HAL
open science

Aspects algorithmiques de la prédiction des structures secondaires d'ARN

Stéphane Vialette

To cite this version:

Stéphane Vialette. Aspects algorithmiques de la prédiction des structures secondaires d'ARN. Informatique [cs]. Université Paris-Diderot - Paris VII, 2001. Français. NNT : . tel-00628623

HAL Id: tel-00628623

https://theses.hal.science/tel-00628623v1

Submitted on 5 Oct 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ PARIS 7 — DENIS DIDEROT
UFR D'INFORMATIQUE

Année 2000–2001

THÈSE

pour l'obtention du Diplôme de

DOCTEUR DE L'UNIVERSITÉ PARIS 7
SPÉCIALITÉ : ALGORITHMIQUE

présentée et soutenue publiquement

par

Stéphane Vialette

le 11 Décembre 2001

Titre :

Aspects algorithmiques de la prédiction des structures secondaires d'ARN

devant le jury constitué de

M.	CHRISTIAN CHOFFRUT,	Directeur de thèse
M.	MAXIME CROCHEMORE,	
M.	MICHEL HABIB,	Rapporteurs
MME.	FRÉDÉRIQUE LISACEK,	
M.	MICHEL MORVAN,	Examinateurs
MME.	MARIE-FRANCE SAGOT,	
M.	ALAIN VIARI.	

Résumé

Cette thèse traite deux types de problèmes algorithmique: des problèmes de triangularisation de matrices booléennes par permutation des lignes et des colonnes et des problèmes de découverte de structures secondaires d'ARN.

Nous étudions des problèmes de triangularisation de matrices booléennes par permutation des lignes et des colonnes. Ce problème apparaît, par exemple, lorsque l'on souhaite calculer «*en place*» un système d'équations. Une façon naturelle d'aborder ce problème est de se placer dans le cadre général de la théorie des graphes et des graphes bipartis en particulier. Nous présentons de nombreux résultats de complexité – essentiellement de NP-complétude – liés à ce problème et introduisons quelques extensions dont nous précisons toujours la complexité.

Certaines familles d'ARN sont très précisément définies par des motifs de séquence, et des contraintes structurelles secondaires et tertiaires. La plupart des outils ne sont pas adaptés puisqu'ils n'intègrent pas toutes les connaissances sur la molécule lors de l'exploration des banques de séquences. D'où l'intérêt d'algorithmes de recherche assurant une recherche en séquence et structure par le biais d'un descripteur défini par l'utilisateur intégrant l'ensemble des connaissances caractérisant l'ARN à détecter. Une nouvelle façon d'aborder ce problème consiste en l'étude de problèmes algorithmiques sur les graphes d'intersection d'un ensemble de 2-intervalles. Cette notion de 2-intervalles se trouve dans la lignée des études actuelles en matière d'algorithmique de graphes où l'on étudie de plus en plus les structures des graphes issues de modèles géométriques. Nous présentons plusieurs résultats de complexité et montrons en particulier que la recherche de motifs dans un ensemble de 2-intervalles est un problème NP-complet.

Nous nous intéressons, plus particulièrement, à appliquer ces travaux pour la prédiction de motifs biologiques structurés. Plus spécifiquement, nous avons mis au point l'algorithme ORANGE pour la prédiction des introns auto-catalytiques de groupe I dans de grandes séquences génomiques. Cet algorithme est une amélioration de l'algorithme CITRON mis au point par F. Lisacek et F. Michel du point de vue de la rapidité d'exécution. De plus, une mise-en-œuvre de l'algorithme ORANGE est accessible en ligne sur Internet.

Abstract

In this work we study two algorithmic problems: triangular forms in boolean matrices and spatial structure search problems in molecular biology.

We first consider the following problem: Given a square boolean matrix, can we permute its lines to yield a triangular matrix? This problem arises when we compute in place a system of equations. One natural way to solve the problem is to use graph theory, especially bipartite graphs. We show several NP-completeness results and introduce some extensions of the problem.

