étant équivalent à rechercher la plus grande clique dans un graphe de comparabilité (problème pour lequel il existe un algorithme linéaire [55]), un algorithme de complexité $\mathbf{O}(n^2)$ (où $n = |\mathcal{D}|$) a été proposé. Nous améliorons ce résultat en proposant un algorithme optimal de complexité $\mathbf{O}(n \log n)$.

L'inefficacité de l'algorithme proposé dans [94] réside dans la construction du graphe de comparabilité qui ne peut se faire qu'en temps quadratique. Nous montrons que cette construction peut être évitée en considérant des trapèzes à la place des 2-intervalles. Nous notons $T = ([x : y], [x' : y'])$ le trapèze entre deux lignes parallèles ayant pour intervalle sur la ligne du haut $[x : y]$ et sur la ligne du bas $[x' : y']$.

Proposition 6.1. *Le sous-problème* 2-IP(UNLIMITED,$\{\sqsubset\}$) *peut être résolu par un algorithme de complexité* $\mathbf{O}(n \log n)$ *où* n *est le nombre de 2-intervalles en entrée.*

Preuve. Soit $\mathcal{D} = \{D_i | 1 \leq i \leq n\}$ une collection de n 2-intervalles sur une ligne. Nous construisons une collection de trapèzes $\mathcal{T} = \{T_1, T_2, \ldots T_n\}$ entre deux lignes parallèles comme suit: pour chaque 2-intervalle $D_i = ([x : y], [x' : y']) \in \mathcal{D}$, nous ajoutons un trapèze $T_i = ([x : y], [-y' : -x'])$ à $\mathcal{T}$ (comme illustré en Figure 6.1).

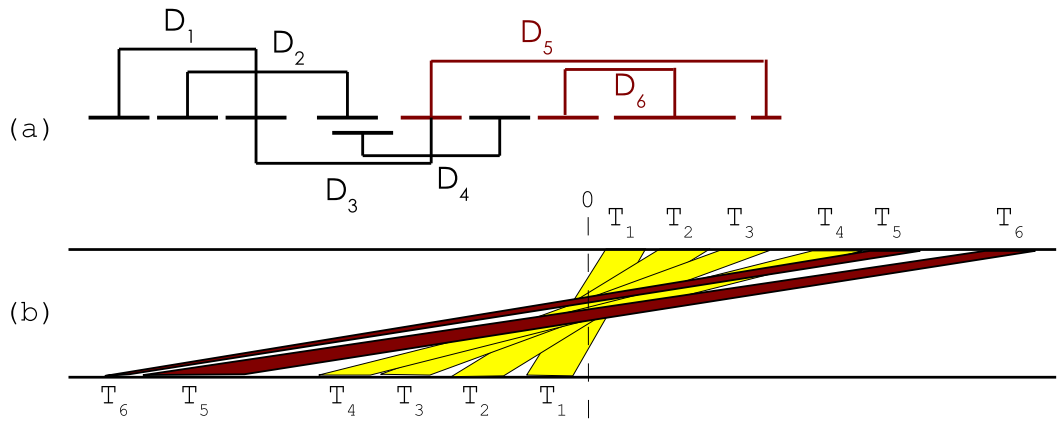


Figure 6.1 – Illustration de la construction d'une collection de trapèzes à partir d'une collection de 2-intervalles. (a) une collection $\mathcal{D}$ de 2-intervalles et (b) la collection de trapèzes correspondante.

Propriété 6.1. *Pour tout* $1 \leq i \leq j \leq n$*, les 2-intervalles* D_i *et* D_j *sont* $\{\sqsubset\}$*-comparables si et seulement si les trapèzes* T_i *et* T_j *ne s'intersectent pas.*

Clairement, la collection $\mathcal{T}$ peut être construite en $\mathbf{O}(n)$. En se basant sur la représentation géométrique des graphes de trapézoïde sous forme de boîtes dans le plan, Felsner *et al.* [50] ont défini un algorithme de complexité $\mathbf{O}(n \log n)$ pour trouver une sous-collection de cardinalité maximum de trapèzes ne s'intersectant pas dans une collection de trapèzes. La preuve de la proposition en découle. □

De plus, en se basant sur les résultats de [51], Felsner *et al.* [50] ont démontré que leur algorithme de complexité $\mathbf{O}(n \log n)$ est optimal. Il découle donc de la Proposition 6.1 que notre algorithme de complexité $\mathbf{O}(n \log n)$ pour résoudre le sous-problème 2-IP(UNLIMITED,$\{\sqsubset\}$) est lui aussi optimal.

6.3 Polynomialité du sous-problème 2-IP(DISJOINT,$\{\sqsubset, \bowtie\}$)

Dans cette section, nous présentons un algorithme de complexité $\mathbf{O}(n^2\sqrt{n})$ pour résoudre le sous-problème 2-IP(DISJOINT,$\{\sqsubset, \bowtie\}$), avec $n = |\mathcal{D}|$. Rappelons qu'étant donné un ensemble de 2-intervalles $\mathcal{D}$ sur un support DISJOINT, ce sous-problème consiste à rechercher un sous-ensemble $\mathcal{D}' \subseteq \mathcal{D}$ qui soit $\{\sqsubset, \bowtie\}$-comparable et de cardinalité maximum.

Notre algorithme se déroule en trois étapes. Tout d'abord, on construit le graphe d'intervalle de tous les intervalles couvrants des 2-intervalles de $\mathcal{D}$. Puis, toutes les cliques maximales de ce graphe sont calculées. Finalement, pour chaque clique maximale nous calculons le nombre maximum de 2-intervalles correspondants et disjoints deux à deux. La taille de la meilleure solution trouvée durant la troisième étape est alors retournée. Clairement, l'efficacité de notre algorithme dépend de l'efficacité de l'algorithme pour trouver toutes les cliques maximales du graphe d'intervalle de tous les intervalles couvrants. Nous détaillons ci-après notre algorithme.

Soit $\mathcal{D} = \{D_i | 1 \leq i \leq n\}$ un ensemble de n 2-intervalles. Considérons l'ensemble $\mathcal{C}_\mathcal{D}$ composé de tous les intervalles couvrants des 2-intervalles de $\mathcal{D}$, *i.e.* $\mathcal{C}_\mathcal{D} = \{\mathsf{ICouvrant}(D) | D \in \mathcal{D}\}$. Soit $\Omega(\mathcal{C}_\mathcal{D})$ le graphe d'intervalle de l'ensemble $\mathcal{C}_\mathcal{D}$. Le graphe $\Omega(\mathcal{C}_\mathcal{D})$ a un sommet v_i pour chaque intervalle $\mathsf{ICouvrant}(D_i)$ de $\mathcal{C}_\mathcal{D}$ et deux sommets v_i et v_j de $\Omega(\mathcal{C}_\mathcal{D})$ sont reliés par un arc si les deux intervalles correspondants $\mathsf{ICouvrant}(D_i)$ et $\mathsf{ICouvrant}(D_j)$ s'intersectent. Une illustration de $\mathcal{C}_\mathcal{D}$ et $\Omega(\mathcal{C}_\mathcal{D})$ pour un ensemble donné de 2-intervalles $\mathcal{D}$ est donnée dans la Figure 6.2. Le plus grand intérêt du graphe d'intervalle $\Omega(\mathcal{C}_\mathcal{D})$ provient du lemme suivant.

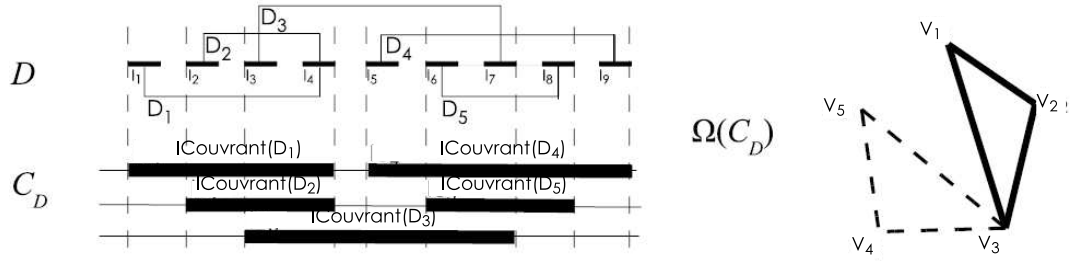


Figure 6.2 – L'ensemble $\mathcal{C}_\mathcal{D}$ et le graphe $\Omega(\mathcal{C}_\mathcal{D})$ pour un ensemble donné de 2-intervalles $\mathcal{D} = \{D_1, D_2, D_3, D_4, D_5\}$.

Lemme 6.1. *Soient $\mathcal{D}$ un ensemble de 2-intervalles et $\mathcal{D}'$ un sous-ensemble $\{\sqsubset, \bowtie\}$-comparable de $\mathcal{D}$. Alors, $C = \{v_i | D_i \in \mathcal{D}'\}$ est une clique de $\Omega(\mathcal{C}_\mathcal{D})$.*

Preuve. Soient D_i et D_j deux 2-intervalles disjoints de $\mathcal{D}'$. Étant donné que D_i et D_j sont $\{\sqsubset, \bowtie\}$-comparables, il en découle que soit les intervalles $\mathsf{ICouvrant}(D_i)$ et $\mathsf{ICouvrant}(D_j)$ se chevauchent, soit un des deux est entièrement inclus dans l'autre. Dans les deux cas, les intervalles $\mathsf{ICouvrant}(D_i)$ et $\mathsf{ICouvrant}(D_j)$ s'intersectent, et par conséquent les sommets v_i et v_j sont reliés par un arc dans $\Omega(\mathcal{C}_\mathcal{D})$. On en conclut que $C = \{v_i | D_i \in \mathcal{D}'\}$ est une clique de $\Omega(\mathcal{C}_\mathcal{D})$. □

Observons que la réciproque est fausse: étant donnés deux sommets d'une clique de $\Omega(\mathcal{C}_\mathcal{D})$, il est possible que les 2-intervalles correspondants ne soient pas $\{\sqsubset, \bowtie\}$-comparables.

Un certain nombre de problèmes **NP**-complets sur des classes générales de graphes sont polynomiaux pour les graphes d'intervalle. Le problème de la recherche de toutes les cliques maximales d'un graphe fait partie de ces problèmes. En effet, toutes les cliques maximales d'un graphe d'intervalle $G = (V, E)$

peuvent être trouvées en $\mathbf{O}(n+m)$, avec $n = |V|$ et $m = |E|$, par une modification de l'algorithme Maximum Cardinality Search (MCS) [16, 89]. De plus, un graphe d'intervalle $G = (V, E)$ a au plus $|V|$ cliques maximales [52].

Soit C une clique maximale de $\Omega(\mathcal{C}_\mathcal{D})$. Comme observé ci-dessus, il est possible que les 2-intervalles correspondants à toute paire de sommets distincts dans la clique maximale C ne soient pas $\{\sqsubset, \bowtie\}$-comparables. Soit $\mathcal{D}' \subseteq \mathcal{D}$ l'ensemble de tous les 2-intervalles correspondants aux sommets de la clique maximale C. Considérant C, soit le graphe $G_C = (V_C, E_C)$ tel que $V_C = \{v_I, v_J | D = (I, J) \in \mathcal{D}'\}$ et $E_C = \{(v_I, v_J) | (I, J) \in \mathcal{D}'\}$. Le lemme suivant est une conséquence immédiate de la définition de G_C et du Lemme 6.1.

Lemme 6.2. *Soient C une clique de $\Omega(\mathcal{C}_\mathcal{D})$ et G_C un graphe construit comme détaillé précédemment. Alors, $\{(I_1, J_1), (I_2, J_2), \ldots (I_k, J_k)\}$ est un sous-ensemble $\{\sqsubset, \bowtie\}$-comparable si et seulement si $\{(I_1, J_1), (I_2, J_2), \ldots (I_k, J_k)\}$ est un couplage de G_C.*

Proposition 6.2. *Le sous-problème* 2-IP(DISJOINT,$\{\sqsubset, \bowtie\}$) *peut être résolu par un algorithme de complexité en temps $\mathbf{O}(n^2\sqrt{n})$, avec $n = |\mathcal{D}|$.*

Preuve. Considérons l'Algorithme 2. D'après les Lemmes 6.1 et 6.2, on peut affirmer que cet algorithme résout bien le sous-problème 2-IP(DISJOINT,$\{\sqsubset, \bowtie\}$). Il reste à en prouver la complexité. Clairement, le graphe d'intervalle $\Omega(\mathcal{C}_\mathcal{D})$ peut être obtenu en $\mathbf{O}(n^2)$. Toutes les cliques maximales de $\Omega(\mathcal{C}_\mathcal{D})$ peuvent être trouvées en $\mathbf{O}(n+m)$, où m est le nombre d'arêtes de $\Omega(\mathcal{C}_\mathcal{D})$ [16, 89]. En résumé, les deux premières étapes peuvent être exécutées en $\mathbf{O}(n^2)$ puisque $m < n^2$. De plus, pour chaque clique maximale C de $\Omega(\mathcal{C}_\mathcal{D})$, le graphe G_C peut être construit en $\mathbf{O}(n)$ étant donné que $|C| \leq n$.

Nous considérons, maintenant, le calcul du couplage maximal de G_C. Micali et Vazirani [78] (voir aussi [92]) ont donné un algorithme en $\mathbf{O}(\sqrt{|V|}|E|)$ pour trouver un couplage maximal dans un graphe $G = (V, E)$. Sachant que G_C a au plus n arêtes (puisque chaque arête correspond à un 2-intervalle) et donc au plus $2n$ sommets, un couplage maximal $\mathcal{M}$ de G_C peut être trouvé en $\mathbf{O}(n\sqrt{n})$. Étant donné que $\Omega(\mathcal{C}_\mathcal{D})$ est un graphe d'intervalle avec n sommets, il a au plus n cliques maximales [52] dans $\Omega(\mathcal{C}_\mathcal{D})$. Nous en concluons que l'algorithme a une complexité globale en $\mathbf{O}(n^2\sqrt{n})$. □

Algorithme 2: Un algorithme de complexité $\mathbf{O}(n^2\sqrt{n})$ pour résoudre le sous-problème 2-IP(DISJOINT,$\{\sqsubset, \bowtie\}$).

Données : Un ensemble de 2-intervalles $\mathcal{D}$ avec un support DISJOINT.

Résultat : La cardinalité du sous-ensemble $\{\sqsubset, \bowtie\}$-comparable de $\mathcal{D}$ de cardinalité maximum.

début

1 Construire le graphe d'intervalle $\Omega(\mathcal{C}_\mathcal{D})$
2 Calculer toutes les cliques maximales de $\Omega(\mathcal{C}_\mathcal{D})$
3 **pour chaque** *clique maximale C de $\Omega(\mathcal{C}_\mathcal{D})$* **faire**
4 Construire le graphe G_C
5 Calculer un couplage maximal $\mathcal{M}$ de G_C
6 Stocker la cardinalité de $\mathcal{M}$ dans $m(C)$
7 **retourner** $\max\{m(C) | C$ *est une clique maximale de* $\Omega(\mathcal{C}_\mathcal{D})\}$

fin

6.4 NP-complétude du sous-problème 2-IP(UNITARY,$\{<,\bowtie\}$)

Le Théorème 6.1 qui suit complète l'analyse des sous-problèmes 2-IP(n_1, n_2) pour tout $n_1 \in \{$UNITARY, UNLIMITED$\}$ et $n_2 \subseteq \{<, \sqsubset, \bowtie\}$ (*cf.* Table 6.1).

Théorème 6.1. *Le sous-problème* 2-IP(UNITARY,$\{<,\bowtie\}$) *est* **NP***-complet.*

Tout d'abord, nous présenterons les deux problèmes de décisions que nous allons utiliser. Puis, nous donnerons un ensemble de lemmes intermédiaires qui seront finalement utilisés dans la Proposition 6.2 pour prouver la **NP**-complétude du problème 2-IP(UNITARY,$\{<,\bowtie\}$). Nous fournissons, ci-après, une transformation polynomiale du problème **NP**-complet EXACT 3-CNF-SAT [53] vers le sous-problème 2-IP(UNITARY,$\{<,\bowtie\}$). Nous rappelons que le problème EXACT 3-CNF-SAT se définit comme suit.

EXACT 3-CNF-SAT
DONNÉES*: Un ensemble $\mathcal{V}_n$ de n variables et un ensemble $\mathcal{C}_q$ de q clauses (chacune composée de trois littéraux) sur $\mathcal{V}_n$.*
QUESTION*: Existe-il un ensemble de valeurs pour $\mathcal{V}_n$ satisfaisant toutes les clauses de $\mathcal{C}_q$?*

Nous rappelons également la définition formelle du sous-problème 2-IP(UNITARY,$\{<,\bowtie\}$).

2-IP(UNITARY,$\{<,\bowtie\}$)
DONNÉES*: Un ensemble de 2-intervalles $\mathcal{D}$ et un entier positif k.*
QUESTION*: Existe-il un sous-ensemble $\mathcal{D}' \subseteq \mathcal{D}$ de cardinalité supérieure ou égale à k, tel que $\mathcal{D}'$ soit $\{<,\bowtie\}$-comparable ?*

Le problème 2-IP(UNITARY,$\{<,\bowtie\}$) appartient clairement à la classe **NP**. Nous allons démontrer que ce sous-problème est également **NP**-dur.

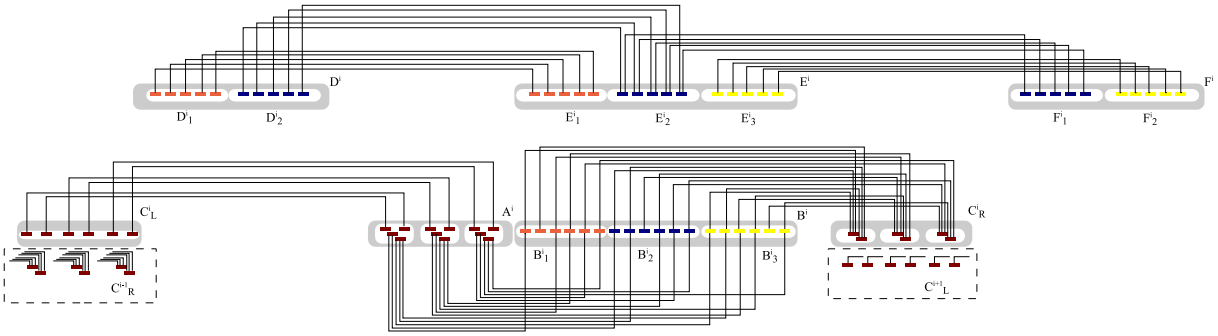


Figure 6.3 – Représentation d'une clause $c_i = (\overline{x_1} \vee x_2 \vee x_3)$ en considérant que x_1, x_2 et x_3 sont les seules variables (*i.e.* $n = 3$).

Afin de bien distinguer les intervalles et les 2-intervalles, nous appelons *intervalle simple* tout intervalle de Support($\mathcal{D}$). Soient $\mathcal{V}_n = \{x_1, x_2, \ldots, x_n\}$ un ensemble de n variables et $\mathcal{C}_q = \{c_1, c_2, \ldots, c_q\}$ une collection de q clauses. Nous commencerons par détailler la représentation d'une unique clause c_i de $\mathcal{C}_q$ comme illustrée en Figure 6.3. Le rectangle en pointillé sur la gauche (*resp.* droite) fait partie de la représentation de la clause c_{i-1} (*resp.* c_{i+1}). L'ajustement précis des représentations de deux clauses consécutives est illustré en Figure 6.6.

Par souci de lisibilité, nous séparons la représentation de c_i en sept groupes (représentés en gris): A^i, B^i, C^i_L, C^i_R, D^i, E^i et F^i. De plus, chacun de ces groupes est lui-même divisé en blocs (représentés en blanc). En tout, il y a $11 + 2n$ blocs pour chaque clause: n blocs pour le groupe A^i; 3 blocs pour le groupe B^i; 1 bloc pour le groupe C^i_L; n blocs pour le groupe C^i_R; 2 blocs pour le groupe D^i; 3 blocs pour le groupe E^i; 2 blocs pour le groupe F^i.

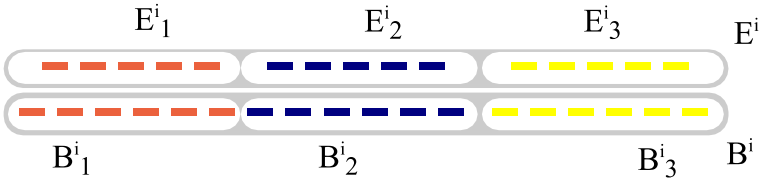


Figure 6.4 – Disposition des groupes B^i et E^i.

Dans l'exemple illustré en Figure 6.3, nous utilisons trois variables booléennes et par conséquent nous obtenons dix-sept blocs. Par souci de lisibilité, dans les figures de cette section, les intervalles du support seront dessinés sur différents niveaux. Il est également important de noter que chaque bloc E^i_j du groupe E^i et chaque bloc B^i_j du groupe B^i est disposé comme illustré en Figure 6.4. Ainsi, pour tout $1 \leq j \leq 3$, chaque intervalle simple du bloc E^i_j intersecte exactement deux intervalles simples du bloc B^i_j.

Nous allons maintenant donner une définition précise de chaque groupe de la représentation d'une clause donnée (comme partiellement illustrée en Figure 6.5). Le groupe A^i est composé de n blocs (un bloc pour chaque variable booléenne de $\mathcal{V}_n$) contenant chacun quatre intervalles simples. Les $4n$ intervalles simples du groupe A^i représentent, de gauche à droite, $\overline{x_1}, x_1, \overline{x_1}, x_1, \overline{x_2}, x_2, \overline{x_2}, x_2, \ldots \overline{x_n}, x_n, \overline{x_n}, x_n$. Par construction, dans tout bloc du groupe A^i, le second (*resp.* troisième) intervalle simple chevauche à la fois le premier et le troisième (*resp.* le second et le quatrième) intervalle simple.

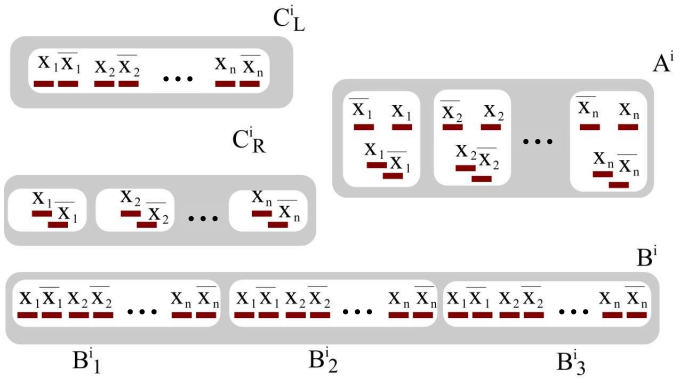


Figure 6.5 – Vue d'ensemble des groupes C^i_L, A^i, B^i, C^i_R.

Le groupe B^i est composé de trois blocs (un pour chaque littéral de la clause c_i) contenant chacun $2n$ intervalles simples. Les $2n$ intervalles simples de chaque bloc du groupe B^i représentent, de gauche à droite, $x_1, \overline{x_1}, x_2, \overline{x_2} \ldots x_n, \overline{x_n}$. Le groupe C^i_L est composé d'un bloc contenant $2n$ intervalles simples. Les $2n$ intervalles simples du groupe C^i_L représentent, de gauche à droite, $x_1, \overline{x_1}, x_2, \overline{x_2} \ldots x_n, \overline{x_n}$.

Le groupe C^i_R est composé de n blocs (un bloc pour chaque variable booléenne de $\mathcal{V}_n$) contenant chacun deux intervalles simples. Les $2n$ intervalles simples du groupe C^i_R répresentent, de gauche à

droite, $x_1, \overline{x_1}, x_2, \overline{x_2} \ldots x_n, \overline{x_n}$. Par construction, dans chaque bloc du groupe C_R^i les deux intervalles simples composant ce bloc se chevauchent.

Les groupes D^i et F^i sont tous les deux composés de deux blocs contenant chacun $2n-1$ intervalles simples. Le groupe E^i est composé de trois blocs contenant chacun $2n-1$ intervalles simples.

L'ensemble de 2-intervalles $\mathcal{D}$ de l'instance du problème 2-IP(UNITARY,$\{<, \bowtie\}$) est obtenu en assemblant dans l'ordre les représentations des clauses c_1 à c_q tel que, pour tout couple de clauses consécutives (c_{i-1}, c_i) avec $2 \leq i \leq q$, les groupes C_R^{i-1} et C_L^i sont ajustés comme illustré en Figure 6.6. Notons que les 2-intervalles entre le groupe C_R^{i-1} et le groupe B^{i-1} sont dessinés au-dessous du groupe C_R^{i-1} dans la Figure 6.6 tandis qu'ils sont dessinés au-dessus dans la Figure 6.3.

Il nous reste à définir formellement le paramètre k: $k = (7n-2) \cdot q$. Clairement, cette construction peut être effectuée en temps polynomial.

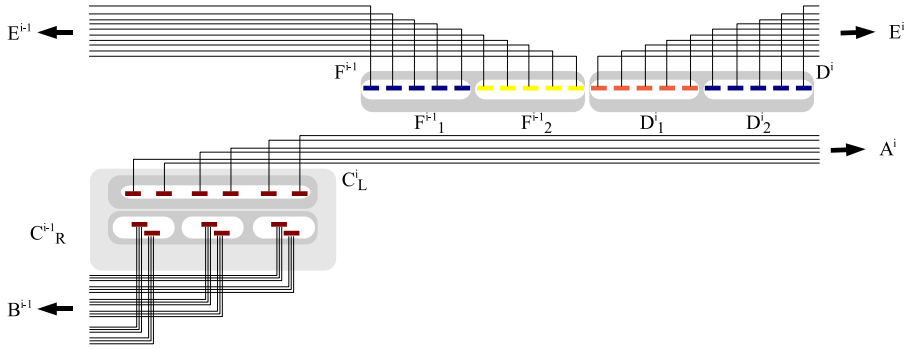


Figure 6.6 – Représentation de la jonction entre la représentation des clauses c_{i-1} et c_i.

Nous allons donner maintenant une description intuitive des différents éléments de l'ensemble de 2-intervalles que nous venons de construire.

Le bloc B_1^i représente la valeur du premier littéral de la clause c_i. Pour ce faire, le 2-intervalle entre l'intervalle simple de B_1^i correspondant à x_m (*resp.* $\overline{x_m}$) et l'intervalle simple du $m^{ème}$ bloc du groupe A^i est omis si le premier littéral de c_i est x_m (*resp.* $\overline{x_m}$). Les blocs B_2^i et B_3^i représentent respectivement (de façon similaire à B_1^i) le second et le troisième littéral de la clause c_i.

Par exemple, dans la Figure 6.7, l'absence du 2-intervalle entre l'intervalle simple correspondant à $\overline{x_1}$ dans B_1^i et un intervalle simple du groupe A^i indique que le premier littéral de la clause c_i est $\overline{x_1}$. De façon similaire, l'absence du 2-intervalle entre l'intervalle simple correspondant à x_2 (*resp.* x_3) dans B_2^i (*resp.* B_3^i) et un intervalle simple du groupe A^i indique que le second (*resp.* troisième) littéral de la clause c_i est x_2 (*resp.* x_3).

Intuitivement, la séquence de blocs $(C_R^{i-1}, C_L^i, A^i, B^i, C_R^i)$ correspond à un mécanisme qui propage la valeur de chaque variable de $\mathcal{V}_n$. Les blocs (D^i, E^i, F^i) correspondent à un mécanisme de sélection de littéral qui indique, pour chaque clause c_i, le littéral (*i.e.* le premier, second ou troisième) qui satisfait c_i. Notons que ces deux notions intuitives seront détaillées et clarifiées par la suite.

Nous débuterons cette preuve par quelques propriétés (Lemmes 6.3 à 6.8) concernant la cardinalité maximale d'un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable de $\mathcal{D}$ dans notre construction. Puis, ces résultats seront utilisés dans la Proposition 6.2 pour prouver la **NP**-complétude du sous-problème. Dans le reste de ce chapitre, nous utiliserons les notations suivantes:

- *un 2-intervalle entre les blocs X et Y* représente un 2-intervalle $D = (I, J)$ où I est un intervalle simple appartenant au bloc X et J un intervalle simple appartenant au bloc Y;

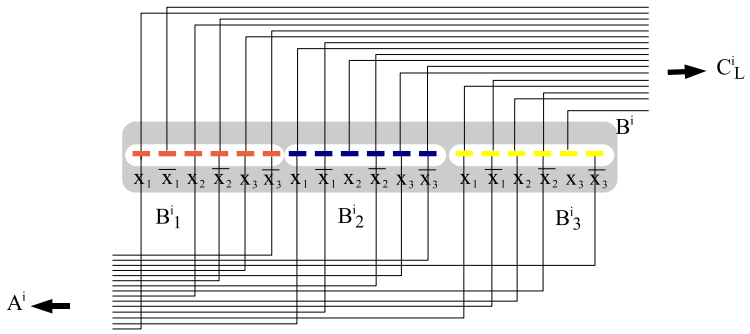


Figure 6.7 – Zoom sur le groupe B^i de la représentation de la clause $c_i = (\overline{x_1} \vee x_2 \vee x_3)$

- pour tout $1 \leq i \leq q$ et $\alpha \subseteq \{A^i, B^i, C^i_L, C^i_R, D^i, E^i, F^i\}$, $\mathcal{D}(\alpha)$ représente un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable entre les blocs des groupes appartenant à α.
  Par exemple, $\mathcal{D}(D^i, E^i, F^i)$ représente un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable entre les blocs D^i_1, D^i_2, E^i_1, E^i_2, E^i_3, F^i_1 et F^i_2;
- pour tout $1 \leq i \leq q$, $\mathcal{D}(c_i)$ représente un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable dans la représentation complète de la clause c_i.

Lemme 6.3. *Étant donnés deux groupes quelconques α et β tels que $\alpha, \beta \subseteq \{A^i, B^i, C^i_L, C^i_R, D^i, E^i, F^i\}$ pour $1 \leq i \leq q$, on a $|\mathcal{D}(\alpha)| + |\mathcal{D}(\beta)| \geq |\mathcal{D}(\alpha \bigcup \beta)|$.*

Preuve. L'union des ensembles α et β peut résulter dans l'un des cas suivant:

1. $\mathcal{D}(\alpha)$ et $\mathcal{D}(\beta)$ ont au moins un 2-intervalle en commum;
2. au moins un 2-intervalle de $\mathcal{D}(\alpha)$ et un 2-intervalle de $\mathcal{D}(\beta)$ ne sont pas disjoints;
3. au moins un 2-intervalle de $\mathcal{D}(\alpha)$ et un 2-intervalle de $\mathcal{D}(\beta)$ ne sont pas $\{<, \bowtie\}$-comparables.

Dans le cas (1), il est clair que le 2-intervalle dupliqué ne sera pas compté plus d'une fois dans $|\mathcal{D}(\alpha \bigcup \beta)|$. Dans le cas (2), seulement un des deux 2-intervalles non disjoints peut être dans $\mathcal{D}(\alpha \bigcup \beta)$. Dans le cas (3), seulement un des deux 2-intervalles non $\{<, \bowtie\}$-comparables peut être dans $\mathcal{D}(\alpha \bigcup \beta)$. Si aucun de ces cas n'apparait alors $\mathcal{D}(\alpha) \bigcup \mathcal{D}(\beta)$ est $\{<, \bowtie\}$-comparable, et donc, $|\mathcal{D}(\alpha)| + |\mathcal{D}(\beta)| = |\mathcal{D}(\alpha \bigcup \beta)|$. Par conséquent, $|\mathcal{D}(\alpha)| + |\mathcal{D}(\beta)| \geq |\mathcal{D}(\alpha \bigcup \beta)|$. □

Par construction, un 2-intervalle peut seulement exister entre deux blocs correspondant à une unique clause (*cf.* Figure 6.3). Par conséquent, la cardinalité maximale d'un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable de $\mathcal{D}$ (*i.e.* la représentation complète de la formule booléenne) peut être déduite de la cardinalité maximale de $\mathcal{D}(c_i)$ avec $c_i \in \mathcal{C}_q$, pour tout $1 \leq i \leq q$. Précisément, la cardinalité maximale d'un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable dans la représentation de toutes les clauses est inférieure ou égale à $q.max_{i \in [1,q]}|\mathcal{D}(c_i)|$ (nous rappelons que q est le nombre de clauses).

Nous calculons, en premier lieu, la cardinalité maximale d'un ensemble $\mathcal{D}(c_i)$ de 2-intervalles $\{<, \bowtie\}$-comparable entre des blocs correspondants à une unique clause c_i.

Lemme 6.4. $|\mathcal{D}(\alpha)| \leq 3n$ *pour* $\alpha = \{A^i, B^i, C^i_L, C^i_R\}$.

Preuve. Par la contrainte de disjonction, au plus un intervalle simple par bloc du groupe A^i peut être impliqué dans un 2-intervalle entre les blocs de A^i et ceux de B^i. Comme il y a n blocs dans le groupe A^i, nous obtenons $|\mathcal{D}(A^i, B^i)| \leq n$. De façon similaire, d'après la contrainte de disjonction, au plus un

intervalle simple par bloc de C_R^i peut être impliqué dans un 2-intervalle entre les blocs de B^i et ceux de C_R^i. Étant donné qu'il y a n blocs dans le groupe C_R^i, $|\mathcal{D}(B^i, C_R^i)| \leq n$. Par conséquent, d'après le Lemme 6.3, $|\mathcal{D}(A^i, B^i, C_R^i)| \leq |\mathcal{D}(A^i, B^i)| + |\mathcal{D}(B^i, C_R^i)| \leq 2n$.

De plus, au plus un intervalle simple par bloc de A^i peut être impliqué dans un 2-intervalle entre les blocs de A^i et ceux de C_L^i. En effet, deux 2-intervalles entre un bloc donné de A^i et un bloc de C_L^i sont $\{\sqsubset\}$-comparables. Étant donné qu'il y a n blocs dans le groupe A^i, $|\mathcal{D}(C_L^i, A^i)| \leq n$. Par conséquent, par le Lemme 6.3, $|\mathcal{D}(C_L^i, A^i, B^i, C_R^i)| \leq |\mathcal{D}(A^i, B^i, C_R^i)| + |\mathcal{D}(C_L^i, A^i)| \leq 3n$. □

Par la suite, pour tout $1 \leq i \leq q$ et $1 \leq j \leq 3$, $\theta(i,j)$ représentera l'ensemble de tous les intervalles simples appartenant aux blocs B_j^i et E_j^i. L'ensemble $\delta(i,j) \subseteq \theta(i,j)$ représentera un ensemble d'intervalles simples disjoints de $\theta(i,j)$ et $k(E,i,j)$ (*resp.* $k(B,i,j)$) représentera le nombre d'intervalles simples du bloc E_j^i (*resp.* B_j^i) présents dans $\delta(i,j)$. Nous rappelons que par construction, chaque intervalle simple du bloc E_j^i intersecte deux intervalles simples du bloc B_j^i (*cf.* Figures 6.4 et 6.8).

Observation 6.1.
- *(a) Si* $k(E,i,j) > 0$ *alors* $|\delta(i,j)| \leq 2n-1$*;*
- *(b) Si* $k(E,i,j) = 0$ *alors* $|\delta(i,j)| \leq 2n$.

Preuve. (a) Si $k(E,i,j) > 0$ alors au moins $k(E,i,j)+1$ intervalles simples du bloc B_j^i ne peuvent pas appartenir à $\delta(i,j)$.
Par conséquent, $k(B,i,j) \leq 2n - (k(E,i,j)+1)$. Et donc, $|\delta(i,j)| \leq k(B,i,j) + k(E,i,j) \leq 2n - (k(E,i,j)+1) + k(E,i,j) \leq 2n-1$.

(b) Si $k(E,i,j) = 0$ alors tous les intervalles simples du bloc B_j^i (i.e. $2n$) peuvent appartenir à $\delta(i,j)$.
Par conséquent, $k(B,i,j) \leq 2n$. Et donc, $|\delta(i,j)| \leq k(B,i,j) + k(E,i,j) \leq 2n$. □

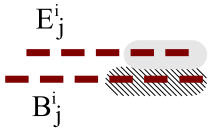


Figure 6.8 – Si deux intervalles simples du bloc E_j^i appartiennent à $\delta(i,j)$ alors au moins trois intervalles simples du bloc B_j^i ne peuvent appartenir à $\delta(i,j)$ et $|\delta(i,j)| \leq 2n-1$.

Lemme 6.5. *Si* $|\mathcal{D}(D^i, E^i, F^i)| > 4n-2$ *alors* $|\mathcal{D}(c_i)| < 7n-2$.

Preuve. Supposons que $|\mathcal{D}(D^i, E^i, F^i)| = 4n-2+\gamma$ avec $\gamma > 0$. Étant donné que chaque bloc du groupe E^i (*i.e.* E_1^i, E_2^i, E_3^i) est composé de $2n-1$ intervalles simples, il y a au moins un intervalle simple dans chaque bloc du groupe E^i impliqué dans un 2-intervalle de $\mathcal{D}(D^i, E^i, F^i)$.

Par conséquent, si on ne considére que les intervalles simples des groupes B^i et E^i, il y a au plus $6n-3$ (*i.e.* $3 \cdot (2n-1)$ par l'Observation 6.1 (a)) intervalles simples disjoints. Par construction, tout 2-intervalle de $\mathcal{D}(A^i, B^i, C_R^i, D^i, E^i, F^i)$ est composé d'un intervalle simple du groupe B^i ou du groupe E^i. Par conséquent, étant donné qu'il y a au plus $6n-3$ intervalles simples disjoints dans les groupes B^i et E^i, il y a également au plus $6n-3$ 2-intervalles dans $\mathcal{D}(A^i, B^i, C_R^i, D^i, E^i, F^i)$. Sachant que $|\mathcal{D}(C_L^i, A^i)| \leq n$ (*cf.* preuve du Lemme 6.4), d'après le Lemme 6.3 $|\mathcal{D}(C_L^i, A^i, B^i, C_R^i, D^i, E^i, F^i)| \leq |\mathcal{D}(C_L^i, A^i)| + |\mathcal{D}(A^i, B^i, C_R^i, D^i, E^i, F^i)| \leq 7n-3 < 7n-2$. Par conséquent, étant donné que $|\mathcal{D}(c_i)|$ ne peut excéder $max|\mathcal{D}(C_L^i, A^i, B^i, C_R^i, D^i, E^i, F^i)|$, si $|\mathcal{D}(D^i, E^i, F^i)| > 4n-2$ alors $|\mathcal{D}(c_i)| < 7n-2$. □

Lemme 6.6. *$|\mathcal{D}(c_i)| \leq 7n-2$. De plus, si $|\mathcal{D}(c_i)| = 7n-2$ alors $|\mathcal{D}(\{D^i, E^i, F^i\})| = 4n-2$ et $|\mathcal{D}(\{C_L^i, A^i, B^i, C_R^i\})| = 3n$.*

Preuve. Supposons, par contradiction, que $|\mathcal{D}(c_i)| > 7n-2$. Par le Lemme 6.3, $|\mathcal{D}(c_i)| \leq |\mathcal{D}(D^i, E^i, F^i)| + |\mathcal{D}(C_L^i, A^i, B^i, C_R^i)|$. Par conséquent, $|\mathcal{D}(D^i, E^i, F^i)| + |\mathcal{D}(C_L^i, A^i, B^i, C_R^i)| > 7n-2$. Sachant que, par le Lemme 6.4, $|\mathcal{D}(C_L^i, A^i, B^i, C_R^i)| \leq 3n$, nous obtenons $|\mathcal{D}(D^i, E^i, F^i)| > 4n-2$. Or, par le Lemme 6.5, si $|\mathcal{D}(D^i, E^i, F^i)| > 4n-2$ alors $|\mathcal{D}(c_i)| < 7n-2$, une contradiction. Par conséquent, nous obtenons $|\mathcal{D}(c_i)| \leq 7n-2$.

Supposons, dorénavant, que $|\mathcal{D}(c_i)| = 7n-2$. Alors, par le Lemme 6.5, $|\mathcal{D}(D^i, E^i, F^i)| \leq 4n-2$. Nous obtenons $|\mathcal{D}(C_L^i, A^i, B^i, C_R^i)| \geq 3n$. Or, par le Lemme 6.4, $|\mathcal{D}(C_L^i, A^i, B^i, C_R^i)| \leq 3n$. Par conséquent, $|\mathcal{D}(C_L^i, A^i, B^i, C_R^i)| = 3n$ et donc $|\mathcal{D}(D^i, E^i, F^i)| = 4n-2$. □

Lemme 6.7. *Si $|\mathcal{D}(c_i)| = 7n-2$ alors l'ensemble $\mathcal{D}(D^i, E^i, F^i)$ contient les 2-intervalles construits à partir des tous les intervalles simples d'exactement deux blocs du groupe E^i (i.e. (E_1^i, E_2^i), (E_1^i, E_3^i) ou (E_2^i, E_3^i)).*

Preuve. Étant donné que $|\mathcal{D}(c_i)| = 7n-2$, par le Lemme 6.6, nous savons que $|\mathcal{D}(C_L^i, A^i, B^i, C_R^i)| = 3n$. De plus, nous savons que $|\mathcal{D}(C_L^i, A^i)| \leq n$ (*cf.* preuve du Lemme 6.4). Alors, par le Lemme 6.3, nous devons avoir $|\mathcal{D}(A^i, B^i, C_R^i)| \geq 2n$. Étant donné que $|\mathcal{D}(A^i, B^i, C_R^i)| \leq 2n$ (*cf.* preuve du Lemme 6.4), nous obtenons $|\mathcal{D}(A^i, B^i, C_R^i)| = 2n$.

Étant donné que $|\mathcal{D}(c_i)| = 7n-2$, par le Lemme 6.6, nous savons que $|\mathcal{D}(D^i, E^i, F^i)| = 4n-2$. De plus, par construction, chaque 2-intervalle de $\mathcal{D}(D^i, E^i, F^i)$ est construit avec un intervalle simple de E^i. Par conséquent, $\sum_{j=1}^{3}(k(E,i,j)) = 4n-2$.

Supposons, par contradiction, que pour tout $1 \leq j \leq 3$, $k(E,i,j) > 0$. Par l'Observation 6.1, pour tout $1 \leq j \leq 3$ nous obtenons $k(B,i,j) \leq 2n - (k(E,i,j)+1)$. Par conséquent, $\sum_{j=1}^{3} k(B,i,j) \leq \sum_{j=1}^{3} 2n-(k(E,i,j)+1) \leq 6n-3-\sum_{j=1}^{3} k(E,i,j)$. Étant donné que $\sum_{j=1}^{3} k(E,i,j) = 4n-2$, nous en concluons que $\sum_{j=1}^{3} k(B,i,j) \leq 2n-1$. Or, par construction, chaque 2-intervalle de $\mathcal{D}(A^i, B^i, C_R^i)$ est construit avec un intervalle simple de B^i. Par conséquent, $|\mathcal{D}(A^i, B^i, C_R^i)| \leq 2n-1$, une contradiction.

Par conséquent au moins un parmi $k(E,i,1)$, $k(E,i,2)$ et $k(E,i,3)$ est égal à zéro. Donc, $\mathcal{D}(D^i, E^i, F^i)$ contient des 2-intervalles construits avec tous les intervalles simples d'exactement deux blocs du groupe E^i (*i.e.* (E_1^i, E_2^i), (E_1^i, E_3^i) ou (E_2^i, E_3^i)). □

Corollaire 6.1. *Si $|\mathcal{D}(c_i)| = 7n-2$ alors l'ensemble $\mathcal{D}(A^i, B^i, C_R^i)$ contient tous les intervalles simples d'un unique bloc du groupe B^i (i.e. B_1^i, B_2^i ou B_3^i).*

Preuve. Par le Lemme 6.6, si $|\mathcal{D}(c_i)| = 7n-2$ alors $|\mathcal{D}(C_L^i, A^i, B^i, C_R^i)| = 3n$. De plus, par construction, chaque 2-intervalle de $\mathcal{D}(A^i, B^i, C_R^i)$ est construit avec un intervalle simple de B^i. Sachant que $|\mathcal{D}(A^i, B^i, C_R^i)| = 2n$ (*cf.* preuve du Lemme 6.7), $\sum_{j=1}^{3}(k(B,i,j)) = 2n$. Par le Lemme 6.7, si $|\mathcal{D}(c_i)| = 7n-2$ alors $\mathcal{D}(D^i, E^i, F^i)$ contient des 2-intervalles construits avec tous les intervalles simples d'exactement deux blocs E_s^i et E_t^i du groupe E^i, avec $1 \leq s,t \leq 3$. Par l'Observation 6.1, $\mathcal{D}(A^i, B^i, C_R^i)$ contient des 2-intervalles construits avec tous les intervalles simples d'exactement un bloc B_u^i du groupe B^i avec $1 \leq u \leq 3$, $u \neq s$ et $u \neq t$. □

Lemme 6.8. *Si $|\mathcal{D}(c_i)| = 7n-2$*

1. *si les $2n$ 2-intervalles de $\mathcal{D}(A^i, B^i, C_R^i)$ contiennent les 2-intervalles construits avec tous les intervalles simples de B_1^i alors $\mathcal{D}(D^i, E^i, F^i)$ est l'ensemble de tous les 2-intervalles entre les blocs E_2^i, E_3^i, F_1^i et F_2^i;*

2. *si les $2n$ 2-intervalles de $\mathcal{D}(A^i, B^i, C^i_R)$ contiennent les 2-intervalles construits avec tous les intervalles simples de B^i_2 alors $\mathcal{D}(D^i, E^i, F^i)$ est l'ensemble de tous les 2-intervalles entre les blocs E^i_1, E^i_3, D^i_1 et F^i_2;*
3. *si les $2n$ 2-intervalles de $\mathcal{D}(A^i, B^i, C^i_R)$ contiennent les 2-intervalles construits avec tous les intervalles simples de B^i_3 alors $\mathcal{D}(D^i, E^i, F^i)$ est l'ensemble de tous les 2-intervalles entre les blocs E^i_1, E^i_2, D^i_1 et D^i_2.*

Preuve. (1) Par le Lemme 6.6, si $|\mathcal{D}(c_i)| = 7n - 2$ alors $|\mathcal{D}(D^i, E^i, F^i)| = 4n - 2$. Par le Corollaire 6.1, le Lemme 6.7 et la contrainte de disjonction, si les $2n$ 2-intervalles de $\mathcal{D}(A^i, B^i, C^i_R)$ contiennent les 2-intervalles construits avec tous les intervalles simples de B^i_1 alors $\mathcal{D}(D^i, E^i, F^i)$ contient les 2-intervalles construits avec tous les intervalles simples de E^i_2 et E^i_3. Par conséquent, $\mathcal{D}(D^i, E^i, F^i)$ est composé des $2n - 1$ 2-intervalles entre les blocs E^i_3 et F^i_2.

De plus, tout 2-intervalle entre les blocs E^i_2 et D^i_2 est $\{\sqsubset\}$-comparable à tout 2-intervalle entre les blocs A^i et B^i_1. Par conséquent, l'ensemble $\mathcal{D}(D^i, E^i, F^i)$ de $4n-2$ 2-intervalles est également composé des $2n - 1$ 2-intervalles entre les blocs E^i_2 et F^i_1.

(2) De façon similaire à (1), si les $2n$ 2-intervalles de $\mathcal{D}(A^i, B^i, C^i_R)$ contiennent les 2-intervalles construits avec tous les intervalles simples de B^i_2 alors $\mathcal{D}(D^i, E^i, F^i)$ contient les 2-intervalles construits avec tous les intervalles simples de E^i_1 et E^i_3. Par conséquent, $\mathcal{D}(D^i, E^i, F^i)$ est composé des $2n - 1$ 2-intervalles entre les blocs E^i_1 et D^i_1 et des $2n - 1$ 2-intervalles entre les blocs E^i_3 et F^i_2.

(3) De façon similaire à (1) et (2), si les $2n$ 2-intervalles de $\mathcal{D}(A^i, B^i, C^i_R)$ contiennent les 2-intervalles construits avec tous les intervalles simples de B^i_3 alors $\mathcal{D}(D^i, E^i, F^i)$ contient les 2-intervalles construits avec tous les intervalles simples de E^i_1 et E^i_2. Par conséquent, $\mathcal{D}(D^i, E^i, F^i)$ est composé des $2n - 1$ 2-intervalles entre les blocs E^i_1 et D^i_1.

De plus, tout 2-intervalle entre les blocs E^i_2 et F^i_1 est $\{\sqsubset\}$-comparable à tout 2-intervalle entre les blocs B^i_3 et C^i_R. Par conséquent, $\mathcal{D}(D^i, E^i, F^i)$ est également composé des $2n-1$ 2-intervalles entre les blocs E^i_2 et D^i_2. □

Par la suite, pour tout $1 \leq m \leq n$, $x_m(U, V)$ (*resp.* $\overline{x_m}(U, V)$) représente le 2-intervalle composé des deux intervalles simples représentant x_m (*resp.* $\overline{x_m}$) dans les blocs U et V.

Observation 6.2. *Supposons que $|\mathcal{D}(c_i)| = 7n - 2$ et que les $2n$ 2-intervalles de $\mathcal{D}(c_i)$ contiennent les 2-intervalles construits avec tous les intervalles simples de B^i_j pour un $1 \leq j \leq 3$ donné.*

- *Si $x_m(C^i_L, A^i) \in \mathcal{D}(c_i)$ alors $x_m(A^i, B^i_j) \in \mathcal{D}(c_i)$.*
- *Si $\overline{x_m}(C^i_L, A^i) \in \mathcal{D}(c_i)$ alors $\overline{x_m}(A^i, B^i_j) \in \mathcal{D}(c_i)$.*

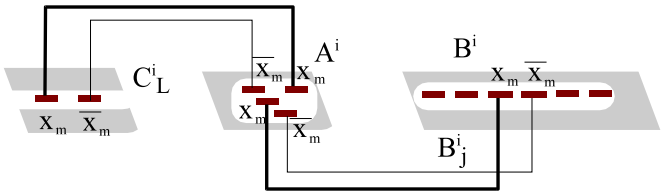


Figure 6.9 – $x_m(C^i_L, A^i) \in \mathcal{D}(c_i)$ implique $x_m(A^i, B^i_j) \in \mathcal{D}(c_i)$.

Preuve. Une illustration de l'Observation 6.2 est donnée en Figure 6.9. Sachant que $|\mathcal{D}(c_i)| = 7n-2$, par le Lemme 6.6, $|\mathcal{D}(C^i_L, A^i, B^i, C^i_R)| = 3n$. De plus, nous avons prouvé (*cf.* preuve du Lemme 6.4) que

$|\mathcal{D}(A^i, B^i)| \leq n$, $|\mathcal{D}(B^i, C^i_R)| \leq n$, et $|\mathcal{D}(C^i_L, A^i)| \leq n$. Par le Lemme 6.3, $|\mathcal{D}(A^i, B^i)|+|\mathcal{D}(B^i, C^i_R)|+|\mathcal{D}(C^i_L, A^i)| \geq |\mathcal{D}(C^i_L, A^i, B^i, C^i_R)|$. Par conséquent, $|\mathcal{D}(A^i, B^i)| = |\mathcal{D}(B^i, C^i_R)| = |\mathcal{D}(C^i_L, A^i)| = n$.

De plus, nous avons prouvé que si $|\mathcal{D}(C^i_L, A^i)| = n$ alors un intervalle simple par bloc de A^i est impliqué dans un 2-intervalle entre C^i_L et A^i (*cf.* preuve du Lemme 6.4). Considérons le $m^{ème}$ bloc de A^i. Alors, par la contrainte de $\{<, \bowtie\}$-comparabilité, soit $x_m(C^i_L, A^i) \in \mathcal{D}(c_i)$, soit $\overline{x_m}(C^i_L, A^i) \in \mathcal{D}(c_i)$.

De façon similaire, nous avons prouvé que si $|\mathcal{D}(A^i, B^i)| = n$ alors un intervalle simple par bloc de A^i est impliqué dans un 2-intervalle entre A^i et B^i (*cf.* preuve du Lemme 6.4). Considérons le $m^{ème}$ bloc de A^i. Nous avons précisé que, par construction, les intervalles simples de ce bloc représentent, de gauche à droite, $\overline{x_m}, x_m, \overline{x_m}, x_m$ (*cf.* Figures 6.5 et 6.9). Par conséquent, soit $x_m(A^i, B^i_j) \in \mathcal{D}(c_i)$, soit $\overline{x_m}(A^i, B^i_j) \in \mathcal{D}(c_i)$.

De plus, par la contrainte de disjonction et le positionnement des intervalles simples de chaque bloc de A^i (*cf.* Figures 6.5 et 6.9), si $x_m(C^i_L, A^i) \in \mathcal{D}(c_i)$ alors $x_m(A^i, B^i_j) \in \mathcal{D}(c_i)$. De façon similaire, si $\overline{x_m}(C^i_L, A^i) \in \mathcal{D}(c_i)$ alors $\overline{x_m}(A^i, B^i_j) \in \mathcal{D}(c_i)$. □

Observation 6.3. *Supposons que* $|\mathcal{D}(c_i)| = 7n - 2$ *et que les* $2n$ *2-intervalles de* $\mathcal{D}(c_i)$ *contiennent les 2-intervalles construits avec tous les intervalles simples de* B^i_j *pour un* $1 \leq j \leq 3$ *donné.*

- *Si* $x_m(A^i, B^i_j) \in \mathcal{D}(c_i)$ *alors* $\overline{x_m}(B^i_j, C^i_R) \in \mathcal{D}(c_i)$.
- *Si* $\overline{x_m}(A^i, B^i_j) \in \mathcal{D}(c_i)$ *alors* $x_m(B^i_j, C^i_R) \in \mathcal{D}(c_i)$.

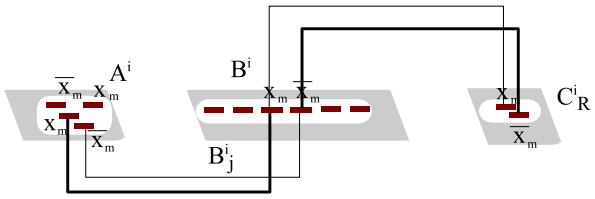


Figure 6.10 – $x_m(A^i, B^i_j) \in \mathcal{D}(c_i)$ implique $\overline{x_m}(B^i_j, C^i_R) \in \mathcal{D}(c_i)$.

Preuve. Une illustration de l'Observation 6.3 est donnée en Figure 6.10. Supposons que $x_m(A^i, B^i_{j_0}) \in \mathcal{D}(c_i)$ pour un $1 \leq j_0 \leq 3$ donné. Par le Corollaire 6.1, étant donné que $|\mathcal{D}(c_i)| = 7n - 2$, l'ensemble $\mathcal{D}(A^i, B^i, C^i_R)$ contient tous les intervalles simples d'un unique bloc B^i_j du groupe B^i. Par conséquent, et d'après l'hypothèse, l'ensemble $\mathcal{D}(A^i, B^i, C^i_R)$ contient tous les intervalles simples du bloc $B^i_{j_0}$.

Nous avons prouvé (*cf.* preuve de l'Observation 6.2) que soit $x_m(A^i, B^i_{j_0}) \in \mathcal{D}(c_i)$, soit $\overline{x_m}(A^i, B^i_{j_0}) \in \mathcal{D}(c_i)$ pour un $1 \leq j_0 \leq 3$ donné. Par la contrainte de disjonction, étant donné que $x_m(A^i, B^i_{j_0}) \in \mathcal{D}(c_i)$, nous obtenons $x_m(B^i_{j_0}, C^i_R) \notin \mathcal{D}(c_i)$. De plus, étant donné que l'ensemble $\mathcal{D}(A^i, B^i, C^i_R)$ contient tous les intervalles simples du bloc $B^i_{j_0}$, $\overline{x_m}(B^i_{j_0}, C^i_R) \in \mathcal{D}(c_i)$. De façon similaire, si $\overline{x_m}(A^i, B^i_{j_0}) \in \mathcal{D}(c_i)$ pour un $1 \leq j_0 \leq 3$ donné alors $x_m(B^i_{j_0}, C^i_R) \in \mathcal{D}(c_i)$. □

Observation 6.4. *Supposons que* $|\mathcal{D}(c_i)| = |\mathcal{D}(c_{i+1})| = 7n - 2$ *et que les* $2n$ *2-intervalles de* $\mathcal{D}(c_i)$ *contiennent les 2-intervalles construits avec tous les intervalles simples de* B^i_j *pour un* $1 \leq j \leq 3$ *donné.*

- *Si* $x_m(B^i_j, C^i_R) \in \mathcal{D}(c_i)$ *alors* $\overline{x_m}(C^{i+1}_L, A^{i+1}) \in \mathcal{D}(c_{i+1})$.
- *Si* $\overline{x_m}(B^i_j, C^i_R) \in \mathcal{D}(c_i)$ *alors* $x_m(C^{i+1}_L, A^{i+1}) \in \mathcal{D}(c_{i+1})$.

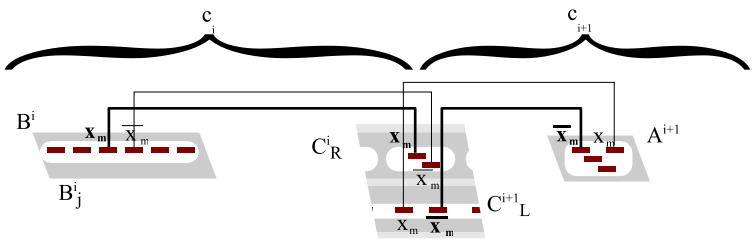


Figure 6.11 – $x_m(B^i_j, C^i_R) \in \mathcal{D}(c_i)$ implique $\overline{x_m}(C^{i+1}_L, A^{i+1}) \in \mathcal{D}(c_{i+1})$

Preuve. Une illustration de l'Observation 6.4 est donnée en Figure 6.11. Si $|\mathcal{D}(c_{i+1})| = 7n-2$ alors $|\mathcal{D}(C^{i+1}_L, A^{i+1})| = n$ (*cf.* preuve de l'Observation 6.2). Par la contrainte de $\{<, \bowtie\}$-comparabilité, soit $x_m(C^{i+1}_L, A^{i+1}) \in \mathcal{D}(c_{i+1})$, soit $\overline{x_m}(C^{i+1}_L, A^{i+1}) \in \mathcal{D}(c_{i+1})$ (*cf.* preuve de l'Observation 6.2).

Par le positionnement des blocs C^i_R et C^{i+1}_L, si $|\mathcal{D}(c_i)| = |\mathcal{D}(c_{i+1})| = 7n-2$ et $x_m(B^i_j, C^i_R) \in \mathcal{D}(c_i)$ alors $\overline{x_m}(C^{i+1}_L, A^{i+1}) \in \mathcal{D}(c_{i+1})$. De façon similaire, si $|\mathcal{D}(c_i)| = |\mathcal{D}(c_{i+1})| = 7n-2$ et $\overline{x_m}(B^i_j, C^i_R) \in \mathcal{D}(c_i)$ alors $x_m(C^{i+1}_L, A^{i+1}) \in \mathcal{D}(c_{i+1})$. □

Les Lemmes 6.3 à 6.8 et les Observations 6.2 à 6.4 nous fournissent l'ensemble des résultats intermédiaires dont nous avons besoin pour prouver la **NP**-complétude du sous-problème 2-IP(UNITARY,$\{<, \bowtie\}$).

Proposition 6.3. *Étant donnée une instance du problème* EXACT-3-CNF-SAT *avec n variables et q clauses, il existe un ensemble de valeurs pour les n variables satisfaisant les q clauses si et seulement si il y a un sous-ensemble $\{<, \bowtie\}$-comparable $\mathcal{D}' \subseteq \mathcal{D}$ tel que $|\mathcal{D}'| \geq (7n-2)q$.*

Preuve. ($\Rightarrow$)
Supposons que nous avons un ensemble de valeurs $\mathcal{V}$ pour les n variables satisfaisant les q clauses. Par définition, pour chaque clause il y a au moins un littéral la satisfaisant. Nous recherchons un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable $\mathcal{D}'$ dans la représentation de la formule booléenne tel que la cardinalité de $\mathcal{D}'$ soit supérieure ou égale à $(7n-2) \cdot q$.

Par le Lemme 6.6, pour toute clause c_i, $|\mathcal{D}(c_i)| \leq 7n-2$. Donc, $|\mathcal{D}'| \leq (7n-2) \cdot q$. Par conséquent, la seule solution à notre problème est l'ensemble $\mathcal{D}'$ tel que $|\mathcal{D}'| = (7n-2) \cdot q$. La formule booléenne étant composée de q clauses, chaque sous-ensemble $\mathcal{D}'(c_i)$ de $\mathcal{D}'$ pour toute clause c_i, $1 \leq i \leq q$, doit satisfaire $|\mathcal{D}'(c_i)| = 7n-2$.

Par la suite, j_i représentera le plus petit indice des littéraux de c_i (*i.e.* 1, 2 ou 3) qui, par leurs valeurs, satisfont c_i. Pour tout $1 \leq i \leq q$, nous définissons $\mathcal{D}'(c_i)$ comme suit.

Pour chaque variable x_m avec $1 \leq m \leq n$:

1. si $x_m = Vrai$ alors $\overline{x_m}(C^i_L, A^i)$, $\overline{x_m}(A^i, B^i_{j_i})$ et $x_m(B^i_{j_i}, C^i_R)$ appartiennent à $\mathcal{D}'(c_i)$;
2. si $x_m = Faux$ alors $x_m(C^i_L, A^i)$, $x_m(A^i, B^i_{j_i})$ et $\overline{x_m}(B^i_{j_i}, C^i_R)$ appartiennent à $\mathcal{D}'(c_i)$;
3. de plus, pour tout $1 \leq j_i \leq 3$ donné:

   si $j_i = 1$ alors $\mathcal{D}'(c_i)$ est aussi composé de l'ensemble de tous les 2-intervalles entre les blocs E^i_2, E^i_3, F^i_1 et F^i_2;

   si $j_i = 2$ alors $\mathcal{D}'(c_i)$ est aussi composé de l'ensemble de tous les 2-intervalles entre les blocs E^i_1, E^i_3, D^i_1 et F^i_2;

si $j_i = 3$ alors $\mathcal{D}'(c_i)$ est aussi composé de l'ensemble de tous les 2-intervalles entre les blocs E^i_1, E^i_2, D^i_1 et D^i_2.

Un exemple de sous-ensemble $\mathcal{D}'(c_i)$ avec $c_i = (\overline{x_1} \vee x_2 \vee x_3)$ et tel que $x_1 = x_2 = x_3 = Vrai$ est donné en Figure 6.12.

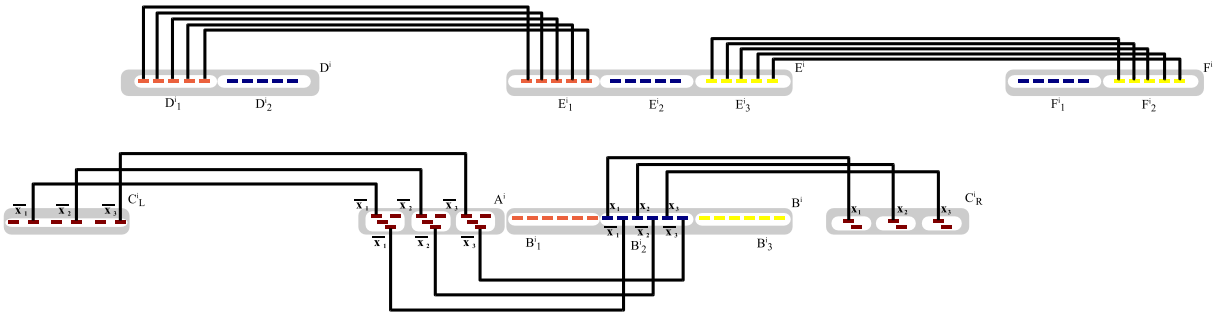


Figure 6.12 – $\mathcal{D}'(c_i)$ avec $c_i = (\overline{x_1} \vee x_2 \vee x_3)$ et $x_1 = x_2 = x_3 = Vrai$

Nous prouverons, dans un premier temps, que pour tout $1 \leq i \leq q$, $\mathcal{D}'(c_i)$ est un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable. Puis, nous prouverons que $\mathcal{D}' = \bigcup_1^q \mathcal{D}'(c_i)$ est un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable tel que $|\mathcal{D}'| = (7n-2)q$.

D'après la définition de $\mathcal{D}'(c_i)$, il est facile de vérifier que $|\mathcal{D}'(c_i)| = 7n-2$. En effet, par (1) ou (2), trois 2-intervalles ont été ajoutés à $\mathcal{D}'(c_i)$ pour chaque variable x_m avec $1 \leq m \leq n$. De plus, par (3), (4) ou (5), pour tout $1 \leq j_i \leq 3$ donné, un ensemble de $4n-2$ 2-intervalles a été ajouté à $\mathcal{D}'(c_i)$. Soit en tout $7n-2$ 2-intervalles.

Pour tout $1 \leq i \leq q$, $\mathcal{D}'(c_i)$ est un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable si et seulement si il n'y a pas de problème d'inclusion ou de disjonction dans $\mathcal{D}'(c_i)$. Tout d'abord, nous prouverons qu'étant donné un $1 \leq j_i \leq 3$, $\mathcal{D}'(C^i_L, A^i, B^i_{j_i}, C^i_R)$ est un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable. Puis, nous prouverons qu'étant donné un $1 \leq j_i \leq 3$, $\mathcal{D}'(D^i, E^i, F^i)$ est également un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable. Finalement, nous prouverons que $\mathcal{D}'(c_i)$, qui est l'union de ces deux ensembles, est un ensemble de 2-intervalles $\{<, \bowtie\}$-comparable.

Considérons uniquement les 2-intervalles de $\mathcal{D}'(C^i_L, A^i, B^i_{j_i}, C^i_R)$: par construction, une inclusion ne peut intervenir qu'entre deux 2-intervalles construits avec des intervalles simples d'exactement deux groupes. Pour tout $1 \leq j_i \leq 3$, par construction, les 2-intervalles entre A^i et $B^i_{j_i}$ (*resp.* $B^i_{j_i}$ et C^i_R) se croisent deux à deux. Par conséquent, une inclusion peut uniquement intervenir lorsque deux intervalles simples d'un même bloc de A^i sont tous deux dans un 2-intervalle entre C^i_L et A^i dans $\mathcal{D}'(C^i_L, A^i, B^i_{j_i}, C^i_R)$.

Clairement, pour chaque variable x_m, soit $\overline{x_m}(C^i_L, A^i) \in \mathcal{D}'(c_i)$, soit $x_m(C^i_L, A^i) \in \mathcal{D}'(c_i)$. Donc, un unique intervalle simple par bloc de A^i est impliqué dans un 2-intervalle entre C^i_L et A^i. Par conséquent, il ne peut y avoir d'inclusion dans $\mathcal{D}'(C^i_L, A^i, B^i_{j_i}, C^i_R)$.

D'après la définition de $\mathcal{D}'(c_i)$ et par la construction de la représentation d'une clause, on peut vérifier qu'il ne peut exister deux 2-intervalles non disjoints dans $\mathcal{D}'(C^i_L, A^i, B^i_{j_i}, C^i_R)$ (voir par exemple Figure 6.12). Par conséquent, $\mathcal{D}'(C^i_L, A^i, B^i_{j_i}, C^i_R)$ est un ensemble de $3n$ 2-intervalles $\{<, \bowtie\}$-comparable.

Considérons uniquement les 2-intervalles de $\mathcal{D}'(D^i, E^i, F^i)$: par construction, il ne peut pas y avoir de problème d'inclusion dans $\mathcal{D}'(D^i, E^i, F^i)$. De plus, un problème de disjonction ne peut intervenir que lorsqu'un intervalle simple du bloc E^i_2 est impliqué dans deux 2-intervalles dans $\mathcal{D}'(D^i, E^i, F^i)$. D'après la définition de $\mathcal{D}'(c_i)$, cette situation ne se présente jamais. Par conséquent, $\mathcal{D}'(D^i, E^i, F^i)$ est un ensemble de $4n-2$ 2-intervalles $\{<, \bowtie\}$-comparable.

Nous considérons, dorénavant, les 2-intervalles de $\mathcal{D}'(c_i)$. Nous venons de prouver que pour tout $1 \leq j_i \leq 3$, $\mathcal{D}'(C_L^i, A^i, B_{j_i}^i, C_R^i)$ et $\mathcal{D}'(D^i, E^i, F^i)$ sont des ensembles de 2-intervalles $\{<, \bowtie\}$-comparables. Par conséquent, il nous reste à vérifier que leur assemblage ne produit pas de problèmes d'inclusion ou de disjonction. Pour ce faire, nous allons examiner les trois cas suivants:

1. $j_i = 1$. $\mathcal{D}'(c_i)$ contient n 2-intervalles entre C_L^i et A^i, n 2-intervalles entre A^i et B_1^i, n 2-intervalles entre B_1^i et C_R^i, $2n-1$ 2-intervalles entre E_2^i et F_1^i et $2n-1$ 2-intervalles entre E_3^i et F_2^i.