At the RNA level, biological signals are defined by a combination of spatial structures and sequence patterns. Until now, few attempts have been made to develop general purpose search programs that take into account both sequence and structure criteria. Therefore, there is a growing demand for search programs combining both descriptive and programming approaches. With this aim in view, we study combinatorial algorithmic problems on 2-intervals intersection graphs. Recent research considering the topic of intersection graphs of geometric objects shows the relevance of the 2-interval notion. We show that pattern matching in 2-intervals set is an NP-complete problem even when restricted to constrained spatial structures.

One important goal of this thesis is to use techniques for the prediction of structural biological patterns, and especially auto-catalytic group I introns. Our ORANGE algorithm for predicting auto-catalytic group I introns is an improved version of the CITRON algorithm of Lisacek and Michel. It is both time efficient and has good biological results. Moreover, an implementation of the ORANGE algorithm is available on the Internet.

Remerciements

Je tiens tout d'abord à remercier Christian Choffrut pour avoir encadré ma thèse. Je lui suis reconnaissant pour ses encouragements, ses précieux conseils, sa rigueur et pour tout ce qu'il m'a appris. Je pense très sincèrement que j'ai eu beaucoup de chance de l'avoir rencontré et qu'il accepte de devenir mon directeur de thèse.

Je tiens à remercier Michel Habib et Maxime Crochemore pour avoir accepté de référer cette thèse. Je suis sensible à l'honneur qu'ils me font.

Je remercie Frédérique Lisacek, non seulement pour tous ses conseils, ses efforts et sa gentillesse, mais aussi pour sa patience et sa bonne humeur. Ses conseils m'ont été précieux pour la rédaction du chapitre 7.

Je remercie, en plus de ceux déjà cités, Marie-France Sagot, Michel Morvan et Alain Viari d'avoir bien voulu faire partie de mon jury.

Je remercie celles et ceux qui ont fait des relectures de cette thèse, en particulier Geneviève Pentecôte et Isabelle Fagnot.

Ma rencontre avec Dan Téodosiu restera un de mes meilleurs souvenirs de ces années de thèse. Dans les nombreuses périodes de doute et de découragement, il m'a toujours encouragé et conseillé avec justesse. Son aide m'a été très précieuse et mon travail lui doit manifestement beaucoup.

Je remercie Pierre-Cyrille Héam, Massimiliano Milani, Cyril Nicaud, Christophe Prieur et Olivier Serre qui ont partagé mon bureau pendant quelques années. Leur aide a été précieuse sur tous les plans.

Je remercie toutes les personnes qui ont contribué à rendre agréable mon séjour au LIAFA. Merci donc à Isabelle Fagnot, Peter Habermehl, Ines Klimann, Roberto Mantaci, Yiannis Michos, Anca Muscholl, Mihaela Sighireanu, Laurent Vuillon, Jean-Baptiste Yunès, Marc Zeitoun et à tous ceux que j'oublie.

Je tiens également à remercier Noëlle Delgado pour sa gentillesse ainsi que Laifa Ahmadi et Zoubir Sami pour avoir facilité mon travail au LIAFA.

Je veux également remercier tous mes amis qui m'accompagnent depuis de nombreuses années maintenant. Je tiens donc tout particulièrement à remercier Céline, Sreng, Nelly, Laurent, Myriam, Dominique et Fabient[1]. Votre amitié m'est très précieuse. Je remercie également Rose et Sébastien[2] pour leur amitié et leur aide. Je veux aussi remercier Isabelle, Laurent, Frédéric et Bruno pour nos longues années d'amitié.

Je remercie mes parents et ma famille pour leur encouragements, et pour tout le reste.

Enfin, je remercie Patricia pour ses encouragements constants et tout l'amour qu'elle me donne. Elle a supporté avec patience mes crises d'angoisse et ma mauvaise humeur qu'a pu engendrer ma thèse. C'est indéniablement grâce à elle si j'ai pu mener ce travail à son terme.

1. malheureux bourreau du lustre de mes parents.
2. oui, *vraiment merci* Seb d'être parti à Orléans en nous laissant Rose ...