   Par construction, tous les 2-intervalles sont disjoints. De plus, tout 2-intervalle entre E_2^i et F_1^i (*resp.* E_3^i et F_2^i) croise tout 2-intervalle entre B_1^i et C_R^i (*cf.* Figure 6.13). Donc, il n'y a pas de problème d'inclusion dans $\mathcal{D}'(c_i)$. Par conséquent, $\mathcal{D}'(c_i)$ est un ensemble de $7n-2$ 2-intervalles $\{<, \bowtie\}$-comparable dans le cas (1).

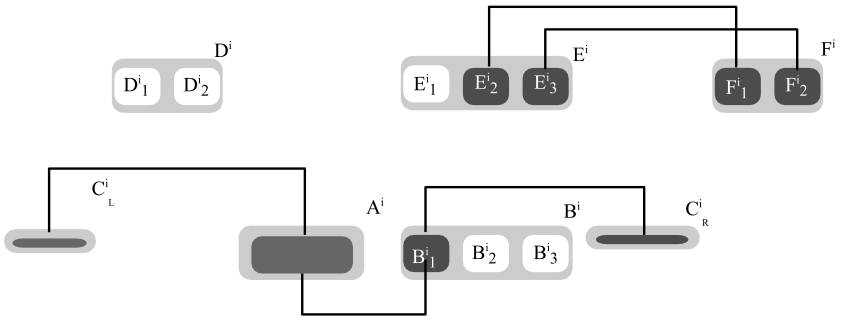


Figure 6.13 – Illustration du cas (1). Les lignes en gras représentent des ensembles de 2-intervalles entre des groupes.

2. $j_i = 2$. $\mathcal{D}'(c_i)$ contient n 2-intervalles entre C_L^i et A^i, n 2-intervalles entre A^i et B_2^i, n 2-intervalles entre B_2^i et C_R^i, $2n-1$ 2-intervalles entre D_1^i et E_1^i et $2n-1$ 2-intervalles entre E_3^i et F_2^i.

   Par construction, tous les 2-intervalles sont disjoints. De plus, tout 2-intervalle entre D_1^i et E_1^i croise tout 2-intervalle entre C_L^i et A^i (*resp.* A^i et B_2^i). De plus, tout 2-intervalle entre E_3^i et F_2^i croise tout 2-intervalle entre B_2^i et C_R^i (*cf.* Figure 6.14). Par conséquent, $\mathcal{D}'(c_i)$ est un ensemble de $7n-2$ 2-intervalles $\{<, \bowtie\}$-comparable dans le cas (2).

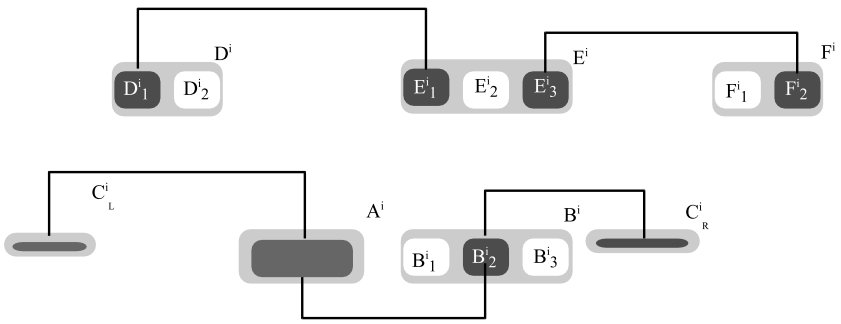


Figure 6.14 – Illustration du cas (2). Les lignes en gras représentent des ensembles de 2-intervalles entre des groupes.

3. $j_i = 3$. $\mathcal{D}'(c_i)$ contient n 2-intervalles entre C_L^i et A^i, n 2-intervalles entre A^i et B_3^i, n 2-intervalles entre B_3^i et C_R^i, $2n-1$ 2-intervalles entre D_1^i et E_1^i et $2n-1$ 2-intervalles entre D_2^i et E_2^i.

Par construction, tous les 2-intervalles sont disjoints. De plus, tout 2-intervalle entre D_1^i et E_1^i (*resp.* D_2^i et E_2^i) croise tout 2-intervalle entre C_L^i et A^i. De plus, tout 2-intervalle entre D_1^i et E_1^i (*resp.* D_2^i et E_2^i) croise tout 2-intervalle entre A^i et B_3^i (*cf.* Figure 6.15). Par conséquent, $\mathcal{D}'(c_i)$ est un ensemble de $7n-2$ 2-intervalles $\{<,\bowtie\}$-comparable dans le cas (3).

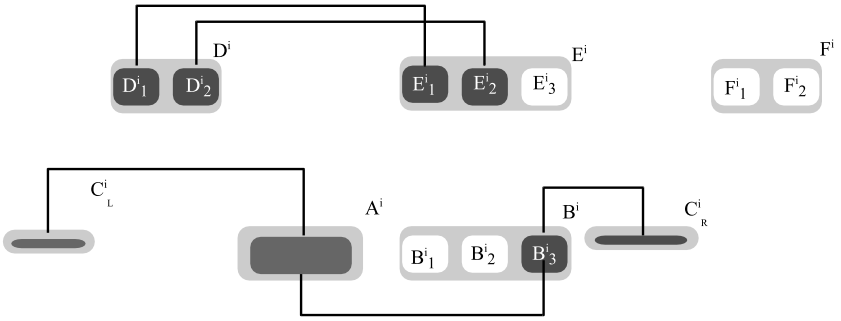


Figure 6.15 – Illustration du cas (3). Les lignes en gras représentent des ensembles de 2-intervalles entre des groupes.

Nous venons de prouver qu'il est possible de trouver un sous-ensemble $\{<,\bowtie\}$-comparable $\mathcal{D}(c_i)$ de $\mathcal{D}'$ pour chaque clause c_i tel que $|\mathcal{D}(c_i)| = 7n-2$. Finalement, il nous reste à vérifier que $\mathcal{D}' = \bigcup_1^q \mathcal{D}'(c_i)$ est $\{<,\bowtie\}$-comparable.

Par construction, il ne peut pas y avoir de problèmes d'inclusion entre deux 2-intervalles de différentes clauses. Il nous reste donc à prouver que, pour tout $1 \leq i < q$, le positionnement des blocs C_R^i et C_L^{i+1} (*cf.* Figure 6.6) n'implique pas de 2-intervalles non disjoints.

Par le positionnement des blocs C_L^{i+1} et C_R^i, un problème de disjonction ne peut intervenir qu'entre l'intervalle simple représentant x_m (*resp.* $\overline{x_m}$) dans C_R^i et l'intervalle simple représentant x_m (*resp.* $\overline{x_m}$) dans C_L^{i+1} pour un $1 \leq m \leq n$ donné.

D'après la définition de $\mathcal{D}'(c_i)$, si $x_m = Vrai$ alors pour tout $1 \leq i \leq q$, $\overline{x_m}(C_L^i, A^i)$ et $x_m(B_{j_i}^i, C_R^i)$ sont dans $\mathcal{D}'(c_i)$. Par conséquent, si $x_m = Vrai$ alors $x_m(B_{j_i}^i, C_R^i) \in \mathcal{D}'(c_i)$ et $\overline{x_m}(C_L^{i+1}, A^{i+1}) \in \mathcal{D}'(c_{i+1})$. Or, nous savons que, pour tout $1 \leq j_i \leq 3$, $x_m(B_{j_i}^i, C_R^i)$ et $\overline{x_m}(C_L^{i+1}, A^{i+1})$ sont disjoints (*cf.* Figures 6.5 et 6.6).

D'après la définition de $\mathcal{D}'(c_i)$, si $x_m = Faux$ alors pour tout $1 \leq i \leq q$, $x_m(C_L^i, A^i)$ et $\overline{x_m}(B_{j_i}^i, C_R^i)$ sont dans $\mathcal{D}'(c_i)$. Par conséquent, si $x_m = Faux$ alors $\overline{x_m}(B_{j_i}^i, C_R^i) \in \mathcal{D}'(c_i)$ et $x_m(C_L^{i+1}, A^{i+1}) \in \mathcal{D}'(c_{i+1})$. Or, nous savons que, pour tout $1 \leq j_i \leq 3$, $\overline{x_m}(B_{j_i}^i, C_R^i)$ et $x_m(C_L^{i+1}, A^{i+1})$ sont disjoints (*cf.* Figures 6.5 et 6.6).

Par conséquent, il ne peut exister, pour un $1 \leq i < q$ donné, de problème de disjonction causé par le positionnement des blocs C_L^{i+1} et C_R^i dans $\mathcal{D}'$. Donc, il existe un ensemble de 2-intervalles $\{<,\bowtie\}$-comparable, dans la représentation de la formule booléenne, de cardinalité égale à $q \cdot (7n-2)$.

$(\Leftarrow)$

Supposons que nous avons un sous-ensemble $\{<,\bowtie\}$-comparable $\mathcal{D}' \subseteq \mathcal{D}$ de cardinalité $q \cdot (7n-2)$. Par le Lemme 6.6, $\mathcal{D}'$ est composé d'un sous-ensemble $\mathcal{D}'(c_i)$ d'au plus $7n-2$ 2-intervalles $\{<,\bowtie\}$-comparables pour chaque clause c_i avec $1 \leq i \leq q$. Par conséquent, pour tout $1 \leq i \leq q$, $|\mathcal{D}'(c_i)| = 7n-2$. Nous définissons l'ensemble des valeurs $\mathcal{V}$ des n variables comme suit.

Pour tout $1 \leq m \leq n$:

- si $\overline{x_m}(C_L^1, A^1) \in \mathcal{D}'$ alors la valeur de la variable x_m est $Vrai$;

- si $x_m(C_L^1, A^1) \in \mathcal{D}'$ alors la valeur de la variable x_m est $Faux$.

Nous avons prouvé (*cf.* preuve de l'Observation 6.2) que pour tout $1 \leq i \leq q$ si $|\mathcal{D}(c_i)| = 7n-2$ alors $|\mathcal{D}(C_L^i, A^i)| = n$. Par conséquent, étant donné que $|\mathcal{D}'(c_1)| = 7n-2$, $\mathcal{D}'(c_1)$ est composé de n 2-intervalles entre les blocs de C_L^1 et A^1. De plus, nous avons prouvé (*cf.* preuve de l'Observation 6.2) que pour tout $1 \leq i \leq q$ si $|\mathcal{D}(c_i)| = 7n-2$ alors soit $x_m(C_L^i, A^i) \in \mathcal{D}(c_i)$, soit $\overline{x_m}(C_L^i, A^i) \in \mathcal{D}(c_i)$. Donc, soit $x_m(C_L^1, A^1) \in \mathcal{D}'(c_1)$, soit $\overline{x_m}(C_L^1, A^1) \in \mathcal{D}'(c_1)$. Par conséquent, $\mathcal{V}$ est un ensemble pour les n variables tel que chaque variable a une unique valeur.

Il nous reste à vérifier que l'ensemble $\mathcal{V}$ satisfait la formule booléenne correspondant à $\mathcal{D}$ (*i.e.* que chaque clause c_i avec $1 \leq i \leq q$ est satisfaite). Tout d'abord, notons qu'une conséquence directe des Observations 6.2 à 6.4 est que, pour tout $1 \leq m \leq n$ et $1 \leq i < q$, si $x_m(C_L^i, A^i) \in \mathcal{D}(c_i)$ alors $x_m(C_L^{i+1}, A^{i+1}) \in \mathcal{D}(c_{i+1})$. De façon similaire, pour tout $1 \leq m \leq n$ et $1 \leq i < q$, si $\overline{x_m}(C_L^i, A^i) \in \mathcal{D}(c_i)$ alors $\overline{x_m}(C_L^{i+1}, A^{i+1}) \in \mathcal{D}(c_{i+1})$.

Par conséquent, pour tout $1 \leq m \leq n$ et $2 \leq i \leq q$, si $x_m(C_L^1, A^1) \in \mathcal{D}'(c_1)$ alors $x_m(C_L^i, A^i) \in \mathcal{D}'(c_i)$. De façon similaire, pour tout $1 \leq m \leq n$ et $2 \leq i \leq q$, si $\overline{x_m}(C_L^1, A^1) \in \mathcal{D}'(c_1)$ alors $\overline{x_m}(C_L^i, A^i) \in \mathcal{D}'(c_i)$.

Par le Corollaire 6.1, étant donné que $|\mathcal{D}'(c_i)| = 7n-2$, l'ensemble $\mathcal{D}'(c_i)$ contient, pour un $1 \leq j_i \leq 3$ donné, tous les intervalles simples d'un unique bloc $B_{j_i}^i$ du groupe B^i. De plus, étant donné que $|\mathcal{D}'(c_i)| = 7n-2$, $\mathcal{D}'(c_i)$ est composé de n 2-intervalles entre les blocs A^i et $B_{j_i}^i$ (*cf.* preuve de l'Observation 6.2). Plus précisément, pour tout $1 \leq m \leq n$, soit $x_m(A^i, B_{j_i}^i)$, soit $\overline{x_m}(A^i, B_{j_i}^i)$ est dans $\mathcal{D}'(c_i)$.

Supposons que x_p est le littéral de la clause c_i à la position j_i, avec $1 \leq j_i \leq 3$. Alors par construction, $x_p(A^i, B_{j_i}^i)$ n'existe pas. Ce qui implique que $\overline{x_p}(A^i, B_{j_i}^i) \in \mathcal{D}'(c_i)$.

De plus, par les Observations 6.2 et 6.3, si $\overline{x_p}(A^i, B_{j_i}^i) \in \mathcal{D}'(c_i)$ alors $x_p(B_{j_i}^i, C_R^i) \in \mathcal{D}'(c_i)$ et $\overline{x_p}(C_L^{i+1}, A^{i+1}) \in \mathcal{D}'(c_{i+1})$. Donc, d'après la définition de $\mathcal{V}$, si $\overline{x_p}(C_L^{i+1}, A^{i+1}) \in \mathcal{D}'(c_{i+1})$ alors la valeur de la variable x_p est $Vrai$. Par conséquent, étant donné que x_p est le littéral de la clause c_i à la position j_i, nous en concluons que la clause c_i est satisfaite.

Supposons que $\overline{x_p}$ est le littéral de la clause c_i à la position j_i, avec $1 \leq j_i \leq 3$. Par un raisonnement similaire, on peut vérifier que la clause c_i est satisfaite par le littéral $\overline{x_p}$ à la position j_i.

Ce raisonnement peut être appliqué à toute clause c_i de la formule booléenne. Donc, $\mathcal{V}$ satisfait chaque clause c_i. Par conséquent, à partir d'un sous-ensemble $\{<, \bowtie\}$-comparable $\mathcal{D}' \subseteq \mathcal{D}$ de cardinalité égale à $q \cdot (7n-2)$, il est possible de trouver un ensemble de valeurs $\mathcal{V}$ pour les n variables satisfaisant la formule booléenne. La proposition est prouvée. □

Nous avons démontré en début de section que le sous-problème 2-IP(UNITARY,$\{<, \bowtie\}$) appartient à la classe **NP**. D'autre part, nous avons proposé une transformation polynomiale à partir du problème **NP**-complet EXACT 3-CNF-SAT. Par conséquent, d'après la Proposition 6.3 le sous-problème 2-IP(UNITARY,$\{<, \bowtie\}$) est **NP**-complet. Le Théorème 6.1 est prouvé.

6.5 Complexité paramétrée du sous-problème 2-IP(UNITARY, $\{<, \bowtie\}$)

D'après le Théorème 6.1, trouver le plus grand sous-ensemble $\{<, \bowtie\}$-comparable dans un ensemble de 2-intervalles sur un support UNITARY est un problème **NP**-complet. Dans cette section, nous donnons

un algorithme de complexité paramétrée pour ce problème, dont la complexité repose sur la structure en croisement de l'ensemble de 2-intervalles $\mathcal{D}$.

Plus précisément, nous considérons la complexité en temps du problème par rapport au *nombre maximum de croisements à droite* de $\mathcal{D}$ (notion définie page 78 et notée $\mathsf{FCrossing}(\mathcal{D})$). En effet, dans le contexte des 2-intervalles, on peut raisonnablement espérer que le nombre maximum de croisements à droite sera négligeable par rapport au nombre de 2-intervalles. Par conséquent, une question naturelle à se poser est de savoir si le problème est dans la classe **FPT** pour le paramètre $\mathsf{FCrossing}(\mathcal{D})$. Nous démontrons dans cette section que le problème peut être résolu, pour tout type de support, par un algorithme de programmation dynamique en $O(n^2 \cdot \mathsf{FCrossing}(\mathcal{D}) \cdot 2^{\mathsf{FCrossing}(\mathcal{D})}(\log(n) + \mathsf{FCrossing}(\mathcal{D})))$, où n est le nombre de 2-intervalles de $\mathcal{D}$, et par conséquent, que le problème est dans la classe **FPT** pour le paramètre $\mathsf{FCrossing}(\mathcal{D})$.

Pour tout $D_i \in \mathcal{D}$, soit $T(D_i)$ la cardinalité du plus grand sous-ensemble $\{<, \bowtie\}$-comparable $\mathcal{D}' \subseteq \mathcal{D}$ pour lequel le 2-intervalle D_i est *l'élément le plus à droite* (notion définie page 78). De plus, pour tout $D_i, D_j \in \mathcal{D}$ tel que $D_j \bowtie D_i$, soit $T(D_j|D_i)$ la cardinalité du plus grand sous-ensemble $\{<, \bowtie\}$-comparable $\mathcal{D}' \subseteq \mathcal{D}$ tel que:

1. le 2-intervalle D_j est l'élément le plus à droite de $\mathcal{D}'$ et
2. le 2-intervalle D_i ne fait pas partie du sous-ensemble $\mathcal{D}'$ mais peut être ajouté à $\mathcal{D}'$ pour obtenir un nouveau sous-ensemble $\{<, \bowtie\}$-comparable de cardinalité $|\mathcal{D}'| + 1$.

Clairement, un sous-ensemble $\{<, \bowtie\}$-comparable $\mathcal{D}' \subseteq \mathcal{D}$ de cardinalité maximum pour lequel le 2-intervalle D_i est l'élément le plus à droite peut être obtenu:

1. soit en ajoutant D_i au sous-ensemble $\{<, \bowtie\}$-comparable de cardinalité maximum $\mathcal{D}'' \subseteq \mathcal{D}$ pour lequel le 2-intervalle le plus à droite D_j précède le 2-intervalle D_i, *i.e.* $D_j < D_i$;
2. soit en ajoutant D_i au sous-ensemble $\{<, \bowtie\}$-comparable de cardinalité maximum $\mathcal{D}'' \subseteq \mathcal{D}$ pour lequel le 2-intervalle le plus à droite D_j croise le 2-intervalle D_i, *i.e.* $D_j \bowtie D_i$, et tel que D_i croise ou précède tout 2-intervalle de $\mathcal{D}''$.

Voici une autre façon d'exprimer ces observations:

$$\forall D_i \in \mathcal{D}, \qquad T(D_i) = 1 + \max \begin{cases} \max\{T(D_j) : D_j < D_i\} \\ \max\{T(D_j|D_i) : D_j \bowtie D_i\} \end{cases} \tag{6.1}$$

Il reste à calculer $T(D_j|D_i)$. Pour ce faire, nous étendons la notation $T(D_j|D_i)$ comme suit: pour tout sous-ensemble $\{\bowtie\}$-comparable $\{D_{i_1}, D_{i_2}, \ldots, D_{i_k}\} \subseteq \mathcal{D}$, avec $k \geq 1$, et satisfaisant $\mathsf{Droite}(D_{i_1}) < \mathsf{Droite}(D_{i_2}) < \ldots < \mathsf{Droite}(D_{i_k})$, soit $T(D_{i_1}|D_{i_2}, \ldots, D_{i_k})$ la cardinalité du plus grand sous-ensemble $\{<, \bowtie\}$-comparable $\mathcal{D}' \subseteq \mathcal{D}$ tel que:

1. le 2-intervalle D_{i_1} est l'élément le plus à droite de $\mathcal{D}'$ et
2. les 2-intervalles $\{D_{i_2}, D_{i_3}, \ldots, D_{i_k}\}$ n'appartiennent pas au sous-ensemble $\mathcal{D}'$ mais peuvent être ajoutés à $\mathcal{D}'$ pour obtenir un nouveau sous-ensemble $\{<, \bowtie\}$-comparable de cardinalité $T(D_{i_1}|D_{i_2}, \ldots, D_{i_k}) + k - 1$.

On peut étendre la relation (6.1) pour obtenir la relation de récurrence calculant l'entrée $T(D_{i_1}|D_{i_2},$

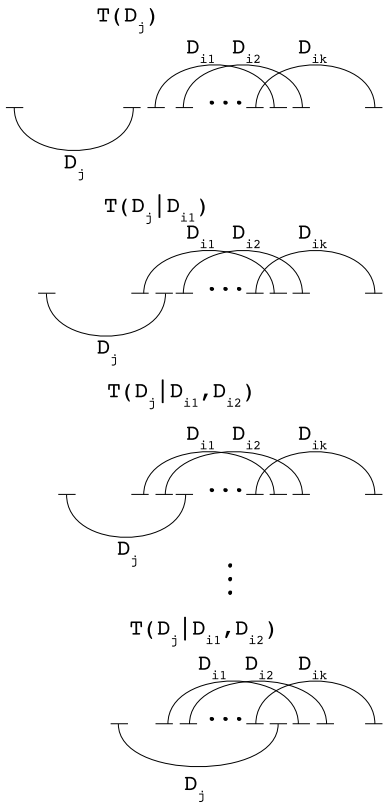


Figure 6.16 – Illustration des différentes conditions de la relation de récurrence (6.2) de la Section 6.5.

$\ldots, D_{i_k})$ de la table de programmation dynamique:

$$T(D_{i_1}|D_{i_2},\ldots,D_{i_k}) = 1+ \max \begin{cases} \max\{T(D_j)|D_j \text{ satisfait la condition } (1)\} \\ \max\{T(D_j|D_{i_1})|D_j \text{ satisfait la condition } (2)\} \\ \max\{T(D_j\ |D_{i_1},D_{i_2})|D_j \text{ satisfait la condition } (3)\} \\ \quad\vdots \\ \max\{T(D_j\ |D_{i_1},\ldots,D_{i_k})|D_j \text{ satisfait la condition } (k+1)\} \end{cases} \tag{6.2}$$

où, pour $1 \leq i \leq k+1$, condition (i) est définie comme suit:

$$\text{condition } (i) \qquad \begin{cases} D_j \bowtie D_{i_r} & \forall\ 0 < r < i \quad \text{(condition de croisement)} \\ D_j < D_{i_s} & \forall\ i \leq s < k+1 \quad \text{(condition de succession)} \end{cases}$$

Une illustration des différentes conditions de cette relation de récurrence est donnée en Figure 6.16.

Il en découle que les entrées de la forme $T(D_i|*)$ dépendent uniquement des entrées de la forme $T(D_j|*)$ où $D_j < D_i$ ou $D_j \bowtie D_i$. Du point de vue du calcul, ceci implique que le calcul des entrées de la forme $T(D_i|*)$ dépend uniquement du calcul des entrées de la forme $T(D_j|*)$ où $\mathsf{Droite}(D_j) <$

$\mathsf{Droite}(D_i)$. Le lemme suivant donne une borne supérieure sur la taille de la table de programmation dynamique T par rapport à $\mathsf{FCrossing}(\mathcal{D})$.

Lemme 6.9. *Le nombre d'entrées distinctes de la table de programmation dynamique T est inférieur ou égal à $|\mathcal{D}| \cdot 2^{\mathsf{FCrossing}(\mathcal{D})}$.*

Preuve. Pour tout 2-intervalle $D_i \in \mathcal{D}$, le nombre de sous-ensembles $\{\bowtie\}$-comparables distincts pour lesquels D_i est l'élément le plus à gauche est inférieur ou égal à $2^{\mathsf{FCrossing}(\mathcal{D})}$, et par conséquent il existe au plus $2^{\mathsf{FCrossing}(\mathcal{D})}$ entrées distinctes de la forme $T(D_i|*)$ dans la table de programmation dynamique T. □

Algorithme 3: Un algorithme de complexité en temps $\mathbf{O}(n^2 \cdot \mathsf{FCrossing}(\mathcal{D}) \cdot 2^{\mathsf{FCrossing}(\mathcal{D})}(\log(n) + \mathsf{FCrossing}(\mathcal{D})))$ pour résoudre le sous-problème 2-IP$(n_1, \{<, \bowtie\})$ $\forall n_1 \in \{$UNLIMITED, UNITARY, DISJOINT$\}$

Données : Un ensemble de n 2-intervalles $\mathcal{D}$.
Résultat : La cardinalité maximum d'un sous-ensemble $\{<, \bowtie\}$-comparable de $\mathcal{D}$.
début
1. Trier l'ensemble $\mathcal{D}$ par rapport à leur intervalle droit. Supposons que l'ensemble ordonné de 2-intervalles est donné par $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$, *i.e.* $\mathsf{Droite}(D_i) < \mathsf{Droite}(D_j)$ implique $i < j$. Tous les sous-ensembles ordonnés considérés dans la suite de l'algorithme sont supposés être ordonnés à l'aide de cet ordre.
2. **pour** i *de* 1 *à* n **faire**
3. Renseigner l'entrée $T(D_i)$.
4. Pour tous les ensembles non-vides ordonnés $\{D_{i_1}, D_{i_2}, \ldots, D_{i_q}\} \subseteq \mathcal{D}$ tels que $\{D_i\} \cup \{D_{i_1}, D_{i_2}, \ldots, D_{i_q}\}$ est un sous-ensemble ordonné de 2-intervalles $\{\bowtie\}$-comparable avec $\mathsf{Droite}(D_i) < \mathsf{Droite}(D_{i_1}) < \ldots < \mathsf{Droite}(D_{i_q})$, renseigner l'entrée $T(D_i|D_{i_1}, D_{i_2}, \ldots, D_{i_q})$ à l'aide de la relation de récurrence (6.2).
5. **retourner** *la plus grande entrée* $T(D_i)$

fin

L'Algorithme 3 permet de trouver la cardinalité du plus grand sous-ensemble $\{<, \bowtie\}$-comparable dans un ensemble de 2-intervalles. En utilisant une structure de données adaptée pour efficacement rechercher des 2-intervalles, nous obtenons le résultat suivant.

Proposition 6.4. *L'Algorithme 3, qui retourne la cardinalité du sous-ensemble $\{<, \bowtie\}$-comparable de cardinalité maximum d'un ensemble de 2-intervalles $\mathcal{D}$, a une complexité $\mathbf{O}(n^2 \cdot \mathsf{FCrossing}(\mathcal{D}) \cdot 2^{\mathsf{FCrossing}(\mathcal{D})}(\log(n) + \mathsf{FCrossing}(\mathcal{D})))$ avec $n = |\mathcal{D}|$.*

Notre approche est basée sur le théorème suivant.

Théorème 6.2 ([44]). *Soit $\mathcal{I}$ une collection finie de n intervalles sur une ligne. Une structure de données mémorisant $\mathcal{I}$ peut être construite en $\mathbf{O}(n \log n)$. En interrogeant cette structure de données, il est possible de trouver les intervalles de $\mathcal{I}$ qui sont entièrement contenus dans un intervalle donné en $\mathbf{O}(n \log n + k)$ où k est le nombre de 2-intervalles entièrement contenus.*

Lemme 6.10. *Soit $\mathcal{D}$ une collection finie de n 2-intervalles. Après une étape de pré-calcul en temps $\mathbf{O}(n \log n)$ et en espace $\mathbf{O}(n \log n)$, il est possible de lister:*

1. *les 2-intervalles de $\mathcal{D}$ qui sont entièrement à gauche d'un 2-intervalle donné;*
2. *les 2-intervalles de $D_i \in \mathcal{D}$ pour lesquels* $\mathsf{Gauche}(D_i)$ *et* $\mathsf{Droite}(D_i)$ *sont entièrement contenus dans deux intervalles donnés.*

en $\mathbf{O}(n \log n + k)$ *où k est le nombre de 2-intervalles listés.*

Preuve. Nous utilisons une structure de données composée de deux sous-structures de données distinctes, comme défini dans le Théorème 6.2.

1. Nous associons à chaque 2-intervalle $D_i \in \mathcal{D}$ son intervalle couvrant $\mathsf{ICouvrant}(D_i)$ et mémorisons tous les intervalles couvrants dans la structure de données du Théorème 6.2. Sélectionner les 2-intervalles de $\mathcal{D}$ qui sont entièrement à gauche d'un 2-intervalle donné D_i est équivalent à sélectionner les intervalles couvrants qui sont entièrement contenus dans la partie précédant l'intervalle $\mathsf{Gauche}(D_i)$. La complexité en temps découle du Théorème 6.2.
2. Nous mémorisons l'intervalle $\mathsf{Gauche}(D_i)$ de chaque 2-intervalle D_i dans la structure de données du Théorème 6.2. Tout d'abord, nous trouvons les 2-intervalles D_i tels que l'intervalle $\mathsf{Gauche}(D_i)$ est complètement contenu dans le premier intervalle requête. Puis, parmi ces 2-intervalles nous sélectionnons ceux tels que l'intervalle $\mathsf{Droite}(D_i)$ est complètement contenu dans le second intervalle requête. Clairement, la première étape est en $\mathbf{O}(n \log n + k)$ et la seconde est en $\mathbf{O}(k)$.

□

Lemme 6.11. *Soit $D_j \in \mathcal{D}$ tel que toutes les entrées de la table de programmation dynamique de la forme $T(D_k|*)$ avec $\mathsf{Droite}(D_k) < \mathsf{Droite}(D_j)$ ont d'ores et déjà été calculées durant une étape précédente. Alors, pour tout sous-ensemble $\{\bowtie\}$-comparable $\{D_{i_1}, D_{i_2}, \ldots, D_{i_k}\} \subseteq \mathcal{D}$, $k \geq 1$, satisfaisant $\mathsf{Droite}(D_j) < \mathsf{Droite}(D_{i_1}) < \mathsf{Droite}(D_{i_2}) < \ldots < \mathsf{Droite}(D_{i_k})$, il est possible de calculer l'entrée de la table de programmation dynamique $T(D_{i_1}|D_{i_2}, \ldots D_{i_k})$ à l'aide de la relation de récurrence* (6.2) *en* $\mathbf{O}(n \cdot \mathsf{FCrossing}(\mathcal{D})(\log(n) + \mathsf{FCrossing}(\mathcal{D})))$.

Preuve. Nous avons besoin d'un couplage injectif associant à tout sous-ensemble $\{\bowtie\}$-comparable $\{D_{i_1}, D_{i_2}, \ldots, D_{i_k}\} \subseteq \mathcal{D}$, $k \geq 1$, satisfaisant $\mathsf{Droite}(D_{i_1}) < \mathsf{Droite}(D_{i_2}) < \ldots < \mathsf{Droite}(D_{i_k})$, son indice dans la table de programmation dynamique T. Soit π une numérotation de $\mathcal{D}$ telle que les 2-intervalles sont numérotés par rapport à leurs intervalles droits, *i.e.* pour tout $D_i, D_j \in \mathcal{D}$, $\mathsf{Droite}(D_i) < \mathsf{Droite}(D_j)$ implique $\pi(D_i) < \pi(D_j)$. Soit $\mathcal{D}^{\bowtie}$ l'ensemble des sous-séquences ordonnées de $\{1, 2, \ldots, n\}$ définies comme suit: pour tout sous-ensemble $\{\bowtie\}$-comparable $\{D_{i_1}, D_{i_2}, \ldots, D_{i_k}\} \subseteq \mathcal{D}$, $k \geq 1$, satisfaisant $\mathsf{Droite}(D_{i_1}) < \mathsf{Droite}(D_{i_2}) < \ldots < \mathsf{Droite}(D_{i_k})$, l'ensemble $\mathcal{D}^{\bowtie}$ contient la séquence ordonnée $(\pi(D_{i_1}), \pi(D_{i_2}), \ldots, \pi(D_{i_k}))$. Clairement, il est possible de comparer deux séquences de $\mathcal{D}^{\bowtie}$, par exemple à l'aide de l'ordre lexicographique, en $\mathbf{O}(\mathsf{FCrossing}(\mathcal{D}))$, étant donné que les séquences de $\mathcal{D}^{\bowtie}$ sont de longueur au plus $\mathsf{Depth}(\mathcal{D}) \leq \mathsf{FCrossing}(\mathcal{D}) + 1$.

Par conséquent, utiliser n'importe quelle structure de données classique pour insérer et rechercher garantit un temps d'exécution logarithmique [40], et il est possible d'insérer ou rechercher une séquence donnée de $\mathcal{D}^{\bowtie}$ en $\mathbf{O}(\mathsf{FCrossing}(\mathcal{D})(\log(n) + \mathsf{FCrossing}(\mathcal{D})))$. Nous nous intéressons maintenant au calcul de $T(D_{i_1}|D_{i_2}, \ldots D_{i_k})$. Pour chaque condition (i) de la relation de récurrence (6.2), nous devons trouver les 2-intervalles D_j satisfaisant $D_j \bowtie \{D_{i_r}|0 \leq r < i\}$ et $D_j < \{D_{i_s}|i \leq s < k+1\}$. D'après le Lemme 6.10, ceci peut être fait en $\mathbf{O}(\log n + p_i)$ où p_i est le nombre de 2-intervalles satisfaisant la condition (i). Alors il en découle qu'il est possible de trouver la valeur maximale de condition (i) en $\mathbf{O}(p_i \cdot \mathsf{FCrossing}(\mathcal{D})(\log(n) + \mathsf{FCrossing}(\mathcal{D})))$. En sommant sur toutes les conditions (i) et en observant que $\sum_{i=1}^{k+1} p_i \leq n$, nous obtenons un algorithme de complexité $\mathbf{O}(n \cdot \mathsf{FCrossing}(\mathcal{D})(\log(n) + \mathsf{FCrossing}(\mathcal{D}))$ pour calculer les entrées de la table de programmation

dynamique $T(D_{i_1}|D_{i_2},\dots D_{i_k})$. Il reste à insérer la séquence ordonnée $(\pi(D_{i_1}),\pi(D_{i_2}),\dots,\pi(D_{i_k}))$ dans la structure de données pour les prochaines requêtes. Or ceci peut être fait en $\mathbf{O}(\mathsf{FCrossing}(\mathcal{D})\text{-}(\log(n)+\mathsf{FCrossing}(\mathcal{D})))$. □

Preuve de la Proposition 6.4. L'exactitude de l'algorithme découle de la relation de récurrence (6.2). Il nous reste à prouver la complexité en temps. Trier l'ensemble des 2-intervalles $\mathcal{D}$ par rapport à leurs intervalles droits peut être fait en $\mathbf{O}(n\log n)$. D'après le Lemme 6.11, chaque entrée de la forme $T(D_i|*)$ peut être calculée en $\mathbf{O}(n\cdot\mathsf{FCrossing}(\mathcal{D})(\log(n)+\mathsf{FCrossing}(\mathcal{D})))$. Étant donné que le nombre d'entrées distinctes dans la table de programmation dynamique T a pour borne supérieure $n\cdot 2^{\mathsf{FCrossing}(\mathcal{D})}$ (Lemme 6.9), il en découle que l'algorithme a une complexité en $\mathbf{O}(n^2\cdot\mathsf{FCrossing}(\mathcal{D})\cdot 2^{\mathsf{FCrossing}(\mathcal{D})}(\log(n)+\mathsf{FCrossing}(\mathcal{D})))$. □

Corollaire 6.2. *Le sous-problème* 2-IP$(n_1,\{<,\bowtie\})$ $\forall n_1\in\{$UNLIMITED, UNITARY, DISJOINT$\}$ *est dans la classe* **FPT** *pour le paramètre* $\mathsf{FCrossing}(\mathcal{D})$.

6.6 Conclusion et perspectives

Les résultats que nous avons fournis dans [24] répondent quasiment entièrement à tous les problèmes restés ouverts dans [93, 94]. Dans le contexte du problème 2-IP nous avons résolu trois problèmes ouverts de [93, 94]. Nous avons donné un algorithme optimal de complexité en temps $\mathbf{O}(n\log n)$ pour le sous-problème 2-IP(UNLIMITED,$\{\sqsubset\}$), améliorant ainsi la meilleure complexité existante pour ce problème. Nous avons également défini un algorithme de complexité en temps $\mathbf{O}(n^2\sqrt{n})$ pour le sous-problème 2-IP(DISJOINT,$\{\sqsubset,\bowtie\}$). Finalement, nous avons prouvé que le sous-problème 2-IP(UNITARY,$\{<,\bowtie\}$) est **NP**-complet, et en supplément, nous avons montré que ce sous-problème dans sa généralité appartient à la classe **FPT** pour un paramètre fortement lié à la structure de croisement de l'ensemble des 2-intervalles. Tous ces résultats sont particulièrement intéressants puisqu'ils sont fortement reliés, dans le contexte de la biologie moléculaire, à la recherche de motifs dans des structures de molécules d'ARN.

Il serait intéressant de répondre au dernier problème ouvert dans ce domaine, *i.e.* de déterminer si il existe un algorithme polynomial pour le sous-problème 2-IP(DISJOINT,$\{<,\bowtie\}$), *i.e.* la recherche du plus grand sous-ensemble $\{<,\bowtie\}$-comparable d'un ensemble de 2-intervalles sur un support DISJOINT. Notons que le problème 2-IP(DISJOINT,$\{<,\bowtie\}$) a une formulation immédiate en termes de couplages contraints sur des graphes généraux: étant donné un graphe G avec un ordonnancement linéaire π de ses sommets, le problème 2-IP(DISJOINT,$\{<,\bowtie\}$) est équivalent à rechercher un couplage de cardinalité maximum $\mathcal{M}$ de G tel que pour tout couple d'arcs distincts $\{u,v\}$ et $\{u',v'\}$ de $\mathcal{M}$ ni $\max\{\pi(u),\pi(v)\}<\min\{\pi(u'),\pi(v')\}$, ni $\max\{\pi(u'),\pi(v')\}<\min\{\pi(u),\pi(v)\}$. Au vu de la Table 6.1, nous conjecturons qu'il existe un algorithme polynomial pour ce problème.

D'autre part, savoir si le sous-problème 2-IP$(n_1,\{<,\bowtie\})$ $\forall n_1\in\{$UNLIMITED, UNITARY, DISJOINT$\}$ est dans la classe **FPT** pour le paramètre $\mathsf{Depth}(\mathcal{D})$ (rappelons en effet que $\mathsf{FCrossing}(\mathcal{D})+1\geq\mathsf{Depth}(\mathcal{D})$) reste un problème ouvert.

CHAPITRE 7

Le problème MRSO

7.1 Introduction

Dans ce chapitre, nous développons des résultats publiés dans [20] et obtenus en collaboration avec Guillaume Fertin, Stéphane Vialette et Danny Hermelin. Les travaux présentés dans ce chapitre diffèrent de ceux présentés précédemment dans cette partie puisqu'ils concernent le design d'ARN. Plus précisément, dans ce chapitre, nous nous intéressons au design de molécules d'ARN codant pour des *sélénoprotéines* (*i.e.* des protéines contenant le $21^{ème}$ acide aminé – la *sélénocystéine*).

Le $21^{ème}$ acide aminé – la Sélénocystéine (Sec) – a été découvert il y a une dizaine d'années [26]. La sélénocystéine est codée par un triplet de nucléotides UGA, qui est généralement interprété comme le signal de terminaison de la traduction (*i.e.* un codon STOP). La synthèse de la sélénocystéine requiert une interaction supplémentaire sous la forme d'un élément SECIS (acronyme de *SelenoCysteine Insertion Sequence*). Un élément SECIS correspond à un motif structurel de l'ARNm, qui est donc défini aussi bien par sa séquence de nucléotides que par sa structure. Il semblerait que ce soit cette structure qui force le ribosome à synthétiser la sélénocystéine.

L'élément SECIS est une région de l'ARNm qui est traduite en acides aminés. Par conséquent, lors de la conception de nouvelles sélénoprotéines, on recherche la séquence d'ARNm la plus proche possible de la séquence SECIS consensus et respectant les contraintes structurelles imposées. La conception de nouvelles sélénoprotéines requiert donc la conception d'une séquence d'ARNm aussi proche que possible de la séquence SECIS consensus, qui soit capable de se conformer dans l'espace comme l'élément SECIS (*i.e.* dont les structures secondaires sont similaires), et dont la traduction est la plus proche possible de la séquence d'acides aminés initiale.

Dans [3, 4], Backofen *et al.* ont formulé une généralisation de ce problème: le problème MRSO – acronyme de mRNA Structure Optimization. Ce problème correspond à la conception d'une séquence d'ARNm de similarité maximale, en termes de codons, à une séquence d'ARNm donnée (et par conséquent à une protéine donnée) et satisfaisant une structure secondaire particulière.

7.2 Notations

Avant de présenter les résultats de complexité connus pour le problème MRSO, nous présentons quelques notions de théorie des graphes [45]. Tous les graphes considérés dans ce chapitre sont *simples* (*i.e.* il n'existe pas d'arête reliant un sommet à lui même et entre deux sommets il existe au plus une arête). Étant donné un graphe G, $\mathbf{V}(G)$ (*resp.* $\mathbf{E}(G)$) représente l'ensemble des sommets (*resp.* arêtes) de G.

Définition 7.1 (Graphe linéaire). Un *graphe linéaire* est un graphe G dont les sommets respectent un ordonnancement linéaire (*i.e.* ordre sur une droite).

Dans le reste de ce chapitre, étant donné un graphe linéaire G, nous supposons que v_i est le sommet portant l'étiquette i dans $\mathbf{V}(G)$. Un *intervalle* $V_{i,j}$ dans un graphe linéaire G est défini comme étant le sous-ensemble de sommets consécutifs $\{v_i, v_{i+1}, \ldots, v_j\} \subseteq \mathbf{V}(G)$. Soient $e_1 = (v_i, v_j)$ et $e_2 = (v_x, v_y)$ deux arêtes d'un graphe linéaire G quelconque, les arêtes e_1 et e_2 *se croisent*, noté $e_1 \bowtie e_2$, si $i < x < j < y$ ou $x < i < y < j$.

Définition 7.2 (Graphe planaire extérieur). Un graphe est *planaire extérieur* s'il peut être représenté dans le plan de façon telle que ses sommets soient disposés sur un cercle et que ses arêtes ne se coupent pas (en dehors de leurs extrémités).

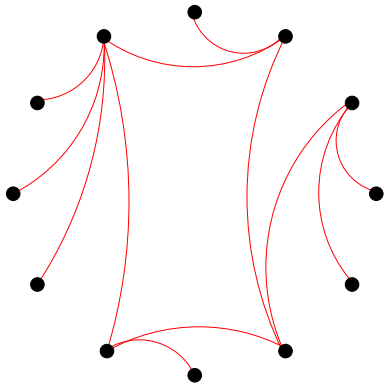


Figure 7.1 – Un graphe planaire extérieur.

Définition 7.3 (Plongement linéaire planaire extérieur d'un graphe). Étant donné un graphe $G = (V, E)$, un *plongement linéaire planaire extérieur* de G est un ordonnancement linéaire π des sommets de V tel que $\forall (i,j),(k,l) \in E$ si $i < k$ alors soit $j < k$, soit $l < j$.

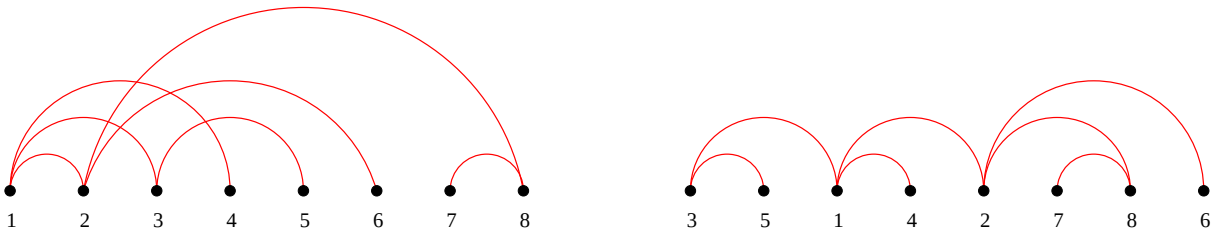


Figure 7.2 – À gauche (*resp.* droite) un plongement linéaire (*resp.* planaire extérieur) d'un graphe.

Étant donné un graphe G, il est possible de vérifier en temps linéaire si G est planaire extérieur, et si c'est le cas, de trouver son plongement linéaire planaire extérieur [79, 87]. Par conséquent, nous supposerons, par la suite, que l'étiquetage du graphe G est induit par l'ordonnancement des sommets de G dans son plongement linéaire planaire extérieur.

Nous introduisons maintenant des notations permettant de décrire et comparer des séquences d'ARNm. Soit $\Sigma = \{A, C, G, U\}$ un alphabet représentant les quatre nucléotides possibles dans une molécule

d'ARNm. Nous appelons *paire de nucléotides complémentaires* toute paire $\{A, U\}$ ou $\{C, G\}$[1]. De plus, à l'instar de la plupart des modèles, nous supposons que les liaisons hydrogènes n'apparaissent qu'entre des nucléotides complémentaires éloignés d'au moins trois positions dans la séquence d'ARNm. Nous appelons *codon* tout mot composé de trois lettres de Σ. Une séquence d'ARNm est alors la concaténation de n codons, en d'autres termes, un mot de Σ^{3n}. Soit $S_N = s_1 \dots s_{3n}$ une séquence d'ARNm. Pour tout $1 \leq i \leq n$, le codon i de S_N correspond au sous-mot de S_N de longueur 3 finissant en position $3i$, noté $C_i^{S_N} = s_{3i-2}\ s_{3i-1}\ s_{3i}$.

Supposons que nous souhaitons calculer la similarité entre deux séquences d'ARNm $S = s_1 \dots s_{3n}$ et $T = t_1 \dots t_{3n}$, et ceci de façon à maximiser la similarité entre les séquences d'acides aminés produites lors de la traduction de S et T. Pour ce faire, il nous faut un ensemble $\mathcal{F} = \{f_1, \dots, f_n\}$ de n fonctions, appelées *fonctions de similarité* de S, tel que pour tout $1 \leq i \leq n$, chaque fonction f_i est de la forme $f_i : \Sigma^3 \to \mathbb{Q}$. La fonction f_i affecte une valeur au codon C_i^T dépendante de son niveau de similarité avec le codon C_i^S. La similarité globale entre les séquences S et T est définie comme suit:

$$\text{sim}(S, T) = \sum_{i=1}^{n} f_i(C_i^T) = \sum_{i=1}^{n} f_i(t_{3i-2}, t_{3i-1}, t_{3i})$$

Notons qu'étant donné un ensemble de n fonctions de similarité $\mathcal{F} = \{f_1, \dots, f_n\}$ de S, le calcul de la similarité globale entre S et T ne nécessite pas d'informations supplémentaires sur la séquence S. Soit S une séquence quelconque d'ARNm de longueur $3n$. Nous appelons *structure secondaire* de S, notée $S_S[S] \subseteq \{\{i, j\} | 1 \leq i < j \leq 3n\}$, tout ensemble de couple d'entiers distincts de $\{1, 2, \dots, 3n\}$. Chaque couple d'entier de $S_S[S]$ représente un lien hydrogène du repliement de S. Étant donné que dans notre modèle, nous supposons que chaque nucléotide ne peut se lier, par un lien hydrogène, qu'à au plus un autre nucléotide, nous supposons également que chaque entier apparaît dans au plus un couple de $S_S[S]$. De plus, pour tout $1 \leq i \leq 3n - 2$, il n'existe pas de couple de la forme $\{i, i + 1\}$ ou $\{i, i + 2\}$ dans $S_S[S]$.

Nous appelons *graphe de structure* de S le graphe linéaire Γ tel que $\mathbf{V}(\Gamma) = \{u_1, u_2, \dots, u_{3n}\}$, $\mathbf{E}(\Gamma) = \{(u_i, u_j) | \{i, j\} \in S_S[S]\}$ et tout sommet est de degré au plus 1 (*cf.* Figure 7.3). Soient Γ un graphe de structure quelconque avec $\mathbf{V}(\Gamma) = \{u_1, \dots, u_{3n}\}$, et $T = t_1 \dots t_{3n}$ une séquence quelconque d'ARNm. Les nucléotides t_i et t_j sont dits *compatibles avec* Γ, si soit (t_i, t_j) forme une paire de nucléotides complémentaires, soit $(u_i, u_j) \notin \mathbf{E}(\Gamma)$. La séquence T toute entière est compatible avec Γ, si toute paire de nucléotides (t, t') de T, est compatible avec Γ.

Nous définissons formellement le problème MRSO comme suit.

MRSO
DONNÉES*: Un ensemble de n fonctions de similarité $\mathcal{F}$, une séquence S d'ARNm de longueur $3n$ et un graphe de structure Γ avec $\mathbf{V}(\Gamma) = \{u_1, u_2, \dots, u_{3n}\}$.*
QUESTION*: Trouver une séquence T d'ARNm compatible avec Γ telle que la similarité selon $\mathcal{F}$ entre S et T est maximisée.*

Le problème MRSO peut également être défini comme un problème d'affectation de nucléotides (*i.e.* de lettres de $\{A, C, G, U\}$) aux sommets de Γ, tel que les nucléotides affectés soient compatibles avec Γ, et telle que la similarité selon $\mathcal{F}$ entre S et la séquence induite par cette affectation soit maximisée.

La priorité étant la similarité entre les séquences d'acides aminés, nous utilisons une représentation plus adaptée du graphe de structure, à savoir le *graphe de structure induit* défini dans [3].

[1] Notons que nos résultats peuvent être étendus à d'autres paires de nucléotides comme les paires wobble par exemple.

Définition 7.4 (Graphe de structure induit). Soit Γ un graphe de structure avec $\mathbf{V}(\Gamma) = \{u_1, u_2, \ldots, u_{3n}\}$. Le *graphe de structure induit* de Γ, noté G_Γ, est le graphe linéaire tel que:

$$\mathbf{V}(G_\Gamma) = \{v_1, v_2, \ldots, v_n\}$$
$$\mathbf{E}(G_\Gamma) = \{(v_x, v_y) | \exists (u_i, u_j) \in \mathbf{E}(\Gamma),\ i \in \{3x-2, 3x-1, 3x\} \text{ et } j \in \{3y-2, 3y-1, 3y\}\}$$

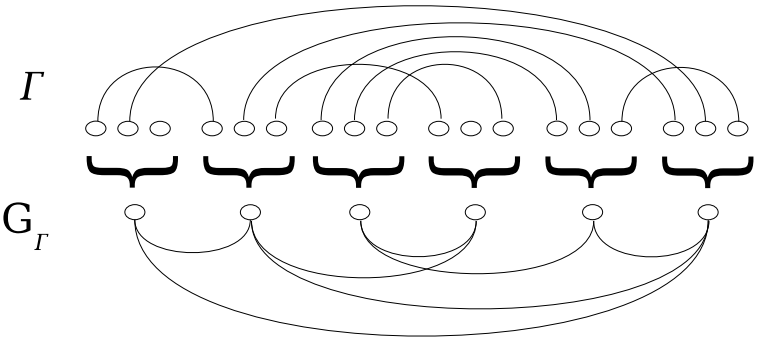


Figure 7.3 – Un graphe de structure Γ et son graphe de structure induit G_Γ.

D'après cette définition, le graphe de structure induit peut être vu comme une version compressée du graphe de structure, où trois sommets consécutifs du graphe de structure sont réunis en un sommet compressé. Chaque sommet compressé représente un codon ou un triplet de nucléotides (*cf.* Figure 7.3). Notons que cette compression n'est pas sans pertes car plusieurs arêtes (au plus trois) de Γ peuvent correspondre à une même arête de G_Γ. Malgré tout, nous pouvons supposer que les arêtes du graphe de structure induit G_Γ sont étiquetés à l'aide des informations nécessaires à la déduction des contraintes de complémentarité définies par les arêtes de Γ. Cet étiquetage ne requiert qu'un accroissement constant de l'espace utilisé. Par conséquent, nous supposons par la suite que ces informations sont implicitement fournies dans G_Γ.

Cette hypothèse nous permet de ne considérer, par la suite, que le graphe de structure induit, et de parler de codons compatibles avec G_Γ plutôt que de nucléotides compatibles avec Γ. Étant donnés deux codons C_i et $C_j \in \Sigma^3$ respectivement affectés aux sommets v_i et v_j, le couple de codons (C_i, C_j) est compatible avec G_Γ, si tout couple (t, t') de nucléotides, avec $t \in C_i$ et $t' \in C_j$, est compatible avec Γ. Nous considérons alors le problème MRSO comme un problème d'affectation de codons, et supposons que les instances de ce problème sont de la forme $(G_\Gamma, \mathcal{F})$, où G_Γ est un graphe de structure induit et Γ un graphe de structure pouvant toujours être inféré à partir de G_Γ.

Dans [3], Backofen *et al.* ont prouvé que le problème MRSO est **NP**-complet en proposant une transformation polynomiale à partir du problème **NP**-complet EXACT 3-CNF-SAT. Ils ont également proposé un algorithme de complexité en temps linéaire pour résoudre le problème MRSO dans le cas restreint où toute instance est de la forme $(G_\Gamma, \mathcal{F})$ tel que G_Γ est un graphe planaire extérieur (ce qui est le cas de la structure secondaire de l'élément SECIS). Dans le reste de ce chapitre, $\mathcal{A}_{\text{OP}}$ représentera cet algorithme. Finalement, ils ont proposé un algorithme de 2-approximation pour le cas général.

Plus récemment, Bongartz *et al.* [27] ont poursuivi l'étude de l'approximation du problème MRSO en prouvant que ce dernier est **APX**-dur. Bongartz *et al.* ont également proposé un algorithme de 4-approximation, qui malgré son plus grand ratio d'approximation, a le mérite d'être plus simple que celui

de Backofen *et al.* Compte tenu de l'**APX**-complétude du problème MRSO dans le cas général, Bongartz *et al.* ont proposé de considérer la complexité paramétrée du problème.

7.3 Complexité paramétrée du problème MRSO

Nous proposons de considérer deux paramètres naturels illustrés en Figure 7.4:

1. le nombre de sommets de degré trois dans le graphe de structure induit G_Γ, noté δ et
2. le nombre d'arêtes se croisant dans G_Γ, noté χ.

Formellement, $\delta = |\{v \in \mathbf{V}(G_\Gamma) : d(v) = 3\}|$ et $\chi = |\{(e_1, e_2) : e_1, e_2 \in \mathbf{E}(G_\Gamma) \wedge e_1 \bowtie e_2\}$. Il faut noter que le paramètre χ est défini sur le plongement linéaire naturel du graphe G_Γ, *i.e.* l'ordonnancement de $\mathbf{V}(G_\Gamma)$ est donné par l'ordonnancement des codons dans la séquence d'ARNm source.

Nous avons sélectionné ces paramètres car nous pensons qu'en pratique leurs valeurs sont petites. En effet, si on considére le paramètre δ, un sommet de degré 3 dans G_Γ correspond à un codon dont les trois nucléotides sont liés par des liaisons hydrogènes à des nucléotides de trois autres codons de la séquence. Bien que cette situation puisse arriver en toute généralité dans le repliement de la séquence d'ARNm, on peut supposer que cette situation est rare en raison des contraintes naturelles géométriques et thermodynamiques d'un tel repliement.

Si on considère le paramètre χ, on peut noter que jusqu'à présent, une grande partie des structures secondaires d'ARNm était considérée comme étant représentable par un graphe planaire extérieur, *i.e.* une représentation sans croisement d'arcs. Par conséquent, on peut supposer que la plupart des structures secondaires d'ARNm possède un petit nombre de croisements d'arcs. Finalement, l'étude de ce paramètre fut proposé dans [27].

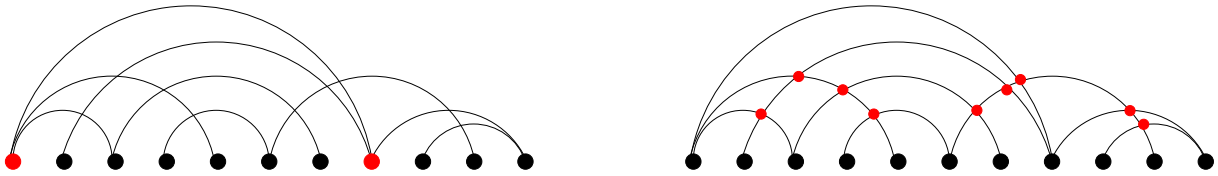


Figure 7.4 – À gauche (*resp.* droite) une illustration du paramètre δ (*resp.* χ): $\delta = 2$ (*resp.* $\chi = 9$).

Comme nous allons le démontrer, il existe deux algorithmes pour résoudre le problème MRSO de complexités en temps paramétrées respectivement par δ et χ. Afin de prouver ce résultat, nous commençons par décrire un algorithme général, puis nous démontrerons comment il peut être appliqué à chacun de ces paramètres. Avant de décrire cet algorithme, nous introduisons les notions de *bipartition et bipartition propre des arêtes* de G_Γ.

Définition 7.5 (Bipartition et bipartition propre des arêtes de G_Γ). Soit G_Γ un graphe de structure induit possédant n sommets. Une *bipartition des arêtes* $\mathcal{P} = \{\mathcal{E}_t, \mathcal{E}_b\}$ de G_Γ est une partition des arêtes de G_Γ en deux parties $\mathcal{E}_t$ et $\mathcal{E}_b$, telle que $\mathcal{E}_t \cup \mathcal{E}_b = \mathbf{E}(G_\Gamma)$, $\mathcal{E}_t \cap \mathcal{E}_b = \emptyset$ et $\mathcal{E}_t \neq \emptyset$. $\mathcal{P}$ est une *bipartition propre des arêtes* de G_Γ si le sous-graphe $G_\Gamma[\mathcal{E}_t]$ est planaire extérieur.

Une représentation commode d'une bipartition $\mathcal{P} = \{\mathcal{E}_t, \mathcal{E}_b\}$ des arêtes d'un graphe de structure induit G_Γ est obtenue en dessinant les sommets sur une ligne d'après le plongement linéaire de G_Γ et en dessinant les arêtes de $\mathcal{E}_t$ (*resp.* $\mathcal{E}_b$) au dessus (*resp.* en dessous) de cette ligne (*cf.* Figure 7.5).

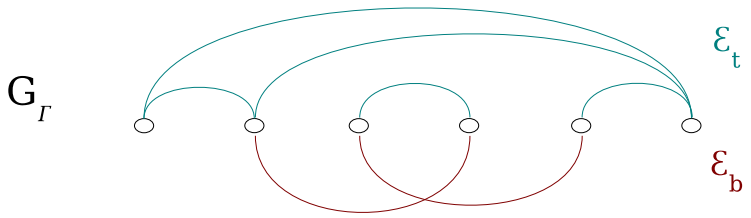


Figure 7.5 – Représentation d'une bipartition $\mathcal{P} = \{\mathcal{E}_t, \mathcal{E}_b\}$ des arêtes d'un graphe de structure induit G_Γ.

Nous décrivons ci-après l'algorithme général que nous noterons $\mathcal{A}_{\text{NEB}}$. L'algorithme $\mathcal{A}_{\text{NEB}}$ ne s'applique qu'aux bipartitions propres des arêtes de G_Γ dont on connaît préalablement le nombre d'arêtes dans $\mathcal{E}_b$. Nous démontrons qu'en considérant le paramètre δ ou χ comme fixé, on peut obtenir facilement une telle bipartition.

L'algorithme $\mathcal{A}_{\text{NEB}}$ utilise l'algorithme $\mathcal{A}_{\text{OP}}$ [3] comme une sous-procédure dans le cas où G_Γ est planaire extérieur. Rappelons que l'algorithme $\mathcal{A}_{\text{OP}}$ résout le problème MRSO dans ce cas particulier en temps linéaire. De plus, l'algorithme $\mathcal{A}_{\text{OP}}$ peut être utilisé même si G_Γ est planaire extérieur et que son plongement linéaire naturel (*i.e.* respectant l'ordre d'apparition des codons) contient des arêtes se croisant. Finalement, il faut noter que l'algorithme $\mathcal{A}_{\text{OP}}$ peut facilement être modifié pour trouver une séquence d'ARNm cible optimale étant donné un sous-ensemble de codons fixés à l'avance.

Par exemple, étant donnée une instance $(G_\Gamma, \mathcal{F} = \{f_1, \ldots, f_n\})$ pour le problème MRSO, on peut modifier l'algorithme $\mathcal{A}_{\text{OP}}$ afin de calculer la séquence d'ARNm optimale débutant avec le codon AAA. Pour ce faire, nous remplaçons l'instance d'origine par une nouvelle instance $(G_\Gamma, \mathcal{F}' = \{f'_1, \ldots, f'_n\})$ telle que

1. $f'_1(AAA) = f_1(AAA)$;
2. pour tout codon $C_1 \neq AAA$, $f'_1(C_1) = -\infty$;
3. pour tout $2 \leq i \leq n$, $f'_i = f_i$.

On peut étendre cet exemple en considérant dans la définition suivante n'importe quelle *affectation de codon* pour les sommets de $\mathbf{V}(G_\Gamma)$, *i.e.* toute fonction ϕ de la forme $\phi : V' \to \Sigma^3$ où $V' \subseteq \mathbf{V}(G_\Gamma)$.

Définition 7.6 (Extension de l'algorithme $\mathcal{A}_{\text{OP}}$). Soient $(G_\Gamma, \mathcal{F} = \{f_1, \ldots, f_n\})$ une instance du problème MRSO, $V' \subseteq \mathbf{V}(G_\Gamma)$ un ensemble de sommets de G_Γ et $\phi : V' \to \Sigma^3$ une affectation des codons pour V'. L'ensemble de fonctions de similarité correspondant à l'affectation ϕ, noté $\mathcal{F}_\phi = \{f_1^\phi, \ldots, f_n^\phi\}$, est défini comme suit:

1. pour tout i tel que $v_i \in \mathbf{V}(G_\Gamma) - V'$, $f_i^\phi = f_i$;
2. pour tout i tel que $v_i \in V'$, $f_i^\phi(\phi(v_i)) = f_i(\phi(v_i))$, $f_i^\phi(C_i) = -\infty$ où $C_i \neq \phi(v_i)$.

L'algorithme $\mathcal{A}_{\text{NEB}}$ (décrit ci-après) correspond à une recherche exhaustive parmi toutes les affectations possibles de codons aux sommets incidents aux arêtes de $\mathcal{E}_b$. Pour chacune de ces affectations, l'algorithme $\mathcal{A}_{\text{NEB}}$ commence par vérifier que l'affectation est compatible avec le graphe $G_\Gamma[\mathcal{E}_b]$, auquel cas, il fait appel à l'algorithme $\mathcal{A}_{\text{OP}}$ avec les fonctions de similarité correspondant à cette affectation. Finalement, l'algorithme $\mathcal{A}_{\text{NEB}}$ renvoie la meilleure solution parmi celles retournées par l'algorithme $\mathcal{A}_{\text{OP}}$.

Algorithme 4: Algorithme $\mathcal{A}_{\text{NEB}}$.

Données : Un graphe de structure induit G_Γ possédant n sommets, un ensemble de fonctions de similarité $\mathcal{F} = \{f_1, \ldots, f_n\}$ et une bipartition propre $\mathcal{P} = \{\mathcal{E}_t, \mathcal{E}_b\}$ des arêtes de G_Γ.

Résultat : Une séquence d'ARNm optimale $T = t_1 t_2 \ldots t_{3n}$ compatible avec G_Γ.

début

  pour chaque *affectation de codons possible ϕ aux sommets incidents aux arêtes de $\mathcal{E}_b$* **faire**

    si *ϕ est compatible avec $G_\Gamma[\mathcal{E}_b]$* **alors**

      (a) Construire l'ensemble $\mathcal{F}_\phi$ de fonctions de similarité correspondant à ϕ.

      (b) Invoquer l'algorithme $A_{\text{OP}}(G_\Gamma[\mathcal{E}_t], \mathcal{F}_\phi)$ pour trouver une séquence d'ARNm optimale $T = t_1 t_2 \ldots t_{3n}$ compatible avec G_Γ.

  retourner *la séquence T ayant la plus grande valeur de similarité.*

fin

Lemme 7.1. *Étant données une instance $(G_\Gamma, \mathcal{F} = \{f_1, \ldots, f_n\})$ pour le problème* MRSO *ainsi qu'une bipartition propre $\mathcal{P} = (\mathcal{E}_t, \mathcal{E}_b)$ des arêtes de G_Γ, l'algorithme $\mathcal{A}_{NEB}$ calcule la séquence d'ARNm optimale pour cette instance et a une complexité en temps $\mathbf{O}(64^{2\epsilon}n)$ avec $\epsilon = |\mathcal{E}_b|$.*

Preuve. Considérons l'Algorithme 4. La compatibilité avec $G_\Gamma[\mathcal{E}_b]$ de toute affectation énumérée étant vérifiée, toute solution retournée par l'algorithme $\mathcal{A}_{\text{NEB}}$ avec un score supérieur à $-\infty$ est faisable. De plus, l'algorithme $\mathcal{A}_{\text{OP}}$ étant optimal, notre solution l'est aussi. Considérons, maintenant, tout sommet de G_Γ. Le nombre d'affectations de codons possibles pour ce sommet est $|\Sigma^3| = 64$. Donc, le nombre d'affectations énumérées est inférieur ou égal à $64^{2\epsilon}$. De plus, la construction d'une telle affectation et la vérification de sa compatibilité avec $G_\Gamma[\mathcal{E}_b]$ peuvent être effectuées en temps $\mathbf{O}(n)$. Les étapes (a) et (b) pouvant être effectuées, également, en temps $\mathbf{O}(n)$, la complexité en temps de l'algorithme $\mathcal{A}_{\text{NEB}}$ est donc bornée par $\mathbf{O}(64^{2\epsilon}(n))$; ce qui prouve le lemme. □

Considérons maintenant nos deux paramètres δ et χ, en commençant par δ. Soient $(G_\Gamma, \mathcal{F})$ une instance du problème MRSO, et $V' = \{v \in \mathbf{V}(G_\Gamma) | d(v) = 3\}$ l'ensemble des sommets de degré trois de G_Γ. Par définition, $\delta = |V'|$. Par la suite, nous démontrons qu'une bipartition propre des arêtes de G_Γ ayant au plus δ arêtes dans $\mathcal{E}_b$ peut facilement être obtenue. Considèrons le lemme connu suivant.

Lemme 7.2. *Si G est un graphe dont les sommets sont au plus de degré deux alors G est planaire extérieur.*

Étant donnée une bipartition des arêtes de G_Γ telle que pour tout $v \in V'$, exactement une arête incidente à v appartient à $\mathcal{E}_b$. Clairement, une telle bipartition avec δ arêtes dans $\mathcal{E}_b$ existe et peut être trouvée en temps linéaire. Soit $\mathcal{P} = \{\mathcal{E}_t, \mathcal{E}_b\}$ une bipartition des arêtes obtenue de cette manière. Le graphe G_Γ étant cubique, tout sommet de G_Γ est incident à au plus deux arêtes de $\mathcal{P}$. Par conséquent, par le Lemme 7.2, $G[\mathcal{E}_t]$ est planaire extérieur et on peut trouver en temps linéaire, un plongement linéaire de G_Γ tel que la bipartition $\mathcal{P}$ soit propre.

Proposition 7.1. *Le problème* MRSO *appartient à la classe* **FPT** *pour le paramètre $\delta = |\{v \in \mathbf{V}(G_\Gamma) : d(v) = 3\}|$.*

Preuve. Nous venons de mentionner que le graphe G_Γ possède une bipartition propre de ses arêtes avec au plus δ arêtes dans $\mathcal{E}_b$ et que cette partition peut être trouvée en temps linéaire. Par conséquent, par le Lemme 7.1, l'algorithme $\mathcal{A}_{\text{NEB}}$ peut être utilisé pour trouver une solution optimale. La complexité en temps de cet algorithme étant $\mathbf{O}(64^{2\delta}n)$, la proposition est prouvée. □

Nous considérons maintenant le paramètre χ. Rappelons que χ représente le nombre d'arêtes se croisant dans G_Γ. Comme pour le paramètre δ, une bipartition propre des arêtes avec χ arêtes dans $\mathcal{E}_b$ peut être trouvée. Il suffit de considérer une bipartition des arêtes ayant une arête dans $\mathcal{E}_b$ pour chaque couple d'arêtes se croisant dans G_Γ. Clairement une telle bipartition est propre et a χ arêtes dans $\mathcal{E}_b$. De plus, cette bipartition peut être obtenue en temps linéaire.

Proposition 7.2. *Le problème* MRSO *appartient à la classe* **FPT** *pour le paramètre* $\chi = |\{(e_1, e_2) : e_1, e_2 \in \mathbf{E}(G_\Gamma) \ \wedge \ e_1 \bowtie e_2\}|$.