Il poursuivit donc son chemin; mais que ce chemin était long! En effet la route qui formait la rue principale du village, ne conduisait pas à la hauteur sur laquelle s'élevait le Château, elle menait à peine au pied de cette colline, puis faisait un coude qu'on eût dit intentionnel, et , bien qu'elle ne s'éloignât pas d'avantage du Château, elle cessait de s'en rapprocher. K. s'attendait toujours à la voir obliquer vers le Château, c'était ce seul espoir qui le faisait continuer; il hésitait à lâcher la route, sans doute à cause de sa fatigue, et s'étonnait de la longueur de ce village qui ne prenait jamais de fin; toujours ces petites maisons, ces petites vitres givrées et cette neige et cette absence d'hommes... Finalement il s'arracha à cette route qui le gardait prisonnier et s'engagea dans une ruelle étroite; la neige s'y trouvait encore plus profonde; il éprouvait un mal horrible à décoller ses pieds qui s'enfonçaient, il se sentit ruisselant de sueur et soudain il dut s'arrêter, il ne pouvait plus avancer.

Franz Kafka, *Das schloss*, Schocken Verlag, Berlin, 1935, Trad. *Le château*, trad. Alexandre Vialatte, Editions Gallimard, 1938, pp 20.

Table des matières

1 Introduction **1**

2 Rappels **11**

2.1 Graphe 11

2.1.1 Voisinage et degré 12

2.1.2 Sous-graphe 12

2.1.3 Chemin, circuit et connexité 13

2.1.4 Quelques graphes particuliers 14

2.1.5 Définitions complémentaires 15

2.2 Matrice 16

2.2.1 Introduction 16

2.2.2 Quelques matrices carrées particulières 16

2.2.3 Matrice de permutation 17

2.2.4 Opérations et relations 17

2.2.5 Matrices triangulaires 18

2.2.6 Compléments 18

2.3 Graphe et matrice 19

2.3.1 Introduction 19

2.3.2 Matrice d'adjacence 19

2.3.3 Graphe biparti 20

2.3.4 Matrice d'incidence sommets-arcs 20

2.3.5 Matrice d'incidence sommets-arêtes 21

3 Localisation **23**

3.1 Définitions 23

3.2 Automate et recherche de motifs 23

3.2.1 Sous-ensemble reconnaissable 23

3.2.2 Ensemble fini de mots 26

3.3 Représentation matricielle 27

3.3.1 Introduction 27

3.3.2 Semi-anneau 29

3.4 Résolution «*en place*» de système d'équations 31

3.4.1 Système d'affectations 31

3.4.2 Calcul en place d'un système d'affectations disjoint 32

3.5 Occurrences d'un sous-ensemble reconnaissable 35

3.5.1 Introduction 35

3.5.2 Définitions 38
3.5.3 Propriétés 39
3.5.4 Calcul en place 43
3.5.5 Application à la recherche des occurrences 45
3.6 Mise en œuvre 50
3.6.1 Calcul de $\overline{a}$ pour $a \in \Sigma$ 50
3.6.2 Complexité 53
3.6.3 Conclusion 54

4 Triangularisation d'une (0,1)-matrice 57
4.1 Introduction 57
4.2 Famille de sous-ensembles 60
4.2.1 Définitions et notations 60
4.2.2 Système de représentants 61
4.2.3 Propriété de croissance 62
4.3 Permanent non nul 65
4.3.1 Introduction 65
4.3.2 Algorithme 67
4.3.3 Conclusion 70
4.4 Arrangement linéaire triangulaire 70
4.4.1 Introduction 70
4.4.2 Arrangement linéaire triangulaire 74
4.4.3 Arrangement linéaire triangulaire et restrictions 79
4.4.4 Le cas des graphes acycliques 81
4.5 Sous-matrice et couplage 86
4.5.1 Introduction 86
4.5.2 Réduction 86
4.5.3 Couplage dans les graphes 88
4.5.4 Sous-graphe induit exact-triangulaire 89
4.5.5 Sous-graphe induit inf/sup-triangulaire 96
4.6 Sous-matrice principale 99
4.6.1 Introduction 99
4.6.2 $\mathbb{T} = \triangle_K$ (ou $\triangledown_K$) 100
4.6.3 $\mathbb{T} = \triangle_K$ (ou $\triangledown_K$) 104
4.7 Arrangement linéaire à cumul croissant 108
4.7.1 Introduction 108
4.7.2 Complexité 110
4.8 Conclusion et problèmes ouverts 113

5 Résultats complémentaires 117
5.1 Silhouette 117
5.1.1 Introduction 117
5.1.2 Définitions 120
5.1.3 Isomorphisme 122
5.1.4 Cas général 125
5.2 Surface 127
5.2.1 Introduction 127