Preuve. Le graphe G_Γ possède une bipartition propre de ses arêtes avec au plus χ arêtes dans $\mathcal{E}_b$ et cette partition peut être trouvée en temps linéaire. Par conséquent, par le Lemme 7.1, l'algorithme $\mathcal{A}_{\text{NEB}}$ peut être utilisé pour trouver une solution optimale. La complexité en temps de cet algorithme étant $\mathbf{O}(64^{2\chi}n)$, la proposition est prouvée. □

7.4 NP-complétude d'un cas restreint du problème MRSO

Dans cette section, nous tentons de déterminer plus précisément ce qui rend le problème MRSO difficile. D'après les résultats de Backofen *et al.*, le problème MRSO est **NP**-complet [3] dans le cas général, mais nous savons également que dans certains cas le problème est polynomial. En effet, lorsque le graphe de structure induit est planaire extérieur, Backofen *et al.* ont démontré que le problème est polynomial.

Il est légitime alors de se demander si c'est toujours le cas lorsque le graphe de structure est, par exemple, seulement planaire. Comme nous le démontrerons, la réponse à cette question est non, même pour des classes restreintes de graphes planaires.

Étant donné un graphe G, on appelle *nombre minimum de pages* de G, le nombre minimum de partitions $\mathcal{P} = \{\mathcal{E}_1, \mathcal{E}_2 \ldots, \mathcal{E}_m\}$ de $\mathbf{E}(G)$ tel que chaque partition induit un graphe planaire extérieur pour un ordonnancement fixe des sommets de G. Clairement, le nombre minimum de pages d'un graphe planaire extérieur est un. Pour les graphes planaires, on sait que ce nombre est borné par quatre et qu'il existe des graphes atteignant cette borne [97]. Par la suite, nous prouverons que le problème MRSO est **NP**-complet même lorsque le nombre minimum de pages du graphe de structure induit est égal à deux. Notre preuve est une extension directe de la preuve de **APX**-complétude proposée dans [27] pour le problème MRSO.

Proposition 7.3. *Le problème* MRSO *est* **NP***-complet lorsque le graphe de structure induit a un nombre minimum de pages égale à* 2*.*

Preuve. Nous proposons une transformation polynomiale à partir du problème **NP**-complet MIS-3P, d'ores et déjà défini dans la Section 4.2.

Étant donnée une instance du problème MIS-3P (*i.e.* un graphe cubique connexe planaire sans isthme G possédant n sommets), d'après *Lin et al.* [74] il existe un algorithme de complexité en temps linéaire pour trouver un plongement en deux pages de G, et par conséquent, sans perte de généralité, on peut supposer que G est donné sous la forme d'un graphe linéaire ayant un nombre minimum de pages égal à deux. Nous définissons l'instance du problème MRSO correspondante comme suit. Le graphe de structure induit G_Γ est tout simplement le graphe G et l'ensemble des fonctions de similarité $f_i : \Sigma^3 \to \mathbb{Q}$, $1 \leq i \leq n$, est défini comme suit:

$$\text{pour tout } \ 1 \leq i \leq n, \quad f_i(t_{3i-2}, t_{3i-1}, t_{3i}) = \begin{cases} 1 & \text{si } t_{3i-2}\, t_{3i-1}\, t_{3i} = AAA \\ 0 & \text{sinon} \end{cases}$$

Comme dans [27], l'idée de la réduction est d'identifier l'ensemble des sommets qui se sont vu affecter le codon AAA dans une solution de l'instance correspondante du problème MRSO. Cet ensemble correspond, par définition, à un ensemble indépendant de G. L'exactitude de la preuve découle directement du Théorème 3 de [27]. □

7.5 Conclusion et perspectives

Dans le contexte du design d'ARN, nous avons considéré le problème MRSO. Étant donnés un ensemble de n fonctions de similarité $\mathcal{F}$, une séquence S d'ARNm de longueur $3n$ et un graphe de structure Γ avec $\mathbf{V}(\Gamma) = \{u_1, u_2, \ldots, u_{3n}\}$, le problème MRSO consiste à trouver une séquence T d'ARNm compatible avec Γ telle que la similarité selon $\mathcal{F}$ entre S et T est maximisée.

Nous avons prouvé qu'il existe des algorithmes pour résoudre le problème MRSO de complexités paramétrées par le nombre de sommets de degré trois et par le nombre de croisements (répondant ainsi à un problème ouvert posé dans [27]). Nous avons également prouvé que le problème est **NP**-complet pour un sous-ensemble d'instances du problème – raffinant ainsi le résultat obtenu dans [27].

L'étude de la complexité paramétrée du problème MRSO pour des fonctions de similarité quelconques est un problème difficile. C'est pourquoi, nous pensons qu'il serait intéressant de considérer des fonctions de similarité de la forme $f_i : \Sigma^3 \to \mathbb{N}^+ \cup \{-\infty\}$ puisqu'elles permettent de représenter la majorité des informations nécessaires aux applications. Dans ce modèle, la valeur $-\infty$ peut être utilisée pour interdire certain codon (le codon STOP par exemple) à certaines positions de T.

PARTIE III

Calcul de distances intergénomiques en présence de gènes dupliqués

CHAPITRE 8

Présentation générale

Les travaux présentés dans cette partie relèvent du domaine de la *génomique comparative*, qui s'articule, en autres, autour de la comparaison de génomes d'espèces différentes. Dans cette partie, nous nous intéressons donc à l'entité à l'origine de la transcription des molécules d'ARN et de la synthèse des protéines, à savoir le gène. Rappelons qu'un *gène* désigne une unité d'information génétique transmise d'une génération à une autre. Un gène correspond à un fragment d'ADN situé sur un chromosome. L'ensemble des gènes d'un individu constitue son *génome*.

On appelle *évolution* le processus par lequel les organismes se modifient par transformations successives à partir d'autres organismes. On dénombre généralement deux types de transformations:

1. les *mutations ponctuelles* constituées essentiellement des substitutions, insertions et suppressions au niveau de la séquence d'ADN (*i.e.* des nucléotides);
2. les *réarrangements génomiques* qui correspondent à des modifications, à l'échelle des gènes, dans l'organisation des génomes.

Plus formellement, on définit un réarrangement génomique comme étant une transformation modifiant l'ordre des gènes d'un chromosome ou d'un génome. En effet, un génome peut être vu comme l'ensemble des chromosomes d'un individu – chaque chromosome étant représenté par une suite de gènes. Étant donnée cette représentation, on catégorise, généralement, les opérations de réarrangements génomiques en deux groupes:

1. les opérations affectant un unique chromosome et
2. les opérations affectant simultanément deux chromosomes.

En effet, au cours de l'évolution, des échanges de portions de chromosomes (*i.e.* des suites de gènes), apparaissant dans un ou plusieurs chromosomes, entraînent une modification de l'ordre des gènes du génome correspondant. Les opérations de réarrangements considérées au sein d'un même chromosome (illustrées en Figure 8.1) sont généralement:

1. la *suppression* d'une portion de chromosome: une suite de gènes consécutifs est ôtée du chromosome;
2. *l'insertion* d'une portion de chromosome: une suite de gènes consécutifs est ajoutée à un chromosome;
3. la *duplication* d'une portion de chromosome: une suite de gènes consécutifs est copiée et insérée à une position différente du chromosome;
4. *l'inversion* d'une portion de chromosome: l'ordre des gènes d'une suite de gènes consécutifs est inversé suite au détachement d'une portion du chromosome et à son insertion à la même position, après un retournement;
5. la *transposition* d'une portion de chromosome: une suite de gènes consécutifs est déplacée d'une position vers une position différente du chromosome;

6. la *transposition inverse* d'une portion de chromosome: une suite de gènes consécutifs est successivement inversée et entièrement transposée ou transposée et entièrement inversée.

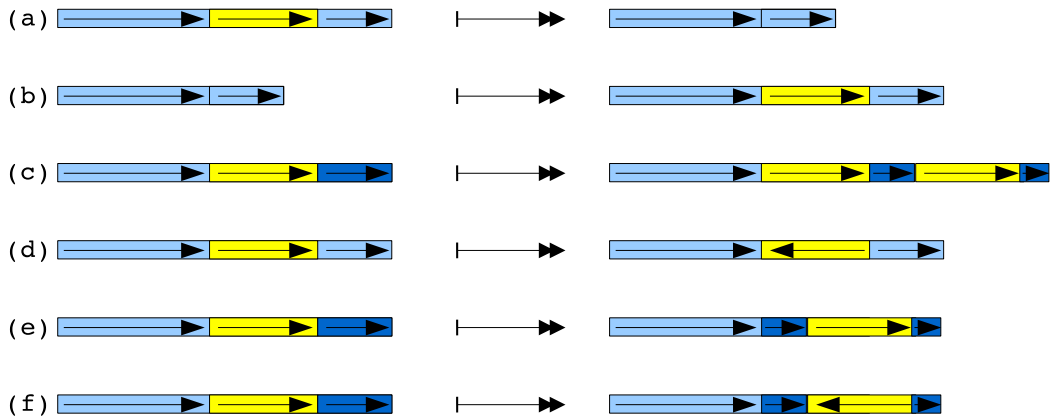


Figure 8.1 – Illustration des opérations de réarrangements affectant un seul chromosome: (a) suppression, (b) insertion, (c) duplication , (d) inversion, (e) transposition et (f) transposition inverse d'une portion de chromosome. Dans cette figure, les flèches sont utilisées pour indiquer l'ordre initial des gènes du chromosome et l'ordre de ces gènes une fois l'opération effectuée.

D'autre part, on considère, généralement, trois opérations de réarrangements entre deux chromosomes (illustrées en Figure 8.2):

1. la *translocation*: deux chromosomes échangent une suite de gènes consécutifs (éventuellement vide pour l'un des deux chromosomes);
2. la *fusion*: deux chromosomes sont réunis en un seul;
3. la *fission*: un chromosome est séparé en deux chromosomes.

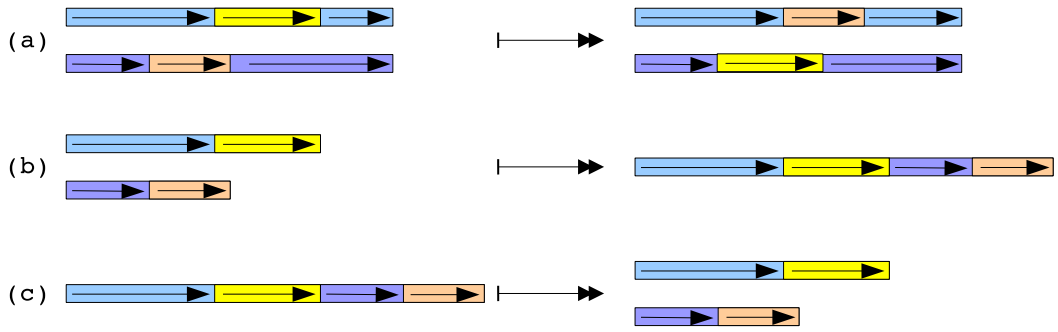


Figure 8.2 – Illustration des opérations de réarrangements affectant deux chromosomes: (a) la translocation, (b) la fusion et (c) la fission.

Dans ce manuscrit nous considérons la modélisation proposée par Sankoff dans [83]: un *génome* G est modélisé par une séquence d'entiers d'un ensemble $\mathcal{F} \subseteq \mathrm{N}$ (appelé également ensemble des *familles de gènes*) telle que chaque entier est signé (*i.e.* pourvu du signe $+$ ou -). Chaque occurrence d'une famille de gènes de $\mathcal{F}$ dans G est appelée *gène*. Les signes symbolisent l'orientation de la portion d'ADN codant pour le gène considéré: le $+$ (*resp.* -) signifiant que la portion d'ADN débute du côté de

l'extrémité terminale $3'$ (*resp.* $5'$) et s'achève du côté terminale $5'$ (*resp.* $3'$). Dans cette représentation l'opération d'inversion, en plus d'inverser l'ordre, inverse également les signes (*i.e.* $+ \rightarrow$ - et - $\rightarrow +$). Il faut noter qu'il existe également une modélisation sans les signes qui est généralement considérée, comme nous le verrons par la suite, pour réduire la difficulté algorithmique des problèmes étudiés.

Étant donnés les génomes de deux organismes A et B, une problème récurrent en biologie est de déterminer un scénario d'évolution entre A et B. On appelle *scénario d'évolution* toute série d'opérations de réarrangements transformant le génome d'un organisme en un autre. La longueur d'un scénario correspond alors au nombre d'événements évolutifs (*i.e.* d'opérations de réarrangements) permettant de transformer un génome en un autre. Cette longueur peut être alors utilisée pour mesurer la *distance d'évolution* existant entre deux organismes. Avant d'illustrer un scénario d'évolution, nous présentons la modélisation *mathématique* des notions de gènes et génomes. Un exemple de scénario d'évolution entre les génomes de deux organismes est donné en Figure 8.3.

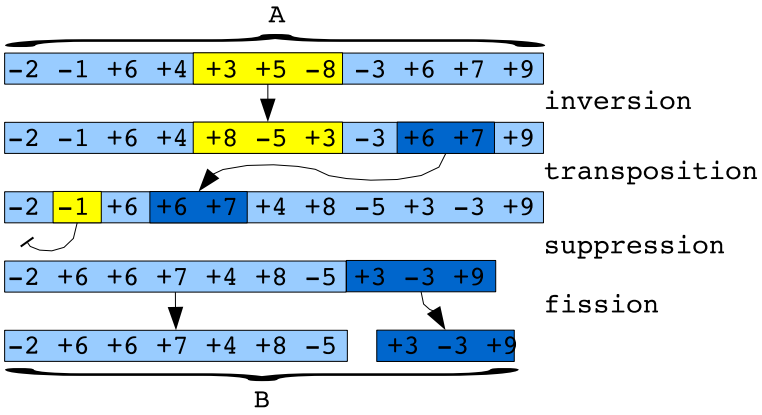


Figure 8.3 – Illustration d'un scénario d'évolution entre les génomes de deux organismes A et B, faisant intervenir une inversion, une transposition, une suppression et une fission. Dans cette exemple, on passe d'un génome composé d'un unique chromosome à un génome composé de deux chromosomes.

Jusqu'à récemment, la plupart des méthodes associées à l'étude de l'ordre des gènes était basée sur la supposition que, dans un génome, il y a au plus une occurrence de toute famille de gènes. Cette supposition permet de représenter les génomes par des permutations (notion définie ci-après) et ainsi de faciliter la résolution des problèmes associés. Malheureusement, il est avéré que cette supposition est très restrictive et seulement justifiée dans le cas de petits génomes de virus, c'est pourquoi la prise en compte de génomes pourvus de plusieurs occurrences d'une même famille de gènes est nécessaire. Dans ce manuscrit, nous définissons une permutation comme suit.

Définition 8.1 (Permutation). Une *permutation* est une séquence ordonnée d'entiers signés distincts deux à deux.

De plus, on appelle *permutation identité de longueur n*, la permutation composée dans l'ordre des entiers positifs de 1 à n. D'après la définition il est évident qu'une permutation ne permet pas de représenter les génomes pourvus de plusieurs occurrences d'une même famille de gènes.

Généralement, l'étude de l'ordre des gènes entre plusieurs génomes repose sur l'évaluation d'une distance d'évolution entre les génomes. L'approche couramment utilisée consiste alors à comparer deux à deux les génomes étudiés. Suivant le principe de parcimonie, les distances d'évolution correspondent

au nombre minimum d'opérations permettant de transformer un génome en un autre. Dans ce manuscrit, nous allons étudier quatre distances d'évolution appliquées aux deux représentations usuelles des génomes: les permutations et les séquences signées. Nous présenterons les résultats obtenus pour le problème du calcul de ces quatre distances avec ou sans la prise en compte de plusieurs occurrences d'une même famille de gènes.

8.1 Distances et permutations

Dans ce manuscrit, nous allons étudier quatre distances basées sur la notion centrale d'adjacence conservée de gènes. Étant donné un génome représenté par une permutation P, deux gènes g et h de P sont *adjacents*, noté $g.h$, s'ils sont consécutifs dans P – *i.e.* $\exists\, 1 \leq i < |P|$, tel que le $i^{ème}$ (*resp.* $i+1^{ème}$) entier de P est g (*resp.* h).

Définition 8.2 (Adjacence conservée). Soient P_1 et P_2 deux permutations et g et h deux gènes adjacents de P_1. L'adjacence $g.h$ est conservée entre P_1 et P_2 si soit $g.h$, soit -h.-g existe dans P_2.

Nous définissons trois autres notions basées sur la définition d'adjacence conservée: les breakpoints [28], les intervalles communs [8, 91] et les intervalles conservés [9].

Définition 8.3 (Breakpoint). Soient P_1 et P_2 deux permutations et g et h deux gènes adjacents de P_1. Un *breakpoint* est une adjacence $g.h$ de P_1 qui n'est pas conservée dans P_2.

Définition 8.4 (Intervalle commun). Soient P_1 et P_2 deux permutations. Un *intervalle commun* entre P_1 et P_2 est une suite consécutive de gènes de P_1 qui, au signe et à l'ordre près, est aussi une suite consécutive de gènes de P_2.

Par la suite, pour toute permutation P_1 et deux gènes g et h tels que g précède h dans P_1, on note $P_1[g, h]$, la suite consécutive des gènes de P_1 situés entre la position de g et celle de h.

Définition 8.5 (Intervalle conservé). Soient P_1 et P_2 deux permutations. Un *intervalle conservé* entre P_1 et P_2 est une suite consécutive de gènes $P_1[g, h]$ telle que:

1. g apparaît avant h ou -h apparaît avant -g dans P_2 et
2. l'ensemble des gènes apparaissant entre g et h est, aux signes près, le même pour P_1 et P_2.

Il est clair que tout intervalle conservé est un intervalle commun. En revanche, la réciproque est fausse. Ces trois notions sont illustrées sur un exemple ci-après.

Exemple 8.2. *Soit* $P_1 =$ -2 -1 $+4$ $+3$ $+5$ -8 $+6$ $+7$ *et* $P_2 =$ -2 -5 -3 -4 $+1$ -8 $+6$ $+7$ *deux permutations.*

- *Les adjacences* -2.-1 *et* $+5$.-8 *sont des breakpoints entre* P_1 *et* P_2.
- P_1[-2, $+7$], P_1[-1, -8] *et* P_1[-8, $+7$], *entre autres, sont des intervalles communs entre* P_1 *et* P_2.
- P_1[-2, $+7$], P_1[$+4$, $+5$] *et* P_1[$+6$, $+7$], *entre autres, sont des intervalles conservés entre* P_1 *et* P_2.

Dans ce manuscrit, nous nous intéressons aux problèmes de calcul du nombre d'inversions, de breakpoints, d'intervalles communs et d'intervalles conservés entre deux permutations. En effet, chacune de ces notions est utilisée pour définir une (voire plusieurs) distances d'évolution.

Dans le cadre de l'étude de la notion d'inversion,le problème SORTING BY REVERSALS, défini ci-après, est un problème majeur.

SORTING BY REVERSALS
DONNÉES*: Deux permutations P_1 et P_2.*
QUESTION*: Trouver un scénario d'évolution pour passer de P_1 à P_2, de longueur minimale et n'impliquant que des inversions.*

La longueur de ce scénario correspond à la distance dite *d'inversion*. Dans [12, 30], la **NP**-complétude du problème SORTING BY REVERSALS appliqué à des permutations non-signées a été démontrée. De plus, ce problème est également **MaxSNP**-dur [30]. Finalement, l'approximation de ce problème a été largement étudiée. On peut citer, entre autres, des algorithmes de 2-approximation [69], de 1.75-approximation [6], de 1.5-approximation [35] et plus récemment de 1.375-approximation [11].

En revanche, ce problème appliqué à des permutations signées peut être résolu de façon exacte par de nombreux algorithmes polynomiaux [10, 67, 88]. Par la suite, n représentera la taille des permutations. Dans [10], Berman *et al.* ont proposé un algorithme de complexité en temps $\mathbf{O}(n^2\alpha(n))$ où α est la fonction inverse d'Ackerman (*cf.* [10] pour plus de détails). Dans [67], Kaplan *et al.* ont proposé un algorithme de complexité en temps $\mathbf{O}(n^2)$. Récemment, la complexité de ce problème a encore été diminuée dans [88]: Tannier *et. al.* ont proposé un algorithme de complexité en temps $\mathbf{O}(n\sqrt{n\log n})$.

En ce qui concerne les trois autre notions, le problème étudié correspond au calcul d'une distance faisant appel aux nombres de breakpoints, d'intervalles communs ou d'intervalles conservés. Il suffit donc de déterminer la complexité du calcul de ces nombres. Dans le cas des breakpoints, un algorithme naïf parcourant les deux permutations a une complexité en temps quadratique, en revanche si on considère que l'une des deux permutations est la permutation identité alors il est possible de calculer le nombre de breakpoints entre une permutation et la permutation identité en $\mathbf{O}(n)$.

De même, dans le cas des intervalles communs, un algorithme naïf peut calculer le nombre d'intervalles communs en temps quadratique. Il existe des algorithmes plus performants, comme dans [91], où Uno *et al.* ont proposé un algorithme de complexité en temps $\mathbf{O}(n+K)$ où $K \leq \binom{n}{2}$ est le nombre d'intervalles communs.

Finalement, dans [9], Bergeron *et al.* ont proposé un algorithme de complexité linéaire pour calculer les intervalles conservés entre deux permutations signées. Comme nous l'avons précisé précédemment, bien que l'utilisation de permutations pour représenter les génomes conduit en général à l'élaboration d'algorithmes polynomiaux, cette représentation ne permet pas de prendre en compte un phénomène répandu dans les génomes: la présence de plusieurs occurrences d'une même famille de gènes. Malheureusement, comme nous allons le montrer dans cette partie, la prise en compte de ce phénomène rend difficile la majorité des problèmes définis précédemment.

8.2 Distances et séquences d'entiers signés

Étant donnés un génome G et une famille de gènes f, le nombre d'occurrences de f dans G est appelé *cardinalité* de f. Une famille de gène est dite *dupliquée* si la cardinalité de f est supérieure ou égale à deux. Dans le cas contraire, f est dite *non-dupliquée*. Tout gène appartenant à une famille dupliquée (*resp.* non-dupliquée) de gènes est un gène dit *dupliqué* (*resp. non-dupliqué*). Deux génomes G et H sont *équilibrés* si, pour toute famille de gènes f, la cardinalité de f dans G et dans H est identique.

Lors de la comparaison de génomes contenant des gènes dupliqués, l'approche généralement adoptée est de se ramener à un problème sur des permutations signées. Pour ce faire, principalement deux stratégies ont été définies: l'exemplarisation et l'utilisation d'un couplage.

La stratégie *d'exemplarisation* fut introduite par Sankoff dans [83]. Étant donnée une séquence signée S représentant un génome, on appelle *séquence exemplaire*, ou *exemplarisation*, de S toute séquence T obtenue à partir de S en supprimant toutes les occurrences, exceptée une, de chaque famille dupliquée de gènes. Clairement, toute séquence exemplaire peut être représentée par une permutation.

Étant données deux séquences signées S_1 et S_2, le problème de calcul d'une distance entre S_1 et S_2 revient à rechercher les séquences exemplaires S'_1, S'_2 de S_1 et S_2 telles que la distance entre S'_1 et S'_2 soit minimisée. Une fois les séquences exemplaires trouvées, il est possible d'utiliser les algorithmes définis sur les permutations pour le calcul de la distance correspondante.

Marron *et al.*, pour leur part, ont proposé dans [86] une stratégie reposant sur l'élaboration d'un *couplage* entre les gènes des deux génomes (illustré en Figure 8.4).

Définition 8.6 (Couplage et couplage maximal entre gènes). Soient $G = g_1 g_2 \dots g_n$ et $H = h_1 h_2 \dots h_m$ deux séquences d'entiers signés. Un *couplage $\mathcal{M}$ entre les gènes* de G et H est un ensemble de couples (g_i, h_j) tel que $g_i \in G$, $h_j \in H$ et $|g_i| = |h_j|$ (*i.e.* les entiers g_i et h_j sont égaux au signe près). Un couplage $\mathcal{M}$ est *maximal* si pour toute famille de gènes f, il n'existe pas d'occurrence de f n'appartenant pas à $\mathcal{M}$ et apparaissant simultanément dans G et dans H.

Il découle de la définition de couplage maximal qu'étant donné un couplage maximal $\mathcal{M}$ entre deux génomes équilibrés G et H, tous les gènes de G et H appartiennent à $\mathcal{M}$.

Clairement, étant données deux séquences d'entiers signés S_1 et S_2, le problème de calcul d'une distance entre S_1 et S_2 revient à rechercher un couplage maximal entre les gènes de S_1 et S_2 tel que la distance soit minimisée. Une fois le couplage trouvé, il est, comme dans le cas de l'exemplarisation, possible d'utiliser les algorithmes définis sur les permutations pour le calcul de la distance correspondante. En effet, par un simple renommage des séquences à l'aide du couplage, il est possible de transformer les séquences S_1 et S_2 en permutations.

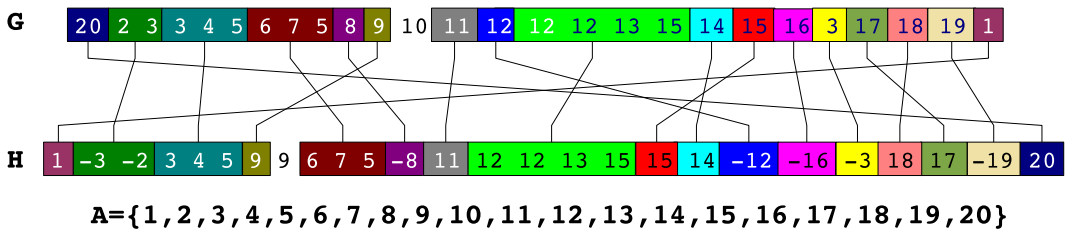


Figure 8.4 – Illustration d'un *couplage $\mathcal{M}$ entre les gènes* de deux génomes non-équilibrés G et H définis sur un ensemble de familles de gènes A.

Nous présentons ci-après les résultats de l'utilisation de ces deux stratégies sur les problèmes présentés dans la section précédente. Dans [71], Kolman a proposé d'étendre le problème SORTING BY REVERSALS de façon à prendre en compte des familles dupliquées de gènes. Pour ce faire il a défini une variante du problème SORTING BY REVERSAL, notée k-SBR, pour laquelle chaque famille dupliquée de gènes de chaque génome est de cardinalité au plus k. Par la suite, nous appelerons *version signée* (*resp. version non-signée*) de k-SBR, le problème appliqué aux séquences d'entiers signés (*resp.* non-signés).

Le sous-problème 1-SBR correspond au problème SORTING BY REVERSALS; sa complexité en découle. La **NP**-complétude de la version signée du sous-problème 2-SBR a été démontrée par Chen *et al.* dans [34]. Ce résultat de complexité reste valide même dans le cas de la version non-signée du sous-problème 2-SBR. De plus, ce résultat implique également la **NP**-complétude du cas général. Il

existe des algorithmes d'approximation de ratio constant pour les versions signées des sous-problèmes 2-SBR et 3-SBR [37, 54, 34]. Le meilleur ratio d'approximation actuel pour le sous-problème 2-SBR est 2.2074 et celui pour le sous-problème 3-SBR est 8 [54]. Finalement, dans [71], Kolman a proposé un algorithme de $\mathbf{O}(k^2)$-approximation pour le problème k-SBR utilisant un algorithme de complexité en temps $\mathbf{O}(k.n)$.

La Table 8.1 présente les résultats connus pour le problème de calcul d'une distance faisant appel aux nombres d'inversion, de breakpoints, d'intervalles communs ou d'intervalles conservés. Comme l'illustre la Table 8.1, la stratégie de couplage n'a pas été étudiée pour la distance de breakpoints. Dans le Chapitre 9, nous prouverons que ce problème est **NP**-complet. D'autre part, la complexité des deux stratégies dans le cas de la distance d'intervalles conservés n'a pas encore été définie. Comme nous le verrons dans le Chapitre 10, le problème est également **NP**-complet dans les deux cas.

Calcul de distances entre des séquences d'entiers signés		
	EXEMPLARISATION	COUPLAGE
INVERSION	**NP**-complet [28]	**NP**-complet [36, 34] $\star$
BREAKPOINTS	**NP**-complet [28]	?
INTERVALLES CONSERVÉS	?	
INTERVALLES COMMUNS	**NP**-complet [33]	

Table 8.1 – Complexité du problème de calcul de distances entre des séquences d'entiers signés en fonction de la stratégie adoptée. $\star$ Chen *et al.* (*resp.* Christie *et al.*) ont démontré la **NP**-complétude de la version signée (*resp.* non-signée) du problème.

Christie *et al.* ont prouvé dans [36] la **NP**-complétude du problème de calcul de la distance d'inversion entre des séquences d'entiers non-signés, même sur un alphabet binaire. Dans [28], Bryant a prouvé, dans le cas de l'exemplarisation, la **NP**-complétude des problèmes de calcul des distances d'inversion et de breakpoints. Bryant a donc complété l'étude de la distance d'inversion. En revanche, la complexité du problème de calcul de la distance de breakpoints, dans le cas de l'utilisation de couplages, est inconnue. Finalement, dans [33], Chauve *et al.* ont prouvé la **NP**-complétude du problème de calcul de la distance d'intervalles communs pour les deux stratégies.

Dans ce chapitre, nous avons présenté deux représentations usuelles des génomes: les permutations (signées ou non) et les séquences signées. Pour chacune de ces représentations nous avons présenté un ensemble de problèmes liés à l'étude de l'ordre des gènes entre plusieurs génomes reposant sur l'évaluation d'une distance d'évolution entre les génomes. Plus précisément, nous avons présenté quatre distances d'évolution appliquées aux deux représentations usuelles des génomes: les distances d'inversion, de breakpoints, d'intervalles communs et d'intervalles conservés.

Globalement, on peut constater que, dans le cas des permutations, dès lors que celles-ci sont signées, le problème de calcul d'une des distances précédemment citées peut être résolu à l'aide d'un algorithme polynomial. Cette représentation ne permet malheureusement pas de prendre en compte un phénomène

répandu dans les génomes: la présence de plusieurs occurrences d'une même famille de gènes. Comme nous l'avons précisé, la prise en compte de ce phénomène (à l'aide des séquences signées) rend difficile le problème de calcul de toutes les distances précédemment citées.

Comme indiqué dans la Table 8.1, il reste des cas ouverts pour les problèmes de calcul des distances de breakpoints et d'intervalles conservés. Dans les Chapitres 9 et 10, nous déterminerons la **NP**-complétude de ces cas ouverts et complétons ainsi l'étude de la complexité de la prise en compte des gènes dupliqués dans le calcul des quatres distances d'évolution précédemment citées.

CHAPITRE 9

Couplage et distance de breakpoints en présence de gènes dupliqués

9.1 Introduction

Dans ce chapitre, nous développons des résultats publiés dans [17] et obtenus en collaboration avec Guillaume Fertin et Cédric Chauve. Comme nous l'avons précisé dans la Section 8.2, l'existence d'un algorithme polynomial pour résoudre le problème du calcul de la distance de breakpoints en utilisant une stratégie de couplage est le dernier cas ouvert pour le problème de calcul de la distance de breakpoints en présence de gènes dupliqués. Le problème se définit comme suit.

MINIMUM BREAKPOINT MATCHING (MBM(L,F))
DONNÉES*: Un entier $\mathcal{P}$ et deux génomes G et H définis sur un ensemble de familles de gènes $\mathcal{F}$ tel qu'il existe au plus* F *familles dupliquées de gènes et où aucun segment dans G ou H (suite de gènes consécutifs) n'est constitué de plus de* L *gènes dupliqués.*
QUESTION*: Existe-il un couplage maximal $\mathcal{M}$ entre les gènes de G et H tel que la distance de breakpoints entre G et H soit inférieure ou égale à $\mathcal{P}$?*

Dans ce chapitre, nous démontrons que ce problème est **NP**-complet, même dans le cas restreint où il n'existe qu'une seule famille dupliquée de gènes. De plus, nous prouverons que le problème MBM(L,1) sur des génomes équilibrés possède un algorithme de (L+1)-approximation.

Calcul de distances entre des séquences d'entiers signés		
	EXEMPLARISATION	COUPLAGE
INVERSION	**NP**-complet [28]	**NP**-complet [36, 34]
BREAKPOINTS	**NP**-complet [28]	**NP-complet [17]**
INTERVALLES CONSERVÉS	?	
INTERVALLES COMMUNS	**NP**-complet [33]	

Table 9.1 – Complexité du problème de calcul de distances entre des séquences d'entiers signés en fonction de la stratégie adoptée.

Dans le reste de ce chapitre nous utiliserons les notations suivantes.

Définition 9.1 (Couplage minimisant une distance entre deux génomes). Soient $G = g_1 g_2 \dots g_n$ et $H = h_1 h_2 \dots h_m$ deux séquences d'entiers signés. Un couplage $\mathcal{M}$ entre les gènes de G et H *minimise une distance* d entre G et H si pour tout couplage $\mathcal{M}'$ entre les gènes de G et de H la distance entre G et H induite par $\mathcal{M}'$ est supérieure ou égale à celle induite par $\mathcal{M}$.

Par la suite, nous notons $D_b(G, H, \mathcal{M})$ le nombre de breakpoints entre deux génomes G et H étant donné un couplage $\mathcal{M}$ entre les gènes de G et H. La distance de breakpoints entre G et H est notée $D_b(G, H)$. Formellement, $D_b(G, H) = D_b(G, H, \mathcal{M})$ où $\mathcal{M}$ est un couplage minimisant la distance de breakpoints.

D'autre part, un *segment de gènes* représente toute suite de gènes consécutifs. Étant donné un segment de gènes $S = g_1, \dots, g_k$, on note $\overline{S}$ = -$g_k, \dots,$ -g_1.

Dans la suite de ce chapitre, nous présentons des résultats de complexité pour le problème du calcul de la distance de breakpoints, utilisant une stratégie de couplage, dans le cas où il n'y a qu'une seule famille dupliquée de gènes. Le premier des deux résultats de ce chapitre est la **NP**-complétude du calcul de la distance de breakpoints en présence d'une unique famille dupliquée de gènes (Théorème 9.1). Ce résultat implique que, dans sa généralité (*i.e.* plus d'une famille dupliquée de gènes), le problème est également **NP**-complet. Il implique également que le problème MINIMUM COVER introduit par Swenson *et al.* [86] est **NP**-complet (Théorème 9.2). Le second résultat de ce chapitre est un algorithme d'approximation pour le calcul, dans un cas restreint, de la distance de breakpoints avec un ratio d'approximation dépendant de la taille de la plus grande suite de gènes dupliqués (Théorème 9.3).

9.2 NP-complétude du calcul de la distance de breakpoints

Théorème 9.1. *Le problème* MBM(L,1) *is* **NP***-complet.*

Notre preuve repose sur une transformation polynomiale du problème de décision **NP**-complet MINIMUM BIN PACKING [53] vers le problème de décision MINIMUM BREAKPOINT MATCHING.

> MINIMUM BIN PACKING
> DONNÉES*: Un ensemble fini* $U = \{u_1, u_2, \dots, u_n\}$*, une taille* $s(u) \in \mathbb{Z}^+$ *pour chaque* $u \in U$ *et deux entiers positifs* $\mathcal{K}$ *et* $\mathcal{C}$*.*
> QUESTION*: Existe-il une partition de* U *en* $\mathcal{K}$ *ensembles disjoints* $U_1, U_2, \dots, U_{\mathcal{K}}$ *telle que la somme des tailles des éléments de chaque* U_i *est inférieure ou égale à* $\mathcal{C}$ *?*

Dans cette section, nous considérons la présence d'une unique famille dupliquée de gènes; par conséquent nous nous intéressons au problème MBM(L,1). Il est facile de constater que le sous-problème MBM(L,1) appartient à la classe **NP**. En effet, étant donnés un entier $\mathcal{P}$ et un couplage maximal entre les gènes de deux génomes, il est possible de calculer, en temps polynomial, la distance de breakpoints et vérifier si cette dernière est inférieure ou égale à $\mathcal{P}$. Le reste de cette section est consacrée à prouver que le sous-problème MBM(L,1) est également **NP**-dur, induisant ainsi le Théorème 9.1.

Nous débutons cette preuve en détaillant la construction des génomes G et H du problème MBM(L,1) à partir d'une instance $(U, \mathcal{K}, \mathcal{C})$ du problème MINIMUM BIN PACKING. Les familles de gènes sont $\{\alpha, \beta, \mathbf{x}, A_1, A_2 \dots, A_{n+\mathcal{N}}, B_1, B_2, \dots, B_{\mathcal{K}+1}\}$[1]. En tout, il y a $\mathcal{K} + \mathcal{N} + n + 4$ familles de gènes, et $\mathbf{x}$ sera l'unique famille dupliquée de gènes. Pour $1 \leq i \leq n$ (*resp.* $n + 1 \leq i \leq n + \mathcal{N}$), $\mathbf{u_i'}$ représente

[1] Par mesure de lisibilité, nous utilisons dans cette preuve des symboles plutôt que des entiers.

une séquence de $s(u_i)$ (*resp.* un) gènes $\mathbf{x}$ consécutifs. Pour $1 \leq j \leq k$, $\mathbf{U'_j}$ représente une séquence de $\mathcal{C}$ gènes $\mathbf{x}$ consécutifs. Nous définissons les génomes G et H comme suit:

$$G = \alpha\ \mathbf{u'_1}\ A_1\ \mathbf{u'_2}\ A_2\ \ldots\ \mathbf{u'_{n+\mathcal{N}}}\ A_{n+\mathcal{N}}\ B_1\ B_2\ \ldots\ B_{\mathcal{K}+1}\ \beta$$
$$H = \alpha\ B_1\ \mathbf{U'_1}\ B_2\ \mathbf{U'_2}\ \ldots\ B_{\mathcal{K}}\ \mathbf{U'_{\mathcal{K}}}\ B_{\mathcal{K}+1}\ A_1\ A_2\ \ldots\ A_{n+\mathcal{N}}\ \beta$$

Une illustration d'une construction de ce type – qui peut être effectuée en temps polynomial – est donnée en Figure 9.1. Afin de compléter la construction de l'instance du sous-problème MBM(L,1), nous définissons formellement $\mathcal{P}$: $\mathcal{P} = 2(n + \mathcal{N} + 1) + \mathcal{K}$.

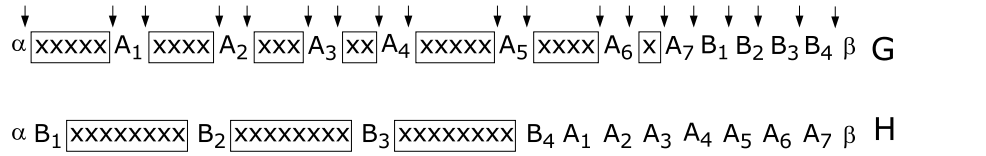


Figure 9.1 – Instance du sous-problème MBM(L,1) associée à l'instance $(U, \mathcal{K}, \mathcal{C})$ du problème MINIMUM BIN PACKING avec $\mathcal{K} = 3$, $\mathcal{C} = 8$ et $U = \{u_1, \ldots, u_6\}$ telle que $s(u_1) = s(u_5) = 5$, $s(u_2) = s(u_6) = 4$, $s(u_3) = 3$ et $s(u_4) = 2$. Les flèches indiquent les breakpoints induis par tout couplage entre G et H.

Dans les deux prochains lemmes, nous considérons une instance $(U, \mathcal{K}, \mathcal{C})$ du problème MINIMUM BIN PACKING et l'instance correspondante, d'après la construction décrite ci-dessus, $(G, H, \mathcal{P})$ du sous-problème MBM(L,1).

Lemme 9.1. *$D_b(G, H) \geq \mathcal{P}$, et, dans tout couplage de G et H minimisant le nombre de breakpoints entre G et H, au moins $\mathcal{P}$ breakpoints impliquent un ou deux gènes non-dupliqués.*

Preuve. Comptons le nombre de breakpoints dans G apparaissant entre deux gènes adjacents et tels que au moins l'un des deux est non-dupliqué. Pour un gène g du génome G, soit $L^G(g)$ (*resp.* $R^G(g)$) le gène situé à la gauche (*resp.* la droite) du gène g dans le génome G. Pour tout gène A_i tel que $1 \leq i \leq n+\mathcal{N}$, $L^G(A_i) \neq L^H(A_i)$ et $R^G(A_i) \neq R^H(A_i)$. Par conséquent, pour tout $1 \leq i \leq n + \mathcal{N}$, le gène A_i induit deux breakpoints (un avec $L^G(A_i)$ et un avec $R^G(A_i)$). De façon similaire, pour tout gène B_j tel que $1 \leq j \leq \mathcal{K}$, $R^G(B_j) \neq R^H(B_j)$. Donc, pour tout $1 \leq j \leq \mathcal{K}$, le gène B_j induit un breakpoint. Jusqu'à présent, nous avons dénombré $2n + 2\mathcal{N} + \mathcal{K}$ breakpoints dans G.

Finalement, $R^G(\alpha) \neq R^H(\alpha)$ et $L^G(\beta) \neq L^H(\beta)$. Par conséquent, en tout le nombre de breakpoints dans G est au moins $2(n + \mathcal{N} + 1) + \mathcal{K} = \mathcal{P}$, et ces $\mathcal{P}$ breakpoints, que nous avons décrit ci-dessus, impliquent tous au moins un gène non-dupliqué. □

Lemme 9.2. *Il existe une partition de U en $\mathcal{K}$ ensembles disjoints $U_1, U_2, \ldots, U_{\mathcal{K}}$ telle que la somme des tailles des éléments de chaque U_i est inférieure ou égale à $\mathcal{C}$ si et seulement si $D_b(G, H) \leq \mathcal{P}$.*

Preuve. ($\Leftarrow$) Supposons que $D_b(G, H) \leq \mathcal{P}$. D'après le Lemme 9.1, nous savons que $D_b(G, H) = \mathcal{P}$, et que dans un couplage entre G et H minimisant le nombre de breakpoints, tous les breakpoints apparaissent entre deux gènes adjacents, tels qu'au moins un des deux est non-dupliqué. En d'autres termes, dans un couplage entre G et H minimisant le nombre de breakpoints, toute paire de gènes $\mathbf{x}$ adjacents dans G est couplée à une paire de gènes $\mathbf{x}$ adjacents dans H.

Par conséquent, pour tout $1 \leq i \leq n$, le segment de gènes dupliqués u'_i de G est couplé avec un segment de gènes $\mathbf{x}$ de H. Plus précisément, pour tout $1 \leq i \leq n$, il existe un entier $1 \leq j \leq \mathcal{K}$ tel que la séquence u'_i est entièrement couplée avec une sous-séquence de U'_j, comme illustré en Figure 9.2.

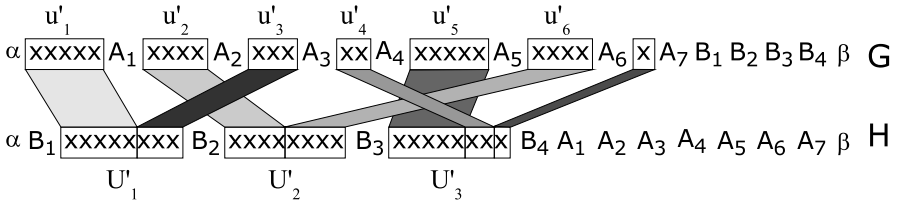


Figure 9.2 – Couplage entre les gènes de G et H correspondant à la partition suivante de U : $U_1 = \{u_1, u_3\}$, $U_2 = \{u_2, u_6\}$ et $U_3 = \{u_4, u_5\}$

Par conséquent, un tel couplage induit une partition de l'ensemble des séquences $\{u'_1, u'_2, \ldots, u'_n\}$ en au plus $\mathcal{K}$ ensembles disjoints $U'_1, U'_2, \ldots, U'_{\mathcal{K}}$. Étant donné que, par construction, $|U'_i| = \mathcal{C}$ pour tout $1 \leq i \leq \mathcal{K}$, cette partition correspond à une solution pour l'instance correspondante $(U, \mathcal{K}, \mathcal{C})$ du problème MINIMUM BIN PACKING.

($\Rightarrow$) Supposons que nous connaissons une partition de U en $\mathcal{K}$ ensembles disjoints $U_1, U_2, \ldots, U_{\mathcal{K}}$, chacun de cardinalité au plus $\mathcal{C}$. Nous construisons un couplage $\mathcal{M}$ entre les gènes de G et H comme suit.

- Chaque gène non-dupliqué de G est couplé, dans $\mathcal{M}$, à son unique copie de H;
- Pour tout $1 \leq j \leq \mathcal{K}$ et $1 \leq i \leq n$, si $u_i \in U_j$ alors la séquence de gènes $\mathbf{x}$ de u'_i dans G est couplée gène à gène avec la première séquence de gènes libres (*i.e.* non-couplés) $\mathbf{x}$ de U'_j dans H.

Le couplage $\mathcal{M}$ étant construit d'après la partition de U, nous opuvons affirmer que, pour tout $1 \leq i \leq n$, le segment de gènes dupliqués u'_i est couplé avec une séquence de gènes $\mathbf{x}$ consécutifs de H. Par conséquent, tous les breakpoints induits par ce couplage apparaissent entre deux gènes adjacents, tels qu'au moins un de ces deux gènes est non-dupliqué. D'après le Lemme 9.1, on en déduit que $D_b(G, H, \mathcal{M}) \leq \mathcal{P}$, et donc que $D_b(G, H) \leq \mathcal{P}$. Le lemme et le Théorème 9.1 sont prouvés. □

À l'aide du résultat du Théorème 9.1, nous pouvons répondre à une question introduite par Swenson *et al.* dans [86]. Dans [86], Swenson *et al.* considérent le problème du calcul d'une distance d'évolution entre deux génomes arbitraires (*i.e.* contenant des gènes dupliqués). Les opérations autorisées sont l'inversion, la suppression et l'insertion de segments de gènes.

Swenson *et al.* ont défini la notion de *cover* comme étant un couplage entre des segments identiques, au renversement près, de gènes d'un génome *source* et d'un génome *cible*. Swenson *et al.* ont également défini la notion de *minimum cover* comme étant un cover contenant le plus petit nombre de segments couplés. Ils ont introduit cette notion dans le but d'obtenir une bonne approximation de la distance d'édition (en termes d'inversion, insertions et suppressions) entre deux génomes. Par la suite, nous prouverons que trouver un *minimum cover* entre deux génomes contenant des gènes dupliqués est un problème **NP**-complet. Notre preuve repose sur une transformation polynomiale à partir du problème MBM(L,F) sur deux génomes équilibrés. Nous définissons formellement, ci-après, le problème de décision MINIMUM COVER tel que présenté dans [86].

MINIMUM COVER
DONNÉES*: Deux génomes G et H et un entier $\mathcal{Q}$.*
QUESTION*: Existe-il un cover de G et H contenant au plus $\mathcal{Q}$ segments?*

Théorème 9.2. *Le problème* MINIMUM COVER *est* **NP***-complet.*

Preuve. Il est facile de constater que le problème MINIMUM COVER est équivalent au problème MBM(L,F) avec $\mathcal{Q} = \mathcal{P} + 1$. □

9.3 Approximation du problème MBM(L,F)

Afin de contourner la difficulté du problème MBM(L,F), nous proposons un algorithme d'approximation pour le calcul, dans un cas restreint, de la distance de breakpoints avec un ratio d'approximation dépendant de la taille de la plus grande suite de gènes dupliqués.

Théorème 9.3. *Soient G et H deux génomes équilibrés tels qu'il existe une unique famille dupliquée de gènes et qu'aucun segment de gènes dupliqués consécutifs ne contient plus de* L *gènes. Le problème* MBM(L,1) *possède un algorithme de $(\mathrm{L}+1)$-approximation.*

Afin de prouver ce résultat, nous introduisons la notion de *copie d'un segment de gènes dupliqués*: la copie d'un segment $\mathbf{a}\ S\ \mathbf{b}$ de G, où $\mathbf{a}$ et $\mathbf{b}$ sont des gènes non-dupliqués et S est un segment de gènes dupliqués, est un segment de H contenant soit $\mathbf{a}\ S\ \mathbf{b}$, soit $\overline{\mathbf{b}}\ \overline{S}\ \overline{\mathbf{a}}$. En d'autres termes, coupler $\mathbf{a}\ S\ \mathbf{b}$ à sa copie dans H n'induit pas de breakpoint impliquant un gène de S. D'autre part, un segment S_1 est dit *parfaitement couplé* à un segment S_2 si $|S_1| = |S_2|$ et soit pour tout $1 \leq i \leq |S_1|$ $(S_1[i], S_2[i]) \in \mathcal{M}$, soit pour tout $1 \leq i \leq |S_1|$ $(S_1[i], S_2[|S_2| - i]) \in \mathcal{M}$.

Lemme 9.3. *Soient G et H deux génomes équilibrés tels qu'il existe une unique famille dupliquée de gènes et qu'aucun segment de gènes dupliqués consécutifs ne contient plus de* L *gènes. Il existe un couplage minimisant la distance de breakpoints entre les gènes de G et H tel que tout segment de gènes dupliqués de G est parfaitement couplé avec sa copie dans H.*

Preuve. Soit $\mathcal{M}$ un couplage minimisant la distance de breakpoints entre deux génomes équilibrés G et H, et $S_1 = \mathbf{a}\ S\ \mathbf{b}$ un segment de gènes dupliqués de G qui n'est pas parfaitement couplé dans $\mathcal{M}$ avec sa copie dans H. Sans perte de généralité, nous supposons que la copie de S_1 dans H est $S_2 = \mathbf{a}\ S\ \mathbf{b}$.

Soit $S_1[i]$ le premier gène de S_1 tel que $(S_1[i], S_2[i]) \notin \mathcal{M}$. Il existe alors deux breakpoints, un à la gauche du gène $S_1[i]$ et un à la gauche du gène de G couplé à $S_2[i]$. Il est immédiat que coupler $S_1[i]$ et $S_2[i]$ dans $\mathcal{M}$ n'induit, dans le pire des cas, pas plus de breakpoints (*cf.* Figure 9.3). Par conséquent, étant donné un couplage minimisant la distance de breakpoints entre deux génomes équilibrés G et H, il est possible de trouver un couplage n'induisant pas plus de breakpoints et où tout segment de gènes dupliqués de G est parfaitement couplé avec sa copie dans H; ce qui prouve le lemme. □

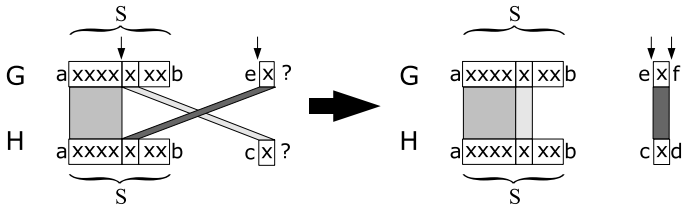


Figure 9.3 – Le couplage parfait des segments de gènes dupliqués de G et de leurs copies dans H n'augmente pas le nombre de breakpoints.

Le lemme suivant découle de la définition d'une copie de segment.

Lemme 9.4. *Soient G et H deux génomes équilibrés tels qu'il existe une unique famille dupliquée de gènes et qu'aucun segment de gènes dupliqués consécutifs ne contient plus de* L *gènes. Soit S un segment de gènes dupliqués de G tel qu'il n'existe pas de copie de S dans T. Alors dans tout couplage entre les gènes de G et H minimisant le nombre de breakpoints, il existe au moins un breakpoint impliquant un gène de S.*

Preuve du Théorème 9.3. Considérons l'Algorithme 5 calculant un couplage entre les gènes de G et H.

Algorithme 5:

Données : Deux génomes équilibrés G et H tels qu'il existe une unique famille dupliquée de gènes et qu'aucun segment de gènes dupliqués consécutifs ne contient plus de L gènes.

Résultat : $D_b(G, H)$.

début

1. Coupler parfaitement dans $\mathcal{M}$ tout segment de gènes dupliqués de G et sa copie dans H.
2. Coupler dans $\mathcal{M}$ chaque gène non-dupliqué de G et son unique copie dans H.
3. Coupler arbitrairement dans $\mathcal{M}$ chaque gène dupliqué restant de G et de H.

retourner $D_b(G, H, \mathcal{M})$

fin

Soit $\mathcal{M}$ le couplage produit par l'Algorithme 5 et $\mathcal{M}_{\text{opt}}$ un couplage optimal dans lequel tout segment de gènes dupliqués de G est couplé à sa copie de H (d'après le Lemme 9.3, un tel couplage existe). Alors, dans $\mathcal{M}$ et dans $\mathcal{M}_{\text{opt}}$, il n'existe pas de breakpoint impliquant deux gènes de G appartenant tous deux à un segment de gènes ayant une copie. D'après la définition d'un couplage, le résultat de l'Algorithme 5 ne contient pas plus de breakpoints impliquant deux gènes non-dupliqués que dans $\mathcal{M}_{\text{opt}}$.

La différence, en termes de nombre de breakpoints, entre les couplages $\mathcal{M}$ et $\mathcal{M}_{\text{opt}}$ ne dépend donc que des segments de gènes dupliqués couplés à l'étape 3 de l'Algorithme 5. Si on considére uniquement ces segments de gènes, étant donné que chaque segment de gènes dupliqués est composé d'au plus L gènes, dans $\mathcal{M}$ ces segments de gènes induisent chacun au plus $\text{L} + 1$ breakpoints. Or, d'après le Lemme 9.4, dans $\mathcal{M}_{\text{opt}}$ il y a au moins un breakpoint induit par chacun de ces segments. La preuve du théorème en découle. □

9.4 Conclusion et perspectives

Dans ce chapitre, nous avons étudié le problème du calcul de la distance de breakpoints entre séquences d'entiers signés en considérant l'utilisation de couplages. Nous avons prouvé que, contrairement aux permutations signées, le calcul de cette distance entre séquences d'entiers signés est **NP**-complet, même dans le cas restreint où il n'existe qu'une seule famille dupliquée de gènes et que les génomes sont équilibrés (*i.e.* les opérations d'insertion et de suppression de gènes sont interdites). Ce résultat est, à l'exception de l'utilisation de l'exemplarisation, le premier résultat de complexité sur ce problème.

Notre étude laisse beaucoup de questions théoriques sans solution. Par exemple, on ne sait pas à partir de quelle valeur du paramètre L le problème du calcul de la distance de breakpoints bascule dans la **NP**-complétude: lorsque $\text{L} = 1$, le problème est équivalent à calculer un couplage de poids minimum dans un graphe pondéré biparti, ce qui peut être fait en temps polynomial [39], en revanche on n'en sait

pas plus pour des valeurs de L plus importantes. De plus, il serait intéressant d'étudier l'approximabilité du problème et de déterminer s'il existe un algorithme d'approximation pour le problème général MBM(L,F) avec un ratio constant.

CHAPITRE 10

La distance d'intervalles conservés en présence de gènes dupliqués

10.1 Introduction

Dans ce chapitre, nous développons des résultats publiés dans [25] et obtenus en collaboration avec Romeo Rizzi. Comme nous l'avons précisé dans la Section 8.2, le problème du calcul de la distance d'intervalles conservés entre deux séquences d'entiers signés n'a pas, jusqu'ici, été étudié. Nous démontrerons que, quelle que soit la stratégie adoptée (exemplarisation ou utilisation de couplages), le problème du calcul de la distance d'intervalles conservés entre deux séquences d'entiers signés est **NP**-complet, même dans des cas restreints. Pour pallier à cette difficulté, nous proposerons une heuristique pour le problème dans le cas de l'utilisation de couplages.

Calcul de distances entre des séquences d'entiers signés		
	EXEMPLARISATION	COUPLAGE
INVERSION	**NP**-complet [28]	**NP**-complet [36, 34]
BREAKPOINTS	**NP**-complet [28]	**NP**-complet [17]
INTERVALLES CONSERVÉS	**NP-complet [25]**	
INTERVALLES COMMUNS	**NP**-complet [33]	

Table 10.1 – Complexité du problème de calcul de distances entre des séquences d'entiers signés en fonction de la stratégie adoptée.

Dans ce chapitre, nous utiliserons la définition formelle de la distance d'intervalles conservés introduite dans [9]. Étant donné un ensemble de génomes $\mathcal{G}$, soit $N_{\mathcal{G}}$ le nombre d'intervalles conservés dans $\mathcal{G}$. Étant donnés deux ensembles de génomes $\mathcal{G}$ et $\mathcal{H}$, la *distance d'intervalles conservés* entre $\mathcal{G}$ et $\mathcal{H}$ est définie par $\mathcal{D}_{ic}(\mathcal{G},\mathcal{H}) = N_{\mathcal{G}} + N_{\mathcal{H}} - 2N_{\mathcal{G}\bigcup\mathcal{H}}$.

Par exemple, soient $\mathcal{G} = \{G_1, G_2\}$ et $\mathcal{H} = \{H_1, H_2\}$ deux ensembles de génomes tels que $G_1 = a\ b\ c\ g\ e\ f$ -$d\ h$, $G_2 = a\ g$ -c -$b\ e$ -f -$d\ h$, $H_1 = a\ e$ -$f\ b\ g\ c$ -$d\ h$ et $H_2 = a\ f$ -c -$g\ b$ -e -$d\ h$. Nous obtenons $\mathcal{D}_{ic}(\mathcal{G},\mathcal{H}) = 7+3-4 = 6$. Dans le reste de ce chapitre, par mesure de lisibilité, nous noterons la distance d'intervalles conservés entre deux singletons $\mathcal{D}_{ic}(\{G\},\{H\})$ par $\mathcal{D}_{ic}(G,H)$.

Par la suite, nous notons $D_{ic}(G,H,\mathcal{M})$ la distance d'intervalles conservés entre deux génomes G et H étant donné un couplage maximal $\mathcal{M}$ entre les gènes de G et H. Formellement, $D_{ic}(G,H) =$

$D_{ic}(G, H, \mathcal{M})$ où $\mathcal{M}$ est un couplage maximal entre les gènes de G et H minimisant la distance d'intervalles conservés.

10.2 Exemplarisation et distance d'intervalles conservés

Dans cette section, nous abordons le problème du calcul de la distance d'intervalles conservés en présence de gènes dupliqués en utilisant la stratégie d'exemplarisation. Nous démontrons que ce problème est **NP**-complet. Nous débutons cette section par la définition formelle de ce problème.

EXEMPLAR CONSERVED INTERVAL DISTANCE (ECID)
DONNÉES*: Deux génomes G et H, et un entier positif k.*
QUESTION*: Existe-t-il deux exemplarisations G' et H' des génomes G et H telles que $\mathcal{D}_{ic}(G', H') \leq k$?*

Théorème 10.1. *Le problème* EXEMPLAR CONSERVED INTERVAL DISTANCE *est* **NP***-complet.*

Afin de prouver le Théorème 10.1, nous proposons une transformation polynomiale à partir du problème **NP**-complet [53] MINIMUM SET COVER. Étant donnés un entier k' et deux génomes exemplaires G' et H', il est possible de calculer en temps polynomial $\mathcal{D}_{ic}(G', H')$ et de vérifier si $\mathcal{D}_{ic}(G', H') \leq k'$ (*cf.* [9]). Par conséquent, le problème ECID appartient à la classe **NP**.

Nous allons prouver que ce problème est également **NP**-dur. Étant donnée une collection C de sous-ensembles d'un ensemble fini E, une *couverture* de E est un sous-ensemble $C' \subseteq C$ tel que chaque élément de E appartient à au moins un membre de C'. Nous définissons formellement le problème de décision MINIMUM SET COVER comme suit.

MINIMUM SET COVER
DONNÉES*: Une collection $C = \{C_1, C_2 \dots C_m\}$ de sous-ensembles d'un ensemble fini $E = \{e_1, e_2 \dots e_n\}$, et un entier positif k'.*
QUESTION*: C contient-elle une couverture C' de E telle que $|C'| \leq k'$?*

Par la suite, nous considérons que pour tout $1 \leq i \leq m$, tout élément e de $C_i \in C$ est également un élément de $C_j \in C$ pour au moins un $1 \leq j \leq m$ donné tel que $i \neq j$. En effet, par définition, si un élément n'appartient qu'à un seul sous-ensemble, alors ce sous-ensemble fait obligatoirement partie de C'. Par la suite, nous prouverons que le problème ECID est **NP**-complet même dans le cas où G est un génome exemplaire. Dans le reste de cette section, nous considérons que les génomes sont représentés par des séquences d'entiers signés. Nous commençons par détailler la construction des deux génomes G et H.

Soit $y = |E| + 2$ si $|E|$ est pair, $y = |E| + 1$ sinon. Soit $z_i = (y+2).(i-1)$ pour tout $1 \leq i \leq m+1$. À partir de (C, E), nous construisons, dans un premier temps, deux génomes exemplaires G et H_1. Puis nous transformons le génome H_1 pour qu'il ne soit plus exemplaire (une illustration est donnée en Figure 10.1).

Soient les sous-séquences composant les génomes G et H_1 suivantes:

- pour tout $1 \leq i \leq m$, nous construisons les séquences de gènes $\alpha_i = z_i$ et $\beta_i = z_i$+1 z_i+2 … z_i+y+1;
- pour tout $1 \leq i \leq 2|E| + m - 1$, nous construisons le gène $\gamma_i = z_{m+1} + i$;

- pour tout $1 \leq i \leq m$, nous construisons le gène $\theta_i = z_i{+}2\ z_i{+}4 \ldots z_i{+}y\ z_i{+}1\ z_i{+}3 \ldots z_i{+}y{-}1\ z_i{+}y{+}1$.

Nous définissons les génomes G et H_1 comme suit.

$$G = \gamma_{|E|+1}\ \gamma_{|E|+2} \cdots \gamma_{|E|+m-1} \alpha_1\ \beta_1\ \ldots \alpha_m\ \beta_m\ \gamma_1\ \gamma_{|E|+m}\ \gamma_2\ \gamma_{|E|+m+1} \cdots \gamma_{2|E|+m-1}\ \gamma_{|E|}$$
$$H_1 = \alpha_1\ \theta_1\ \gamma_{|E|+1}\ \alpha_2\ \theta_2\ \gamma_{|E|+2} \cdots \gamma_{|E|+m-1}\ \alpha_m\ \theta_m\ \gamma_{|E|+m}\ \gamma_{|E|+m+1}\ \cdots\ \gamma_{2|E|+m-1}$$

Nous transformons alors H_1 en un génome non-exemplaire H comme suit: pour $1 \leq i \leq m$ et $1 \leq j \leq |E|$, si $e_j \in C_i$ alors le gène γ_j est inséré entre le $j^{ème}$ et le $j{+}1^{ème}$ gènes de θ_i. Nous appelons ECID-construction toute construction de ce type. Une illustration d'une ECID-construction est donnée en Figure 10.1. Intuitivement, pour $1 \leq i \leq m$ et $1 \leq j \leq |E|$, la suite de gènes θ_i est une copie la suite de gènes β_i sans adjacence conservée (*i.e.* un mélange) et agrémenté de gènes γ_j. Clairement, notre construction peut être effectuée en temps polynomial. De plus, le résultat d'une telle construction est en effet une instance du problème ECID.

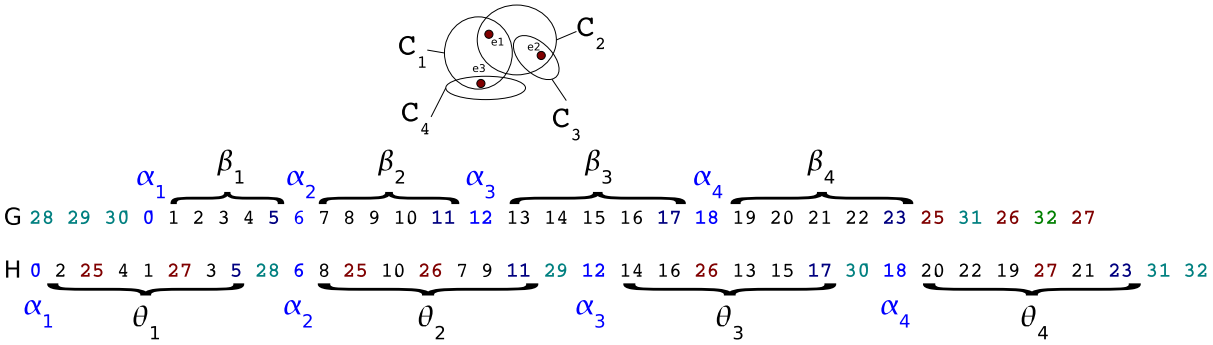


Figure 10.1 – Exemple d'une ECID-construction où $E = \{e_1, e_2, e_3\}$ et $y = 4$.

Nous allons maintenant prouver que notre transformation est une réduction valide du problème MINIMUM SET COVER vers le problème ECID. Notons, dans un premier temps que, par construction, il n'y a que $|E|$ familles dupliquées de gènes dans G et H, à savoir les gènes γ_i pour $1 \leq i \leq |E|$.

Lemme 10.1. *Pour tout $1 \leq i \leq m$ et tout $1 \leq j \leq |E|$, les seuls intervalles conservés pouvant exister entre G et toute exemplarisation H' de H sont des intervalles de la forme $[\alpha_i, z_i + y + 1]$ où tous les gènes γ_j de $[\alpha_i, z_i + y + 1]$ dans H' ont été supprimés.*

Preuve. Étant donnés un génome G et une exemplarisation H' d'un génome H, par construction, pour tout $1 \leq i \leq m-1$, le gène α_1 est situé dans G après le gène $\gamma_{|E|+i}$. Tandis que, pour tout $1 \leq i \leq m-1$, le gène α_1 est situé dans H' avant le gène $\gamma_{|E|+i}$.

De plus, les gènes $\gamma_{|E|+1}$, $\gamma_{|E|+2}$, $\ldots$ $\gamma_{|E|+m-1}$ sont consécutifs dans G, tandis qu'ils ne le sont pas dans H'. Par conséquent, pour tout $1 \leq i \leq m-1$ et tout gène δ, l'intervalle de la forme $[\gamma_{|E|+i}, \delta]$ ou $[\delta, \gamma_{|E|+i}]$ n'est pas un intervalle conservé.

Par construction, pour tout $1 \leq i \leq m$, la suite de gènes θ_i est constitué de l'ensemble des gènes de la suite β_i ainsi que d'au moins un gène γ_j où $1 \leq j \leq |E|$. De plus, pour tout $1 \leq i \leq m$, la définition de θ_i assure que deux gènes adjacents dans β_i ne le sont pas dans θ_i. Par conséquent, pour tout $1 \leq i \leq m$ et tout gène δ, l'intervalle de la forme $[\psi, \delta]$ ou $[\delta, \psi]$ où ψ est un gène de β_i n'est pas un intervalle conservé.

Par construction, pour tout $1 \leq i \leq |E|$, le gène α_m est situé dans G avant tout gène γ_i. Tandis que, pour tout $1 \leq i \leq |E|$, le gène α_m est situé dans H' après tout gène γ_i. De plus, pour tout $1 \leq i \leq |E|$, le gène γ_i est adjacent dans G à un gène de $\{\gamma_{|E|+j} | m \leq j \leq |E| + m - 1\}$, tandis que γ_i et $\gamma_{|E|+j}$ ne le sont pas dans H'. Par conséquent, pour tout $1 \leq i \leq |E|$ et tout gène δ, l'intervalle de la forme $[\gamma_i, \delta]$ ou $[\delta, \gamma_i]$ n'est pas un intervalle conservé.

Finalement, pour tout $1 \leq i \leq m$ et tout gène δ, il existe dans H' au moins un gène de $\{\gamma_{|E|+i} | 1 \leq i \leq m - 1\}$ appartenant à un intervalle de la forme $[\alpha_i, \delta]$ tel que le gène δ est situé après le gène α_{i+1}. Tandis que, pour tout $1 \leq i \leq m - 1$ et tout $1 \leq j \leq m$, le gène $\gamma_{|E|+i}$ est situé dans G avant tout gène α_j. Par conséquent, pour tout $1 \leq i \leq |E|$ et tout gène δ, l'intervalle de la forme $[\gamma_i, \delta]$ ou $[\delta, \gamma_i]$ n'est pas un intervalle conservé. La preuve du lemme en découle. □

Lemme 10.2. *Soit $I = (C, E, k')$ une instance du problème* MINIMUM SET COVER *pour une collection $C = \{C_1, C_2 \dots C_m\}$ de sous-ensembles d'un ensemble fini $E = \{e_1, e_2 \dots e_n\}$, et $I' = (G, H, k)$ une instance du problème* ECID *obtenue à partir d'une* ECID*-construction de I.*

La collection C contient une couverture C' de E de taille inférieure ou égale à k' si et seulement si il existe une exemplarisation H' du génome H telle que $\mathcal{D}_{ic}(G, H') \leq k = |G|.|G - 1| - 2(m - k')$.

Preuve. ($\Rightarrow$) Supposons que C contient une couverture C' de E de taille inférieure ou égale à k'. Soit $f : e_i \rightarrow \{1, 2, \dots m\}$ une fonction qui, étant donné un élément e_i de E, retourne l'indice du sous-ensemble couvrant cet élément dans C'. Soit $I' = (G, H, k)$ une instance du problème ECID obtenue à partir d'une ECID-construction de I. Nous recherchons une exemplarisation H' du génome H telle que $\mathcal{D}_{ic}(G, H') \leq k$. Nous définissons l'exemplarisation H' du génome H comme suit: pour chaque $e_j \in E$ et pour tout $p \in \{1, 2, \dots, m\}/\{f(e_j)\}$, supprimer γ_j de θ_p.

Par construction, les seuls gènes dupliqués de H sont les gènes $\gamma_1, \gamma_2, \dots, \gamma_{|E|}$. Par conséquent, le génome H' est exemplaire puisque, pour tout $1 \leq i \leq |E|$, toutes les occurrences sauf une de la famille de gènes γ_i ont été supprimées. Il nous reste à prouver que $\mathcal{D}_{ic}(G, H') \leq k$. Par définition, pour chaque $C_j \notin C'$ et chaque $e_i \in C_j$, $f(e_i) \neq j$ et par conséquent le gène dupliqué γ_i de θ_j a été supprimé. Étant donné que, pour tout $1 \leq j \leq m$, tous les gènes dupliqués de θ_j ont été supprimés dans H', il existe un intervalle conservé $[\alpha_j, z_j + y + 1]$ entre G et H'.

En tout, il en existe au moins $m - k'$. Par conséquent, il y a au moins $m - k'$ intervalles conservés entre les génomes G et H'. Donc, sachant que le nombre d'intervalles conservés entre un génome G et lui-même est égal à $\frac{|G|.|G-1|}{2}$ et que $|G| = |H'|$, on obtient $\mathcal{D}_{ic}(G, H') \leq |G|.|G - 1| - 2(m - k')$.

($\Leftarrow$) Supposons maintenant que H' est une exemplarisation de H telle que $\mathcal{D}_{ic}(G, H') \leq k$. Nous définissons l'ensemble C' comme suit: pour tout $1 \leq j \leq |E|$ et tout $1 \leq p \leq m$, si $\gamma_j \in \theta_p$ alors $f(e_j) = p$ et $C_p \in C'$. Nous allons maintenant prouver que C' définit une couverture de E de taille inférieure ou égale à k'.

Le génome H' étant une exemplarisation du génome H, il existe exactement une occurrence de chaque famille de gènes dans H'. Par conséquent, pour tout $e_i \in E$, C' contient au moins un élément de C contenant e_i. L'ensemble C' est donc une couverture de l'ensemble E. Il nous reste à prouver que $|C'| \leq k'$.

Par définition, puisque $\mathcal{D}_{ic}(G, H') \leq |G|.|G - 1| - 2(m - k')$, il existe au moins $m - k'$ intervalles conservés entre G et H'. D'après le Lemme 10.1, pour tout $1 \leq i \leq m$ et tout $1 \leq j \leq |E|$, les seuls intervalles conservés pouvant exister entre G et une exemplarisation H' de H sont les intervalles $[\alpha_i, z_i + y + 1]$ tels que tous les gènes γ_j de $[\alpha_i, z_i + y + 1]$ dans H' ont été supprimés. Par conséquent, il existe au moins $m - k'$ intervalles de ce type dans H'. Donc, il existe au plus k' intervalles $[\alpha_i, z_i + y + 1]$

contenant un gène γ_j où $1 \leq i \leq m$ et $1 \leq j \leq |E|$. On obtient donc $|C'| \leq k'$. Le lemme et, par conséquent, le Théorème 10.1 sont prouvés. □

10.3 Couplage et distance d'intervalles conservés

Étant donné le Théorème 10.1, on est en droit de se demander si plutôt que de supprimer toutes les occurrences sauf une des familles dupliquées de gènes, on ne pourrait pas calculer la distance d'intervalles conservés en prenant compte tous les gènes dupliqués [17, 86].

Dans cette section, nous abordons le problème du calcul de la distance d'intervalles conservés en présence de gènes dupliqués en utilisant des couplages. Nous démontrons que ce problème est également **NP**-complet. Nous débutons cette section par la définition formelle de ce problème.

MINIMUM CONSERVED INTERVAL MATCHING (MCIM)
DONNÉES*: Deux génomes G et H, et un entier positif k.*
QUESTION*: Existe-t-il un couplage maximal $\mathcal{M}$ entre les gènes de G et H tel que la distance d'intervalles conservés entre G et H est inférieure ou égale à k?*

Théorème 10.2. *Le problème* MINIMUM CONSERVED INTERVAL MATCHING *est* **NP***-complet.*

Afin de prouver le Théorème 10.2, nous proposons une réduction à partir du problème MINIMUM BIN PACKING (*cf.* Section 9.2). Notons que le problème MCIM appartient à la classe **NP** puisqu'étant donnés un entier k et un couplage maximal entre les gènes de deux génomes G et H, il est possible de calculer en temps polynomial le nombre d'intervalles conservés entre G et H et de vérifier que la distance est bien inférieure ou égale à k (*cf.* [9]).

Par la suite, nous prouverons que le problème MCIM est également **NP**-dur, même lorsqu'il n'y a qu'une seule famille dupliquée de gènes, ce qui implique le Théorème10.2. Soit $\mathcal{N} = k'.\mathcal{C} - \sum_{i=1}^{n} s(u_i)$.

Nous construisons les génomes G et H à partir d'une instance $(U, k', \mathcal{C})$ du problème MINIMUM BIN PACKING comme indiqué dans la Section 9.2. Les génomes obtenus G et H sont donc de la forme:

$$G = \alpha\ \mathbf{u'_1}\ A_1\ \mathbf{u'_2}\ A_2\ \ldots\ \mathbf{u'_{n+\mathcal{N}}}\ A_{n+\mathcal{N}}\ B_1\ B_2\ \ldots\ B_{k'+1}\ \beta$$
$$H = \alpha\ B_1\ \mathbf{U'_1}\ B_2\ \mathbf{U'_2}\ \ldots\ B_{k'}\ \mathbf{U'_{k'}}\ B_{k'+1}\ A_1\ A_2\ \ldots\ A_{n+\mathcal{N}}\ \beta$$

Une illustration de cette construction est donnée en Figure 10.2. Afin de compléter la construction de l'instance du problème MCIM, nous définissons formellement le paramètre k comme suit:
$k = \frac{|G|.(|G|-1)}{2} + \frac{|H|.(|H|-1)}{2} - 2q$ où $q = 1 + \sum_{i=1}^{n} \frac{s(u_i).(s(u_i)-1)}{2}$.

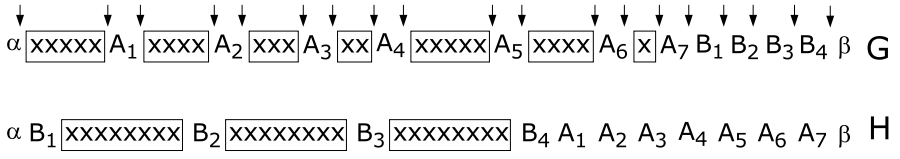


Figure 10.2 – Instance du problème MCIM associée à l'instance du problème MINIMUM BIN PACKING pour $k' = 3$, $\mathcal{C} = 8$ et $U = \{u_1, \ldots, u_6\}$ où $s(u_1) = s(u_5) = 5$, $s(u_2) = s(u_6) = 4$, $s(u_3) = 3$ et $s(u_4) = 2$.

Dans les trois prochains lemmes, nous considérons une instance $(U, k', \mathcal{C})$ du problème MINIMUM BIN PACKING et l'instance correspondante (G, H, k) du problème MCIM obtenue à l'aide de la construction présentée précédemment.

Lemme 10.3. *Étant donné un couplage $\mathcal{M}$, un segment de p gènes dupliqués parfaitement couplé dans $\mathcal{M}$ induit plus d'intervalles conservés (i.e. $\frac{p(p-1)}{2}$) qu'un segment de gènes dupliqués de longueur p qui n'est pas parfaitement couplé.*

Preuve. Par définition, dans G, un segment S de p gènes dupliqués parfaitement couplé induit $\frac{p(p-1)}{2}$ intervalles conservés. Si S n'est pas parfaitement couplé alors S est couplé avec au moins deux segments de H. Supposons, sans perte de généralité, que, pour un $p' \in [1 \dots p[$ donné, les segments $S[1, p']$ et $S[p'+1, p]$ de G sont parfaitement couplés dans H avec deux segments disjoints T et T' (*i.e.* les segments T et T' ne partagent aucun gène et ne sont pas adjacents dans H). Une illustration de ce cas est donnée en Figure 10.3. Par définition, le couplage de S induit $\frac{p'(p'-1)}{2} + \frac{(p-p')(p-p'-1)}{2} = \frac{2p'^2+p^2-2pp'-p}{2}$ intervalles conservés. Étant donné que $p' < p$, on a $2pp' > 2p'^2$. On a donc, $\frac{2p'^2+p^2-2pp'-p}{2} < \frac{p(p-1)}{2}$; ce qui prouve le lemme. □

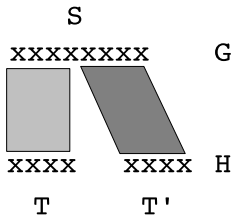


Figure 10.3 – Un segment de gènes dupliqués S de G couplé avec deux segments disjoints T et T' de H.

Lemme 10.4. *$\mathcal{D}_{ic}(G, H) \geq k$ et pour tout couplage $\mathcal{M}$ entre les gènes de G et H, tout intervalle conservé I est soit l'intervalle $[\alpha, \beta]$, soit un intervalle $I = [p, q]$ tel que $S[p, q]$ est un segment uniquement composé de gènes dupliqués.*

Preuve. Par construction, le gène α (*resp.* β) est le premier (*resp.* dernier) gène des deux génomes G et H. De plus, les génomes G et H étant équilibrés, ils sont composés du même ensemble de gènes. Par conséquent, l'intervalle $[\alpha, \beta]$ est un intervalle conservé. Par construction, pour tout $1 \leq i < n + \mathcal{N}$, au moins un gène $\mathbf{x}$ apparaît dans G après le gène A_i. Tandis que, pour tout $1 \leq i < n + \mathcal{N}$, tout gène $\mathbf{x}$ apparaît dans H avant le gène A_i.

De plus, il existe des gènes entre le gène $A_{n+\mathcal{N}}$ et le gène β dans G tandis que dans H ce n'est pas le cas. Par conséquent, pour tout $1 \leq i < n + \mathcal{N}$ et tout gène γ, l'intervalle $[A_i, \gamma]$ n'est pas un intervalle conservé. De plus, pour tout $1 \leq i \leq n + \mathcal{N}$, le gène $B_{k'+1}$ est situé dans H avant le gène A_i tandis qu'il est situé dans G après le gène A_i. Par conséquent, pour tout $1 \leq i < n + \mathcal{N}$ et tout gène γ, l'intervalle $[\gamma, A_i]$ n'est pas un intervalle conservé.

De façon similaire, par construction, pour tout $1 \leq j < k' + 1$, au moins un gène $\mathbf{x}$ apparaît dans H après le gène B_j. Tandis que, pour tout $1 \leq j < k' + 1$, tout gène $\mathbf{x}$ apparaît dans G avant le gène B_j. De plus, dans H il existe des gènes entre le gène $B_{k'+1}$ et le gène β tandis que dans G ce n'est pas le cas. Donc, pour tout $1 \leq j \leq k' + 1$ et tout gène γ, l'intervalle $[B_j, \gamma]$ n'est pas un intervalle conservé. De plus, pour tout $1 \leq j \leq k' + 1$, le gène $A_{n+\mathcal{N}}$ est situé dans H avant le gène B_j tandis qu'il est situé

dans G après le gène B_j. Par conséquent, pour tout $1 \leq j \leq k'+1$ et tout gène γ, l'intervalle $[\gamma, B_j]$ n'est pas un intervalle conservé.

On en déduit, que les seuls intervalles conservés pouvant exister sont des intervalles entre deux gènes $\mathbf{x}$: $[\mathbf{x}, \mathbf{x}]$. De plus, pour tout $1 \leq j \leq k'+1$ et $1 \leq i < n+\mathcal{N}$, le gène B_j étant situé dans G après le gène $A_{n+\mathcal{N}}$ et le gène A_i étant situé dans H après le gène $B_{k'+1}$, $I = [\mathbf{x}, \mathbf{x}]$ est un intervalle conservé si $A_i \notin I$ et $B_j \notin I$.

Par conséquent, dans tout couplage $\mathcal{M}$ entre les gènes de G et H, tout intervalle conservé I est soit de la forme $[\alpha, \beta]$, soit de la forme $I = [p, q]$ tel que $S[p, q]$ est un segment uniquement composé de gènes dupliqués. On en déduit donc, d'après le Lemme 10.3, qu'il y a au plus $\frac{p(p-1)}{2}$ intervalles conservés pour chaque segment de gènes dupliqués de longueur p. Par conséquent, dans tout couplage, il existe au plus $\frac{s(u_1)(s(u_1)-1)}{2} + \frac{s(u_2)(s(u_2)-1)}{2} + \ldots + \frac{s(u_n)(s(u_n)-1)}{2} = \sum_{i=1}^{n} \frac{s(u_i).(s(u_i)-1)}{2}$ intervalles conservés entre deux gènes $\mathbf{x}$. Donc, dans tout couplage, il existe au plus $1 + \sum_{i=1}^{n} \frac{s(u_i).(s(u_i)-1)}{2}$ intervalles conservés; par conséquent $\mathcal{D}_{ic}(G, H) \geq \frac{|G|.(|G|-1)}{2} + \frac{|H|.(|H|-1)}{2} - 2q$ où $q = 1 + \sum_{i=1}^{n} \frac{s(u_i).(s(u_i)-1)}{2}$. □

Lemme 10.5. *Il existe une partition de U en k' ensembles disjoints $U_1, U_2, \ldots, U_{k'}$ telle que pour tout $1 \leq i \leq k'$ la somme des tailles des éléments appartenant à U_i est inférieure ou égale à $\mathcal{C}$ si et seulement si $\mathcal{D}_{ic}(G, H) \leq k$.*

Preuve. ($\Leftarrow$) Supposons que $\mathcal{D}_{ic}(G, H) \leq k$. D'après le Lemme 10.4, nous savons que $\mathcal{D}_{ic}(G, H) = k$, et que, quelque soit le couplage entre les gènes de G et H, tout intervalle conservé I est de la forme $[\alpha, \beta]$ ou $I = [p, q]$ tel que $S[p, q]$ est un segment composé uniquement de gènes dupliqués. De plus, si $\mathcal{D}_{ic}(G, H) = k$ alors le nombre d'intervalles conservés doit être maximal (*i.e.* $1 + \sum_{i=1}^{n} \frac{s(u_i).(s(u_i)-1)}{2}$).

Par conséquent, d'après le Lemme 10.3, pour tout $1 \leq i \leq n$, le segment de gènes dupliqués u'_i de G doit être couplé avec une séquence de gènes $\mathbf{x}$ consécutifs de H. Plus précisément, pour tout $1 \leq i \leq n$, il existe un $1 \leq j \leq k'$ tel que le segment u'_i est parfaitement couplé avec un sous-segment de U'_j comme illustré en Figure 10.4.

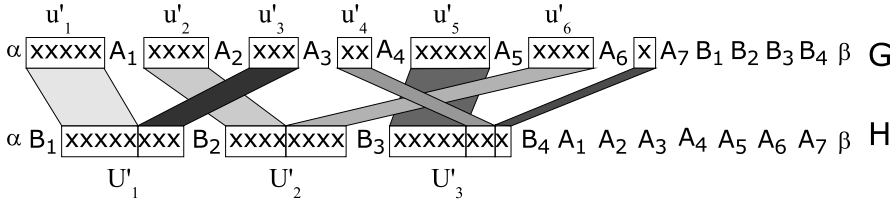


Figure 10.4 – Instance du problème MCIM associée à l'instance du problème MINIMUM BIN PACKING avec $k' = 3$, $\mathcal{C} = 8$ et $U = \{u_1, \ldots, u_6\}$ où $s(u_1) = s(u_5) = 5$, $s(u_2) = s(u_6) = 4$, $s(u_3) = 3$ et $s(u_4) = 2$ et le couplage correspondant à la partition de U suivante : $U_1 = \{u_1, u_3\}$, $U_2 = \{u_2, u_6\}$ et $U_3 = \{u_4, u_5\}$.

Par conséquent, un tel couplage induit une partition P de l'ensemble des séquences $\{u'_1, u'_2, \ldots, u'_n\}$ en au plus k' séquences disjointes $U'_1, U'_2, \ldots, U'_{k'}$. D'après la construction, pour tout $1 \leq i \leq k'$ $|U'_i| = \mathcal{C}$, donc P correspond à une solution de l'instance du problème MINIMUM BIN PACKING correspondante.

($\Rightarrow$) Supposons donnée une partition P de U en k' ensembles disjoints $U_1, U_2, \ldots, U_{k'}$ chacun de cardinalité au plus $\mathcal{C}$. Nous construisons un couplage $\mathcal{M}$ entre les gènes de G et H comme suit:

1. chaque gène non-dupliqué dans G est couplé à son unique copie dans H et

2. pour tout $1 \leq j \leq k'$ et pour chaque $u_i \in U$, si $u_i \in U_j$ alors la séquence de gènes $\mathbf{x}$ de u'_i dans G est parfaitement couplée avec la première séquence de gènes $\mathbf{x}$ libres (*i.e.* pas encore couplés) de U'_j dans H.

Le couplage $\mathcal{M}$ étant construit d'après la partition P, nous pouvons affirmer que, pour tout $1 \leq i \leq n$, le segment de gènes dupliqués u'_i est couplé à un segment de gènes $\mathbf{x}$ de H. Par conséquent, pour tout $1 \leq i \leq n$, le segment de gènes dupliqués u'_i de G induit $\frac{s(u_i).(s(u_i)-1)}{2}$ intervalles conservés. Donc, le couplage $\mathcal{M}$ induit $1 + \sum_{i=1}^{n} \frac{s(u_i).(s(u_i)-1)}{2}$ intervalles conservés (*cf.* preuve du Lemme 10.4). On en déduit que $\mathcal{D}_{ic}(G, H, \mathcal{M}) \leq k$, et donc que $\mathcal{D}_{ic}(G, H) \leq k$. Le lemme, et par conséquent le Théorème10.2, sont prouvés. □

10.4 Heuristique pour le problème MCIM

Dans cette section, nous présentons une heuristique pour le problème MCIM. Notre approche repose sur l'intuition suivante: *les longs segments de gènes identiques, au renversement près, dans deux génomes couplés ensemble dans un couplage devraient minimiser la distance induite par le couplage.* Remarquons qu'étant donnés deux génomes sans gènes dupliqués, cette méthode renvoie la solution optimale. Malheureusement, ce n'est pas toujours le cas lorsque l'on considère des familles dupliquées de gènes. Malgré tout, cette approche fonctionne bien sur des données réelles. Dans le reste de cette section, nous considérons que les génomes sont représentés par des séquences d'entiers signés.

Le corps de notre algorithme pour résoudre le problème MCIM est constitué de la boucle suivante:

Identifier le plus long segment commun de gènes s entre G et H (*i.e.* un segment apparaissant, à un renversement complet près, dans G et dans H);
tant que *il existe un segment commun de gènes non-marqués* **faire**
  Remplacer s dans chaque génome par un entier g_c, appelé *gène compressé*, tel que $g_c \notin \mathcal{F}$ (ceci implique que le gène g_c n'est pas dupliqué);
  Marquer le gène g_c comme traité;
  Mémoriser le nombre de gènes de s dans $N_s[g_c]$ et l'ensemble des gènes de s dans $C[g_c]$;
  Identifier le plus long segment commun de gènes non-marqués s entre G et H.

Par la suite, G' et H' représentent respectivement les versions modifiées des génomes G et H. Une fois que l'algorithme a quitté la boucle, tout gène non-marqué g de G' et H' est supprimé; g étant, par définition, non commun (*i.e.* il y a plus d'occurrences de la famille de gènes g dans l'un des deux génomes). Cette première étape de l'algorithme conduit à la détermination d'un exemplarisation G' (*resp.* H') de G (*resp.* H). Clairement, cette étape fournit le couplage entre les gènes de G et H sous une forme compressée: le couplage correspondant $\mathcal{M}$ est obtenu en couplant parfaitement les segments de gènes correspondant à chaque gène compressé de G, comme illustré en Figure 10.5.

Dans un seconde étape, l'algorithme calcule la distance d'intervalles conservés induite par le couplage $\mathcal{M}$. D'après le Lemme 10.3, chaque gène compressé g_c de G', tel que $k = N_s[g_c]$, induit $\frac{k(k-1)}{2}$ intervalles conservés entre G et H. De plus, certains intervalles conservés entre G' et H' peuvent exister. Par conséquent, G' et H' étant exemplaires, l'algorithme calcule l'ensemble des intervalles conservés S_{ci} entre G' et H' en temps polynomial en utilisant l'algorithme défini dans [9].

Les génomes G' et H' étant composés de gènes compressés, pour chaque intervalle conservé $[g_{c1}, g_{c2}] \in S_{ci}$, $N_s[g_{c1}] \cdot N_s[g_{c2}]$ intervalles conservés entre G et H sont induits. En effet, si $[g_{c1}, g_{c2}] \in S_{ci}$

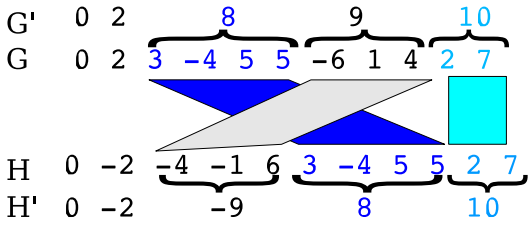


Figure 10.5 – Le couplage des gènes de G et H déduit des exemplarisations G' et H'.

alors un segment de gènes $g_{c1}\lambda g_{c2}$ apparaît dans G' et soit un segment de gènes $g_{c1}\lambda' g_{c2}$, soit un segment de gènes $-g_{c2}\lambda' - g_{c1}$ apparaît dans H' où λ et λ' sont des segments de gènes similaires à l'ordre et au signe près. Par conséquent, pour tous les gènes $g_i \in C[g_{c1}]$ et $g_j \in C[g_{c2}]$, $[g_i, g_j]$ est un intervalle conservé entre G et H. À la fin de cette étape, on renvoie donc $\mathcal{D}_{ic}(G, H, \mathcal{M}) = |G|.|G-1| - 2(\sum_{g_c \in G'} \frac{N_s[g_c].(N_s[g_c]-1)}{2} + \sum_{[g_{c1},g_{c2}]\in S_{ci}} N_s[g_{c1}] \cdot N_s[g_{c2}])$.

Nous avons implémenté cette approche en utilisant un arbre des suffixes. En effet, les plus longs segments de gènes communs entre deux génomes G et H peuvent être trouvés en temps linéaire par un parcours de l'arbre des suffixes construit sur les génomes G, H et le génome correspondant au renversement complet de H. Afin d'évaluer les performances de notre algorithme, nous l'avons appliqué à un ensemble de vingt génomes de bactéries provenant de la base NCBI[1].

Les données et programmes utilisés et mentionnés dans cette section peuvent être récupérés à l'adresse suivante

`http://www.sciences.univ-nantes.fr/info/perso/permanents/blin/Cocoon05/`

Les principales caractéristiques de cet ensemble de génomes sont illustrées en Figure 10.6. Il faut noter que, dans cette approche, nous avons identifié les familles dupliquées de gènes uniquement à l'aide du nom des gènes, à savoir: deux gènes appartiennent à la même famille de gènes s'ils portent le même nom. Nous sommes conscients que cette approche est très restrictive, c'est pourquoi nous sommes actuellement en train de poursuivre ces tests avec des définitions plus élaborées de gènes dupliqués [19, 18].

Afin d'évaluer l'efficacité de notre algorithme, nous avons implémenté également l'algorithme glouton consistant à calculer tous les couplages possibles et nous avons alors comparé les résultats obtenus par les deux algorithmes. En moyenne, l'écart entre la solution optimale opt et notre solution est inférieure à $0,12\%$ de opt. Notons que, de par la méthode utilisée pour sélectionner les familles dupliquées de gènes, plus de deux tiers des génomes étudiés ont des gènes dupliqués avec, pour la plupart, des familles dupliquées de cardinalité deux. L'efficacité de notre algorithme repose sur le fait que le nombre de gènes dupliqués n'est pas significatif comparativement à la taille des génomes. De plus, notre algorithme renvoyant la solution optimale dans le cas de génomes sans gènes dupliqués, les gènes dupliqués n'ont qu'un très faible impact sur les résultats.

10.5 Conclusion et perspectives

La supposition qu'il n'existe pas plus d'une occurrence d'une famille de gènes par génome a été jusqu'alors une contrainte nécessaire dans la plupart des méthodes de comparaison de génomes. Mais comme nous l'avons précisé, cette supposition est très restrictive et n'est justifiée que dans le cas de génomes de petits virus.

[1] http://www.ncbi.nlm.nih.gov/

Nom	Taille du génome	Taille de l'alphabet	Nombre de gènes dupliqués	Distribution des gènes dupliqués						
				Nombre de familles de gènes de taille						
				2	3	4	5	6	7	10
Aquifex Eeolicus	1517	1517	0							
Borrelia Burgdorferi	848	848	0							
Brucella Melitensis 1	3189	3178	19	7			1			
Caulobacter Crescentus	3719	3719	0							
Chlamydia Trachomatis	890	889	2	1						
Deinococcus Radiodurans 1	2922	2910	15	1				1	1	
Escherichia Coli K12	4232	4196	47	3	3		2	2		1
Haemophilus Influenzae	1699	1698	2	1						
Halobacterium sp. NRC-1	2048	2037	19	6	1	1				
Helicobacter Pylori 26	1563	1563	0							
Mycoplasma Genitalium	479	479	0							
Neisseria Meningitidis	2019	2016	6	3						
Rhizobium Loessense	6713	6713	0							
Rickettsia Conorii	1370	1367	6	3						
Streptococcus Pneumoniae	2093	2092	2	1						
Thermoplasma Acidophilum	1473	1472	2	1						
Treponema Pallidum	1028	1026	4	2						
Ureaplasma Urealyticum	606	606	0							
Vibrio cholerae C1	3813	3812	2	1						
Xylella Fastidiosa	2753	2752	2	1						
TOTAL	44974	44890	128	31	4	1	3	3	1	1
MOYENNE	2249	2245	6	2	2	1	2	2	1	1

Figure 10.6 – Principales caractéristiques du jeu de données.

Dans ce chapitre, nous avons étudié la complexité du problème de calcul de la distance d'intervalles conservés en considérant les deux stratégies classiques pour gérer les gènes dupliqués: l'exemplarisation et l'utilisation de couplages. Nous avons montré que ces deux stratégies induisent la **NP**-complétude du problème. Pour contourner ce résultat, nous avons proposé une heuristique pour le problème utilisant des couplages qui semble être efficace sur des données réelles.

Comme nous l'avons évoqué précédemment, la répartition des gènes en familles de gènes, où chaque famille est supposée représenter les groupes de gènes homologues, est très délicate. En effet, cette répartition induit l'encodage des génomes sous forme de séquences d'entiers signés, qui sont les entrées de notre algorithme et qui permettent donc l'élaboration des matrices de distances. Par conséquent, l'analyse des résultats de notre algorithme est très fortement dépendante de cette répartition initiale. Du fait de cette importance de la répartition des gènes en familles de gènes, et de manière à évaluer la qualité de notre approche, nous sommes actuellement en train de générer une seconde répartition des gènes basée sur l'utilisation de BLAST.

Il faut noter également que l'algorithme glouton mentionné à la section précédente a une complexité en temps $\mathbf{O}(k^{k.l}n)$ où k est la cardinalité maximale de toute famille dupliquée de gènes, l est le nombre de familles dupliquées de gènes et n est la taille du plus grand génome. Par conséquent, le problème MCIM appartient à la classe **FPT** pour les paramètres k et l. Ce résultat théorique n'est, a priori, pas convaincant en pratique, il serait donc intéressant d'étudier plus longuement l'approche paramétrée du problème afin de pouvoir déterminer si le problème appartient à la classe **FPT** pour d'autres paramètres.

Conclusion générale

Dans ce manuscrit, nous nous sommes intéressés, du point de vue algorithmique, à des problèmes issus du domaine de la biologie. Plus précisément, nous avons étudié d'une part la comparaison des structures de molécules d'ARN et d'autre part, le calcul de distances intergénomiques en présence de gènes dupliqués.

Dans le cadre de la comparaison de structures de molécules d'ARN, nous avons considéré quatre problèmes utilisant deux formalismes de représentation des molécules d'ARN: les séquences arc-annotées et les 2-intervalles. Dans un premier temps, nous avons étudié le problème L-EDIT qui correspond au calcul de la distance d'édition entre deux structures de molécules d'ARN. Nous avons prouvé que le dernier cas ouvert du problème L-EDIT, à savoir le sous-problème L-EDIT(NESTED,NESTED), est **NP**-complet. Pour ce faire, nous avons proposé une transformation polynomiale non triviale utilisant la notion de plongement en deux pages.

D'après Lin *et al.* [75], il existe un algorithme d'α-approximation pour résoudre le problème L-EDIT(NESTED,NESTED) de ratio $\alpha = max\{\frac{2w_a}{w_b+w_r}, \frac{w_b+w_r}{2w_a}\}$. Il serait intéressant de poursuivre l'étude de l'approximabilité du problème L-EDIT, et plus particulièrement, de déterminer s'il est possible ou non de concevoir un algorithme d'approximation ayant un ratio constant.

Le second problème auquel nous nous sommes intéressés est le problème ARC-PRESERVING SUBSEQUENCE (APS). Ce problème consiste à rechercher une sous-structure particulière, appelée motif, dans une molécule d'ARN. Nous avons prouvé que le dernier cas ouvert pour le problème APS, à savoir APS(CROSSING,PLAIN), est un problème **NP**-complet. Afin de mieux déterminer la frontière entre les cas polynomiaux et les cas **NP**-complets, nous avons proposé une nouvelle subdivision des niveaux de structures d'arcs. Dans ce nouveau modèle, nous avons déterminé l'ensemble des complexités du problème en fournissant deux preuves de **NP**-complétude et deux algorithmes polynomiaux.

Afin de contourner la **NP**-complétude du problème dans sa majorité, nous pensons qu'il serait intéressant d'étudier la complexité paramétrée du problème. En effet, la considération de paramètres sur la structure d'arc, tels que la largeur, la hauteur ou même la profondeur devrait permettre d'obtenir des complexités paramétrées efficaces en pratique.

Le troisième problème que nous avons étudié est le problème de la recherche d'un sous-ensemble de cardinalité maximale de 2-intervalles respectant un modèle de comparaison donné, le problème 2-IP. Nous avons résolu trois cas ouverts issus de [93, 94] en proposant deux algorithmes polynomiaux et une preuve de **NP**-complétude. Finalement, nous avons proposé un algorithme de complexité paramétrée par un paramètre fortement lié à la structure de croisement de l'ensemble de 2-intervalles.

Nous envisageons deux poursuites possibles de l'étude de ce problème. D'une part, il existe encore un cas ouvert pour le problème 2-IP, à savoir le sous-problème 2-IP(DISJOINT,$\{<, \bowtie\}$). Nous avons d'ores et déjà étudié ce problème sans pouvoir déterminer si le problème est **NP**-complet ou polynomial. Malgré tout, nous conjecturons que ce sous-problème est polynomial. D'autre part, il nous semble intéressant de poursuivre l'étude de la complexité paramétrée de ce problème. Un premier défi serait de déterminer si le problème appartient à la classe **FPT** pour le paramètre $\mathsf{Depth}(\mathcal{D})$ (*i.e.* la cardinalité maximale d'un sous-ensemble $\{\bowtie\}$-comparable de $\mathcal{D}$).

Le quatrième problème auquel nous nous sommes intéressés diffère quelque peu des trois autres et porte sur le design d'ARN. Nous avons étudié la complexité paramétrée du problème consistant à

inférer de nouvelles sélénoprotéines, le problème MRSO. Pour ce faire, nous avons utilisé un ensemble de notions de théorie des graphes qui se prêtent bien au problème étudié. Nous avons prouvé que le problème appartient à la classe **FPT** pour deux paramètres: le nombre de sommets de degré trois et le nombre de croisements (répondant ainsi à un problème ouvert posé dans [3]). Finalement, nous avons démontré que le problème restait **NP**-complet dans le cas où le graphe peut être plongé en deux pages.

Nous pensons qu'il serait intéressant de poursuivre l'étude de la complexité paramétrée de ce problème afin de considérer des fonctions de similarité plus élaborées. Par exemple, celles de la forme $fi : \Sigma^3 \mapsto \mathbb{N} \bigcup \{\text{-}\infty\}$ où $\Sigma = \{A, C, G, U\}$ où la valeur -∞ peut être utilisée pour interdire certain codon pour une position donnée.

Le deuxième axe de recherche que nous avons suivi concerne le calcul de distances intergénomiques en présence de gènes dupliqués. Nous avons considéré, chose peu étudiée jusqu'à présent, la présence de gènes dupliqués dans le calcul de deux distances évolutives: la distance de breakpoints et la distance d'intervalles conservés. Nous avons complété l'étude du cas général de ces deux distances en prouvant que le calcul de ces deux distances est un problème **NP**-complet si l'on considère l'approche utilisant des couplages. Nous avons également prouvé que l'utilisation de l'exemplarisation dans le cas de la distance d'intervalles conservés induit la **NP**-complétude du problème. Finalement, pour cette dernière distance, nous avons proposé une heuristique basée sur l'intuition selon laquelle les longs segments de gènes communs à deux génomes couplés dans un couplage devrait induire une petite distance entre deux génomes.

Ce sujet de recherche est encore actuellement en cours d'étude. En effet, dans [18, 19] nous avons initié l'étude d'une approche simple pour la reconstruction phylogénétique de génomes de procaryotes basée sur la construction de couplages entre gènes et de distances exprimées en termes de conservation de structure: les breakpoints, les intervalles communs et les intervalles conservés. Malgré sa simplicité, notre approche donne des résultats intéressants sur un ensemble de douze génomes de γ-Proteobactéries. En effet, les arbres phylogénétiques générés concordent avec l'arbre de référence proposé dans [72]. Nous sommes, actuellement, en train de poursuivre l'interprétation de nos résultats. Ces résultats n'étant pas finalisés, nous ne les avons pas inclus dans ce manuscrit.

Bibliographie

[1] K.A. ABRAHAMSON, R.G. DOWNEY et M.F. FELLOWS. Fixed-parameter intractability II. In *Proceedings of 10th Annual Symposium on Theoretical Aspects of Computer Science (STACS'93)*, volume 665 of *Lecture Notes in Computer Science*, pages 374–385. Springer-Verlag, 1993.

[2] Julien ALLALI. *Comparaison de structures secondaires d'ARN*. Thèse de Doctorat, Université de Marne-la-Vallée, 2004.

[3] R. BACKOFEN, N.S. NARAYANASWAMY et F. SWIDAN. On the complexity of protein similarity search under mRNA structure constraints. In *Proceedings of the 19th Symposium on Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *Lecture Notes in Computer Science*, pages 274–286. Springer-Verlag, 2002.

[4] R. BACKOFEN, N.S. NARAYANASWAMY et F. SWIDAN. Protein similarity search under mRNA structural constraints: application to targeted selenocystein insertion. *In Silico Biology*, 2(3):275–290, 2002.

[5] V. BAFNA, S. MUTHUKRISHNAN et R. RAVI. Computing similarity between RNA strings. In *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching (CPM'95)*, volume 937 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1995.

[6] V. BAFNA et P. PEVZNER. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[7] R. BAR-YEHUDA, M.M. HALLDORSSON, J. NAOR, H. SHACHNAI et I. SHAPIRA. Scheduling split intervals. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 732–741, 2002.

[8] S. BÉRARD, A. BERGERON, C. CHAUVE et C. PAUL. Conserved structures in evolution scenarios. In *Proceedings of the 2nd RECOMB Comparative Genomics Satellite Workshop*, volume 3388 of *Lecture Notes in Bioinformatics*, pages 1–15. Springer, 2004.

[9] A. BERGERON et J. STOYE. On the similarity of sets of permutations and its applications to genome comparison. In *Proceedings of the 9th International Computing and Combinatorics Conference (COCOON'03)*, volume 2697 of *Lecture Notes in Computer Science*, pages 68–79. Springer-Verlag, 2003.

[10] P. BERMAN et S. HANNENHALLI. Fast sorting by reversal. In *Proceedings of the7th Annual Symposium on Combinatorial Pattern Matching (CPM'96)*, volume 1075 of *Lecture Notes in Computer Science*, pages 168–185. Springer-Verlag, 1996.

[11] P. BERMAN, S. HANNENHALLI et M. KARPINSKI. 1.375-approximation algorithm for sorting by reversals. Rapport technique, Technical Report DIMACS TR:2001-41, 2001.

[12] P. BERMAN et M. KARPINSKI. On some tighter inapproximability results. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming (ICALP'99)*, volume 1644 of *Lecture Notes in Computer Science*, pages 200–209. Springer-Verlag, 1999.

[13] F. BERNHART et P.C. KAINEN. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979.

[14] T. BIEDL, G. KANT et M. KAUFMANN. On triangulating planar graphs under the four-connectivity constraint. *Algorithmica*, 19:427–446, 1997.

[15] B. BILLOUD, M-A. GUERRUCCI, M. MASSELOT et J.S. DEUTSCH. Cirripede phylogeny using a novel approach: Molecular morphometrics. *Journal of Molecular Biology and Evolution*, 19:138–148, 2000.

[16] J.R.S. BLAIR et B. PEYTON. An introduction to chordal graphs and clique trees. *Graph Theory and Sparse Matrix Computation*, 56:1–29, 1993.

[17] G. BLIN, C. CHAUVE et G. FERTIN. The breakpoints distance for signed sequences. In *Proccedings of the 1st Annual Conference on Algorithms and Computational Methods for Biochemical and Evolutionary Networks (CompBioNets'04)*, volume 3 of *Texts in Algorithms*, pages 3–16. KCL Publications, 2004.

[18] G. BLIN, C. CHAUVE et G. FERTIN. Genes order and phylogenetic reconstruction: application to γ-proteobacteria. In *Proceedings of the 3rd RECOMB Comparative Genomics Satellite Workshop*, To appear in Lecture Notes in Bioinformatics. Springer-Verlag, 2005.

[19] G. BLIN, C. CHAUVE et G. FERTIN. Ordre des gènes et reconstruction phylogénétique : Application aux γ-protéobactéries. In *Poster session of the 6ème Journées Ouvertes Biologie Informatique Mathématique (JOBIM'05)*, 2005.

[20] G. BLIN, G. FERTIN, D. HERMELIN et S. VIALETTE. Fixed-parameter algorithms for protein similarity search under mRNA structure constraints. In *Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG'05)*, To appear in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[21] G. BLIN, G. FERTIN, R. RIZZI et S. VIALETTE. What makes the arc-preserving subsequence problem hard ? In *Proceedings of the 1st International Workshop on Bioinformatics Research and Applications (IWBRA'05)*, volume 3515 of *Lecture Notes in Computer Science*, pages 860–868. Springer-Verlag, 2005.

[22] G. BLIN, G. FERTIN, R. RIZZI et S. VIALETTE. What makes the arc-preserving subsequence problem hard ? *To appear in Special issue of Lecture Notes in Computer Science Transactions on Computational Systems Biology*, 2005.

[23] G. BLIN, G. FERTIN, I. RUSU et C. SINOQUET. RNA sequences and the EDIT(NESTED,NESTED) problem. Rapport technique, RR-IRIN-03.07, IRIN, Université de Nantes. Submitted for publication, 2003.

[24] G. BLIN, G. FERTIN et S. VIALETTE. New results for the 2-interval pattern problem. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM'04)*, volume 3109 of *Lecture Notes in Computer Science*, pages 311–322. Springer-Verlag, 2004.

[25] G. BLIN et R. RIZZI. Conserved interval distance computation between non-trivial genomes. In *Proceedings of the 11th International Computing and Combinatorics Conference (COCOON'05)*, volume 3595 of *Lecture Notes in Computer Science*, pages 22–31. Springer-Verlag, 2005.

[26] A. BÖCH, K. FORCHHAMMER, J. HEIDER et C. BARON. Selenoprotein synthesis: a review. *Trends in Biochemical Sciences*, 16(2):463–467, 1991.

[27] D. BONGARTZ. Some notes on the complexity of protein similarity search under mRNA structure constraints. In *Proceedings of the 30th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'04)*, volume 2932 of *Lecture Notes in Computer Science*, pages 174–183. Springer-Verlag, 2004.

[28] D. BRYANT. The complexity of calculating exemplar distances. In *Comparative Genomics*, pages 207–212. Kluwer Academic Publication, 2000.

[29] G. CAETANO-ANOLLES. Tracing the evolution of RNA structure in ribosomes. *Nucleic Acids Research*, 30(11):2575–2587, 2002.

[30] A. CAPRARA. Sorting by reversals is difficult. In *Proceedings of the 1st Annual International Conference on Computational Molecular Biology (RECOMB'97)*, pages 75–83. ACM, 1997.

[31] W. CHAIA et V. STEWART. RNA Sequence Requirements for NasR-mediated, Nitrate-responsive Transcription Antitermination of the Klebsiella oxytoca M5al nasF Operon Leader. *Journal of Molecular Biology*, 292:203–216, 1999.

[32] P. CHARTRAND, X.H. MENG, R.H. SINGER et R.M. LONG. Structural elements required for the localization of ASH1 mRNA and of a green fluorescent protein reporter particle in vivo. *Current Biology*, 9(6):333–336, 1999.

[33] C. CHAUVE, G. FERTIN et S. VIALETTE. In *Communication personnelle*, 2005.

[34] X. CHEN, J. ZHENG, Z. FU, P. NAN, Y. ZHONG, S. LONARDI et T. JIANG. Assignment of orthologous genes via genome rearrangement. In *IEEE Transactions on Computational Biology and Bioinformatics (TCBB)*, To appear, 2005.

[35] D.A. CHRISTIE. A $\frac{3}{2}$-approximation algorithm for sorting by reversals. In *Proceedings of the 9th Annual SACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, pages 244–252. ACM-press, 1998.

[36] D.A. CHRISTIE et R.W. IRVING. Sorting strings by reversals and by transpositions. *SIAM Journal on Discrete Mathematics*, 14(2):193–206, 2001.

[37] M. CHROBAK, P. KOLMAN et J. SGALL. The greedy algorithm for the minimum common string partition problem. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'04)*, volume 3122 of *Lecture Notes in Computer Science*, pages 84–95. Springer-Verlag, 2004.

[38] S.A. COOK. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, ACM Press, pages 151–158, 1971.

[39] W. COOK et A. ROHE. Computing minimum-weight perfect matchings. *Journal on Computing*, 11:138–148, 1999.

[40] T.H. CORMEN, C.E. LEISERSON, R.L. RIVEST et C. STEIN. *Introduction to algorithms (second edition)*. MIT Press and McGraw Hill, 1992.

[41] P. CRESCENZI et V. KANN. A compendium of NP optimization problems. Rapport technique SI/RR-95/02, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", 1995.

[42] M. CROCHEMORE, D. HERMELIN, G.M. LANDAU et S. VIALETTE. Approximating the 2-interval pattern problem. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA'05)*, To appear in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[43] I. DAGAN, M.C. GOLUMBIC et R.Y. PINTER. Trapezoid graphs and their coloring. *Discrete Applied Mathematics*, 21:35–46, 1988.

[44] M. de BERG, M.van KREVELD, M. OVERMARS et O. SCHWARZKOPF. *Computational Geometry: Algorithms and Applications (2nd edition)*. Springer-Verlag, 2000.

[45] R. DIESTEL. *Graph Theory (2nd edition)*. Springer-Verlag, 2000.

[46] R. DOWNEY et M. FELLOWS. Parameterized complexity. *Springer-Verlag*, 1998.

[47] R.G. DOWNEY et M.F. FELLOWS. Fixed-parameter intractability. In *Proceedings of the 7th Annual Structure in Complexity Theory Conference*, pages 36–49, 1992.

[48] P.A. EVANS. *Algorithms and Complexity for Annotated Sequence Analysis*. Thèse de Doctorat, University of Victoria, 1999.

[49] A.D. FARRIS, G. KOELSCH, G.J. PRUIJN, W.J. van VENROOIJ et J.B. HARLEY. Conserved features of Y RNAs revealed by automated phylogenetic secondary structure analysis. *Nucleic Acids Research*, 27:1070–1078, 1999.

[50] S. FELSNER, R. MÜLLER et L. WERNISCH. Trapezoid graphs and generalizations: Geometry and algorithms. *Discrete Applied Math.*, 74:13–32, 1997.

[51] M.L. FREDMAN. On computing the length of longest increasing subsequences. *Disrete Mathematics*, 11:29–35, 1975.

[52] D.R. FULKERSON et O.A. GROSS. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965.

[53] M.R. GAREY et D.S. JOHNSON. *Computers and Intractability: a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[54] A. GOLDSTEIN, P. KOLMAN et J. ZHENG. Minimum common string partition problem: Hardness and approximations. In *Proceedings of the 15th International Symposium on Algorithms and Computation (ISAAC'04)*, volume 3341 of *Lecture Notes in Computer Science*, pages 484–495. Springer-Verlag, 2004.

[55] M.C. GOLUMBIC. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[56] J. GRAMM. A polynomial-time algorithm for the matching of crossing contact-map patterns. In *Proceedings of the 4th Workshop on Algorithms in Bioinformatics (WABI'04)*, Lecture Notes in Computer Science, pages 38–49. Springer-Verlag, 2004.

[57] J. GRAMM, J. GUO et R. NIEDERMEIER. Pattern matching for arc-annotated sequences. In *Proceedings of the 22nd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'02)*, volume 2556 of *Lecture Notes in Computer Science*, pages 182–193. Springer-Verlag, 2002.

[58] J. GUO. *Exact Algorithms for the Longest Common Subsequence Problem for Arc-Annotated Sequences*. Thèse de Doctorat, Universitat Tubingen, Federal Republic of Germany, 2002.

[59] D. GUSFIELD. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1998.

[60] R.R. GUTELL et C.R. WOESE. Higher order structural elements in ribosomal RNAs: pseudo knots and the use of noncanonical pairs. In *National Academy of Science USA*, volume 87, pages 663–667, 1990.

[61] K. HELLENDOORN, P.J. MICHIELS, R. BUITENHUIS et C.W. PLEIJ. Protonatable hairpins are conserved in the 5'-untranslated region of tymovirus RNAs. *Nucleic Acids Research*, 24:4910–4917, 1996.

[62] C. HERRBACH, A. DENISE et S. DULUC. A new operation scheme and a polynomial algorithm for tree-based pairwise comparison of RNA secondary structures. Submitted for publication, 2005.

[63] D.S. HIRSCHBERG. *The Longest Common Subsequence Problem*. Thèse de Doctorat, Princeton University - Canada, 1975.

[64] L. HOFACKER, M. FEKETE, C. FLAMM, M.A. HUYNEN, S. RAUSCHER, P.E. STOLORZ et P.F. STADLER. Automatic detection of conserved RNA structure elements in complete RNA virus genomes. *Nucleic Acids Research*, 26:3825–3836, 1998.

[65] T. JIANG, G.H. LIN, B. MA et K. ZHANG. The longest common subsequence problem for arc-annotated sequences. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (CPM'00)*, volume 1848 of *Lecture Notes in Computer Science*, pages 154–165. Springer-Verlag, 2000.

[66] V. JUAN, C. CRAIN et C. WILSON. Evidence for evolutionarily conserved secondary structure in the H19 tumor suppressor RNA. *Nucleic Acids Research*, 28:1221–1227, 2000.

[67] H. KAPLAN, R. SHAMIR et R. TARJAN. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM Press, 1997.

[68] R.M. KARP. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[69] J.D. KECECIOGLU et D. SANKOFF. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13(1-2):180–210, 1995.

[70] P.N. KLEIN. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms (ESA'98)*, Lecture Notes in Computer Science, pages 91–102. Springer-Verlag, 1998.

[71] P. KOLMAN. Approximating reversal distance for strings with bounded number of duplicates in linear time. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS'05)*, To appear in Lecture Notes in Computer Science, 2005.

[72] E. LERAT, V. DAUBIN et N.A. MORAN. From gene tree to organismal phylogeny in prokaryotes: the case of γ-proteobacteria. *PLoS Biology*, 1(1):101–109, 2003.

[73] V. I. LEVENSHTEIN. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1965.

[74] G.H. LIN, Z.Z. CHEN, T. JIANG et J. WEN. The longest common subsequence problem for sequences with nested arc annotations. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP'01)*, volume 2076 of *Lecture Notes in Computer Science*, pages 444–455. Springer-Verlag, 2001.

[75] G.H. LIN, B. MA et K. ZHANG. Edit distance between two RNA structures. In *Proceedings of the 5th International Conference on Computational Biology (RECOMB'01)*, pages 211–220. ACM Press, 2001.

[76] L.TICHIT. *Algorithmique des structures biologiques :l'édition d'arborescences pour la comparaison de structures secondaires d'ARN*. Thèse de Doctorat, Université Bordeaux I, 2003.

[77] G. MENDEL. Recherche sur les hybrides végétaux. *Bulletin Scientifique de la France et de la Belgique (traduit en français par A. Chappelier)*, 41:371–419, 1907.

[78] S. MICALI et V.V. VAZIRANI. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21th Annual IEEE Symposium on Foundation of Computer Science*, pages 17–27. IEEE, 1980.

[79] S.L. MITCHELL. Linear algorithms to recognize outerplanar and maximal outerplanar graphs. *Information Processing Letters*, 9(5):229–232, 1979.

[80] J. MONNOT, V. Th. PASCHOS et S. TOULOUSE. *Approximation polynomiale des problèmes NP-difficiles - Optima locaux et rapport différentiel*. Hermes Science, 2003.

[81] C.H. PAPADIMITRIOU et M. YANNAKAKIS. Optimization, approximation and complexity classes. *Journal of Computer & System Sciences*, 43:425–440, 1991.

[82] W. SAENGER. Principles of nucleic acid structure. *Springer-Verlag*, 1984.

[83] D. SANKOFF. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.

[84] D. SANKOFF et B. KRUSKAL. *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison*. Addison-Wesley, 1983.

[85] W.R. SCHMITT et M.S. WATERMAN. Linear trees and rna secondary structure. *Discrete Applied Mathematic*, 51(3):317–323, 1994.

[86] K.M. SWENSON, M. MARRON, J.E EARNEST-DEYOUNG et B.M.E. MORET. Approximating the true evolutionary distance between two genomes. Rapport technique TR-CS2004-15, Department of Computer Science, University of New Mexico, 2004.

[87] M.M. SYSLO. Characterizations of outerplanar graphs. *Discrete Mathematics*, 26:47–53, 1979.

[88] E. TANNIER et M.F. SAGOT. Sorting by reversals in subquadratic time. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM'04)*, volume 3109 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2004.

[89] R.E. TARJAN et M. YANNAKAKIS. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *Journal on Computing*, 13(3):566–579, 1984.

[90] S.W.M. TEUNISSEN, M.J.M. KRUITHOF, A.D. FARRIS, J.B. HARLEY, W.J. van VENROOIJ et G.J.M. PRUIJN. Conserved features of Y RNAs: a comparison of experimentally derived secondary structures. *Nucleic Acids Research*, 28:610–619, 2000.

[91] T. UNO et M. YAGIURA. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.

[92] V.V. VAZIRANI. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V|}|E|)$ maximum matching algorithm. *Combinatorica*, 14(1):71–109, 1994.

[93] S. VIALETTE. Pattern matching over 2-intervals sets. In *In Proceedings of the 13th Annual Symposium Combinatorial Pattern Matching (CPM'02)*, volume 2373 of *Lecture Notes in Computer Science*, pages 53–63. Springer-Verlag, 2002.

[94] S. VIALETTE. On the computational complexity of 2-interval pattern matching. *Theoretical Computer Science*, 312(2-3):223–249, 2004.

[95] H.Y. WANG et S.C. LEE. Secondary structure of mitochondrial 12S rRNA among fish and its phylogenetic applications. *Molecular Biology and Evolution*, 19:138–148, 2002.

[96] J. WUYTS, P. de RIJK, Y. V. de PEER, G. PISON, P. ROUSSEEUW et R. de WACHTER. Comparative analysis of more than 3000 sequences reveals the existence of two pseudoknots in area V4 of eukaryotic small subunit ribosomal RNA. *Nucleic Acids Research*, 28:4698–4708, 2000.

[97] M. YANNAKAKIS. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38:36–67, 1986.

[98] K. ZHANG et D. SHASHA. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[99] K. ZHANG, L. WANG et B. MA. Computing the similarity between RNA structures. In *Proceedings of the 10th Symposium on Combinatorial Pattern Matching (CPM'99)*, volume 1645 of *Lecture Notes in Computer Science*, pages 281–293. Springer-Verlag, 1999.

[100] M. ZUKER. RNA folding. *Methods in Enzymology*, 180:262–288, 1989.

Liste des tableaux

Partie I — Concepts généraux

Partie II — Comparaison de structures de molécules d'ARN

Partie III — Calcul de distances intergénomiques en présence de gènes dupliqués

Table des figures

Partie I — Concepts généraux

Partie II — Comparaison de structures de molécules d'ARN

Partie III — Calcul de distances intergénomiques en présence de gènes dupliqués

Table des matières

Index

Glossaire

A

Acide aminé Petite molécule dont l'enchaînement compose les protéines. Il existe 20 acides aminés différents utilisés pour fabriquer les protéines.

Acide DésoxyriboNucléique (ADN) Support biochimique de l'information génétique chez tous les êtres vivants (à l'exception de quelques virus qui utilisent l'ARN). Principal composant des chromosomes, l'ADN se présente le plus souvent sous forme de deux longs filaments (ou chaînes) torsadées l'un dans l'autre pour former une structure en double hélice. Chacune de ces chaînes est formée de l'assemblage de quatre nucléotides différents, désignés par l'initiale de la base azotée qui entre dans leur composition : A (Adénine), C (Cytosine), G (Guanine) et T (Thymine).

Acide RiboNucléique (ARN) L'ARN se présente le plus souvent sous la forme d'une unique chaîne formée de l'assemblage de quatre nucléotides différents, désignés par l'initiale de la base azotée qui entre dans leur composition : A (Adénine), C (Cytosine), G (Guanine) et U (Uracile). L'ARN est produit par transcription de l'ADN. Dans les cellules, on distingue plusieurs types d'ARN suivant leur fonction. Les trois types principaux sont : les ARN messagers, les ARN de transfert et les ARN ribosomaux.

Adénine Base azotée de la famille des purines.

ADN polymérase Dans la réplication de l'ADN, l'enzyme qui lie les bases nucléotidiques complémentaires pour former le brin nouvellement synthétisé.

Algorithme Méthode de résolution de problème énoncée sous la forme d'une série d'opérations à effectuer.

Arbre de décision Représentation graphique d'une séquence de décisions.

ARN de transfert (ARNt) Petits ARN responsables du transport des acides aminés jusqu'aux ribosomes lors de la traduction des ARNm : chaque ARNt transporte un acide aminé, de façon spécifique. Sa séquence comporte une série de trois nucléotides, nommée anticodon, qui reconnaît le codon (*cf.* code génétique) correspondant à l'acide aminé qu'il transporte.

ARN messager (ARNm) Photocopie du gène, il sert à transférer l'information génétique de son lieu de stockage (le chromosome) jusqu'au lieu de synthèse des protéines (les ribosomes). Les ARNm des cellules eucaryotes doivent subir une maturation, comprenant souvent un processus d'excision de leurs introns et d'épissage de leurs exons avant leur traduction en protéines.

ARN polymérase Lors de la transcription, cet enzyme se fixe à l'ADN, enfile les nucléotides pour former le brin synthétisé d'ARN et se détache de l'ADN lorsqu'il a fini.

ARN ribosomal (ARNr) Constituant principal des ribosomes, la machinerie cellulaire où a lieu la traduction en protéines de l'information contenue dans les ARNm.

B

Base azotée Voir nucléotide.

C

Chaîne eulérienne Dans un graphe G, c'est une suite de sommets reliés entre eux par des arêtes et passant par toutes les arêtes de G une et une seule fois.

Chemin hamiltonien Dans un graphe G, c'est une suite de sommets reliés entre eux par des arêtes et passant par tous les sommets de G une et une seule fois.

Chloroplaste Petit organe de la cellule végétale où se produit la photosynthèse.

Chromosome Elément essentiel, situé au coeur du noyau de chaque cellule, se présentant comme un très long filament sur lequel s'enchaînent les différents gènes.Il se présente souvent sous la forme de 2 bras : un bras long et un bras court.

Code génétique Système de correspondance permettant de traduire une séquence d'acides nucléiques en protéine. Dans ce système, un triplet de nucléotides, ou codon, désigne un acide aminé. Comme il existe 4 nucléotides, il y a 4x4x4 = 64 codons différents. À un codon donné correspond un seul et unique acide aminé. Par contre, il n'existe que 20 acides aminés différents dans les protéines, c'est pourquoi plusieurs codons peuvent désigner un même acide aminé - on dit que le code génétique est redondant.Certains de ces 64 codons ne désignent aucun acide aminé. Ces triplets " non-sens " indiquent à la machinerie cellulaire la fin de la lecture de l'information contenue dans les gènes, et provoquent l'arrêt de fabrication des protéines. On les appelle codons STOP.Tous les êtres vivants possèdent le même code génétique (à quelques variantes près): il est universel.

Codon Triplet de bases (ex. ACT ou GTC), codant un acide aminé ou encore le début ou la fin du message génétique (codon STOP).

Complexité d'un algorithme mesure du nombre d'opérations atomiques (*i.e.* indivisibles) qu'il effectue sur les entrées du problème. Elle est par conséquent exprimée en fonction de la taille des entrées du problème.

Connexe (graphe) Un graphe est connexe s'il existe une chaîne reliant chaque couple de sommets.

Cubique (graphe) Un graphe dont tous les sommets sont de degré 3.

Cytoplasme Le contenu d'une cellule, à l'exception du noyau.

Cytosine Base azotée de la famille des pyrimidines.

D

Degré Nombre d'arêtes incidentes à un sommet.

Désoxyribose Molécule de sucre entrant dans la composition des nucléotides constituant l'ADN.

Dogme central de la biologie moléculaire Le dogme central a été introduit pour la première fois par Francis Crick, puis établi par Jacques Monod, Francois Jacob et André Wolf en 1965. Il assied les mécanismes de stockage de l'information biologique, de réplication et de l'expression génique (transcription, traduction).

E

Efficacité Étant donnés deux algorithmes A et B pour résoudre un problème, A est dit *plus efficace* que B si sa complexité est inférieure à celle de B.

Enzyme Protéine dont la fonction est de catalyser, c'est-à-dire de faciliter et d'accélérer, les réactions bio-chimiques au sein de la cellule.

Epissage Mécanisme consistant, sur l'ARN messager qui vient d'être transcrit, à éliminer (exciser) les introns et réunir (épisser) les exons entre eux. Le produit de l'épissage est un ARN messager mature, prêt à être traduit en protéine.

Eucaryotes / Procaryotes L'ensemble des organismes vivants peut être classé en trois grands groupes : les eucaryotes, les eubactéries, les archaebactéries. A l'intérieur de chacune de leurs cellules, les eucaryotes possèdent un noyau : petit sac entouré d'une membrane semi-perméable renfermant les chromosomes. Les animaux, et donc l'homme, les plantes et les champignons, sont des eucaryotes. Les eubactéries et les archaebactéries ne possèdent pas de vrai noyau, mais une structure beaucoup plus simple, non entourée d'une membrane. C'est de ce " proto-noyau " que vient le nom générique qui les désigne : procaryotes.

Exon Chez les eucaryotes, les gènes sont le plus souvent constitués de deux types de séquence nucléotidique : l'une est dite codante et l'autre non codante. Les parties codantes, appelées exons, portent l'information qui sera directement utilisée pour fabriquer les protéines. Entre les exons se trouvent les introns, non " lus " lors de la traduction.

F

Feuillet beta Structure d'une molécule biologique formée par le repliement enaccordéon d'une séquence d'acides aminés sur elle-même.

G

Gène Fragment d'ADN portant les informations nécessaires à la fabrication d'une ou plusieurs protéine(s). Un gène comprend la séquence en nucléotides qui sera transcrite puis traduite en acides aminés, mais aussi des séquences permettant de réguler cette fabrication de protéine en fonction des conditions cellulaires. La longueur d'un gène peut varier de quelques centaines, à plus d'un million de nucléotides.

Génétique Science de l'hérédité. La génétique étudie les caractères héréditaires des individus, leur transmission au fil des générations et leurs variations (mutations). C'est l'étude de cette transmission héréditaire qui a permis l'établissement des lois de Mendel.

Génome Ensemble de l'information génétique d'un organisme. Une copie du génome est présente dans chacune de ses cellules. Le génome est transmis de génération en génération. Par extension, le génome se réfère aussi au support physique de cette information génétique, c'est-à-dire la macromolécule d'ADN.

Génotype Information génétique contenue dans les cellules de chaque individu, issue de la recombinaison des chromosomes au cours de la reproduction sexuée.

Graphe Ensemble fini de sommets et d'arêtes.

Groupe phosphate Groupe composé de phosphate e td'oxygène lié à une molécule organique.

Guanine Base azotée de la famille des purines.

H

Hélice alpha Structure d'une molécule biologique formée par le repliement en tire-bouchon d'une séquence d'acides aminés sur elle-même.

Hérédité Transmission de l'information génétique d'une génération à une autre.

Human Genome Project Projet du génome humain: projet de recherche réunissant des scientifiques du monde entier en vue de l'établissement de la carte génétique complète de l'être humain. `http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml`.

I

Intron Fragment non-codant d'un gène, situé entre deux exons. Les introns sont présents dans l'ARNm immature et absents dans l'ARNm mature.

Isthme Dans un graphe G, un isthme est une arête dont la suppression augmente le nombre de composantes connexes de G.

L

Liaison hydrogène Lien relativement faible entre un atome d'hydrogène et un atome d'azote ou d'oxygène.

Liaison peptidique Liaison entre le groupe amine (-NH2) d'un acide aminé et le groupe acide carboxylique (-COOH) d'un autre acide aminé. Elle est formée par élimination d'une molécule d'eau (H2O).

Liaison phosphodiester Une liaison phosphodiester est une liaison dans laquelle un groupe phosphate est lié à un groupement hydroxyle (OH) du sucre.

Locus Emplacement précis d'un gène sur un chromosome.

M

Maturation des ARNm Ensemble des modifications des ARNm préalables à leur traduction. L'épissage est un des processus de maturation des ARNm.

Méiose Processus de division des cellules avec réduction par deux du nombre de chromosomes.

Micro ARN (miARN) ARN de petite taille (19 à 25 nucléotides).

Mitochondrie C'est ce qui produit l'énergie de la cellule. Elle est en suspension dans le cytoplasme.

N

Nucléotide Motif structural de base des acides nucléiques, formé de l'assemblage de plusieurs molécules : un sucre (ribose pour l'ARN, désoxyribose pour l'ADN), un groupe phosphate et une base azotée (dans le cas de l'ARN cette base peut être l'Adénine - A, la Cytosine - C, la Guanine - G ou l'Uracile - U ; idem dans le cas de l'ADN, excepté que l'Uracile est remplacé par la Thymine - T).

O

Optimalité Un algorithme A, résolvant un problème P, est dit *optimal* si sa complexité est la complexité minimale parmi tous les algorithmes résolvant P.

P

Phénotype Expression effective du matériel génétique au cours du développement d'un individu.

Planaire (graphe) Un graphe est planaire s'il peut être dessiné sur un plan sans que les arêtes ne s'intersectent (en dehors des sommets).

Polypeptide Au moins dix acides aminés reliés ensemble.

Problème de décision Un problème est dit de décision lorsque la réponse à la question qu'il pose est soit *oui*, soit *non*.

Programmation dynamique Méthode d'optimisation procédant par énumération implicite des solutions. Cette approche consiste à aborder des problèmes d'optimisation avec une stratégie consistant en deux points essentiels:

- décomposer le problème en une séquence de sous-problèmes;
- établir une relation de récurrence entre les solutions optimales des problèmes.

.

Promoteur Courte séquence spécifique d'ADN, située au début des gènes, sur laquelle se fixe l'enzyme qui effectue la transcription (l'ARN polymérase). Etant nécessaire pour que la transcription débute, le promoteur est indispensable au fonctionnement d'un gène.

Protéine L'un des quatre matériaux de base de tout organisme. Les protéines sont formées d'un enchaînement spécifique d'acides aminés (de quelques dizaines à plusieurs centaines).

R

Recouvrement Un ensemble de sous-ensembles est un recouvrement si leur union est l'espace tout entier.

Réduction voir transformation polynomiale.

Réplication Mécanisme de synthèse de l'ADN permettant de transmettre l'information génétique d'une cellule ou d'un organisme à sa descendance. Chaque molécule-fille d'ADN est constituée d'un brin de la molécule-mère, qui sert de modèle à un nouveau brin. Ceci conduit à la duplication des molécules d'ADN de tout le génome.

Ribose Sucre, pentose, qui entre dans la composition des nucléotides qui constituent l'ARN.

Ribosome Petit organisme constitué de protéines et d'ARN (les ARNr), responsable de la traduction des ARNm.

S

Séquençage Le séquençage consiste, par des méthodes chimiques ou de biologie moléculaire, à déterminer l'ordre des nucléotides de l'ADN ou des acides aminés des protéines.

Stable (dans un graphe) Un stable dans un graphe G est un sous-graphe de G sans arête.

T

Théorie de la complexité L'étude formelle de l'efficacité des algorithmes ainsi que de la difficulté inhérente aux problèmes.

Thymine Base azotée de la famille des pyrimidines.

Traduction Étape de la synthèse (fabrication) des protéines au cours de laquelle le brin d'ARN messager obtenu lors de la transcription est converti en une chaîne d'acides aminés qui donnera une protéine. Les ARNm procaryotes sont traduits tels quels, les ARNm eucaryotes subissent une maturation préalable.

Transcription Processus de fabrication d'un ARNm à partir de la séquence codante d'un gène qui sert de modèle. L'enzyme responsable de cette réaction est appelée ARN polymérase.

Transcrit ARN messager synthétisé à partir d'un segment d'ADN.

Transformation polynomiale Il existe une transformation polynomiale d'un problème de décision $\mathcal{P}_1$ en un problème de décision $\mathcal{P}_2$ si, étant donnée toute instance I_1 de $\mathcal{P}_1$, il est possible de construire une instance I_2 de $\mathcal{P}_2$ en un temps polynomial (par rapport à la taille de I_1) tel que la réponse à I_1 est "oui" si et seulement si la réponse à I_2 est "oui" également.

U

Uracile Base azotée de la famille des pyrimidines.

Combinatoire et Bio-informatique: Comparaison de structures d'ARN et calcul de distances intergénomiques

Guillaume BLIN

Résumé

Nous présentons un ensemble de résultats concernant deux types de problèmes biologiques: (1) la comparaison de structures de molécules d'ARN et (2) le calcul de distances intergénomiques en présence de gènes dupliqués. Dans ce manuscrit, nous déterminons la complexité algorithmique de certains problèmes liés soit à la comparaison de structures de molécules d'ARN (distance d'édition, problème APS, recherche de motifs de 2-intervalles, design d'ARN), soit aux réarrangements génomiques (distances de breakpoints et d'intervalles conservés).
L'approche adoptée pour l'ensemble de ces problèmes a été de déterminer, si possible, des algorithmes exacts et rapides répondants aux problèmes posés. Pour tout problème pour lequel cela ne semblait pas possible, nous avons essayé de prouver qu'il ne peut être résolu de façon rapide. Pour ce faire, nous démontrons que le problème en question est algorithmiquement difficile. Enfin, le cas échéant, nous poursuivons l'étude de ce problème en proposant, essentiellement, trois types de résultats: (1) Approximation, (2) Complexité paramétrée, (3) Heuristique. Nous utilisons, dans ce manuscrit, des notions d'optimisation combinatoire, de mathématique, de théorie des graphes et d'algorithmique.

Mots-clés : Bio-informatique, Structures d'ARN, Réarrangements génomiques, Distances intergénomiques, 2-intervalles, Séquences arc-annotées, Complexité algorithmique

Abstract

We present a set of results concerning two types of biological problems: (1) RNA structure comparison and (2) intergenomic distance computation considering non trivial genomes. In this thesis, we determine the algorithmic complexity of a set of problems linked to either RNA structure comparison (edit distance, APS problem, 2-interval pattern extraction, RNA design), or genomic rearrangements (breakpoints and conserved intervals distances).
For each studied problem, we try to find an exact and fast algorithm resolving it. If we do not find such an algorithm, we try to prove that it is impossible to find one. To do so, we prove that the corresponding problem is difficult. Finally, we continue the study of each difficult problem by proposing three types of results: (1) Approximation, (2) Parameterized complexity, (3) Heuristic. We use in this thesis notions of combinatorics, mathematics, graph theory and algorithmics.

Keywords: Computational biology, RNA structures, Genomic rearrangements, Intergenomic distances, 2-intervals, Arc-annotated sequences, Algorithmic complexity