5.2.2 Résultats expérimentaux 130
5.2.3 Minimisation de la surface d'une (0,1)-matrice 131
5.2.4 Le cas des graphes de séparation 144
5.2.5 Surface minimale et forme triangulaire 145
5.2.6 Résultat négatif d'approximation absolue 148
5.3 Famille de couples de sous-ensembles 150
5.3.1 Introduction 150
5.3.2 Complexité 151
5.4 Partition 154
5.4.1 Introduction 154
5.4.2 Partition en sous-matrices inf-triangulaires 155
5.4.3 Partition en sous-matrices exact-triangulaires 156
5.5 Conclusion et problèmes ouverts 161

6 Structures secondaires d'ARN 163
6.1 Introduction 163
6.2 Structure secondaire 166
6.2.1 Présentation 166
6.2.2 Références 168
6.3 Famille de 2-intervalles 168
6.3.1 Ensemble partiellement ordonné 168
6.3.2 Intervalle 169
6.3.3 2-intervalle 169
6.4 Graphe d'intersection 171
6.4.1 Graphe d'intersection 171
6.4.2 Graphe d'intervalles 172
6.4.3 Graphe de t-intervalle 174
6.5 Sous-famille de 2-intervalles disjoints 175
6.5.1 Introduction 175
6.5.2 Sous-famille de 2-intervalles $\mathcal{R} = \{<, \sqsubset, \sqsupset\}$-comparables . . 176
6.5.3 Sous-famille de 2-intervalles $\{<, \sqsubset\}$-comparables 180
6.5.4 Sous-famille de 2-intervalles $\{\sqsubset, \sqsupset\}$-comparables 183
6.5.5 Sous-famille de 2-intervalles $\{<\}$-comparables 185
6.5.6 Sous-famille de 2-intervalles $\{\sqsubset\}$-comparables 185
6.5.7 Sous-famille de 2-intervalles $\{\sqsupset\}$-comparables 186
6.5.8 Sous-famille de 2-intervalles $\{<, \sqsupset\}$-comparables 189
6.6 Recherche de motifs 191
6.6.1 Introduction 191
6.6.2 Définitions 193
6.6.3 Sous-famille de 2-intervalles $\mathcal{R}$-comparables et recherche de motifs 195
6.6.4 Modèle de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsupset\}$ 195
6.6.5 Modèle de comparaison $\mathcal{R} = \{\sqsubset, \sqsupset\}$ 198
6.6.6 Modèle de comparaison $\mathcal{R} = \{<, \sqsubset\}$ 203
6.6.7 Modèle de comparaison $\mathcal{R} = \{<\}$ 206
6.6.8 Modèle de comparaison $\mathcal{R} = \{\sqsubset\}$ 206
6.6.9 Modèle de comparaison $\mathcal{R} = \{\sqsupset\}$ 207

6.7 Conclusion et problèmes ouverts 207

7 Introns auto-catalytiques de groupe I **209**
7.1 (Très) Brefs rappels de biologie moléculaire 209
7.2 Un intron qui s'excise tout seul 212
7.3 Le programme *citron* 213
7.3.1 Présentation générale 213
7.3.2 Détails de mise en œuvre 216
7.4 Le programme *orange* 217
7.4.1 Introduction 217
7.4.2 Présentation générale 217
7.4.3 Recherche des hélices $(P7,P7')$ 218
7.4.4 Recherche des combinaisons $\{(P7,P7'),(P8,P8')\}$ 220
7.4.5 Recherche des combinaisons $\{(P6,P6'),(P7,P7')\}$ 221
7.4.6 Recherche des combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$. 221
7.4.7 Recherche des hélices $(P3,P3')$ 223
7.4.8 Score d'une solution 223
7.4.9 Décomposition en solutions non chevauchantes 223
7.4.10 Configuration 225
7.4.11 Filtrage 225
7.5 Quelques détails de mise en œuvre 226
7.5.1 Gestion des solutions partielles 226
7.5.2 Recherche des combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$. 226
7.6 Internet 228

A Formes triangulaires **229**
A.1 Matrice triangulaire 229
A.2 Sous-matrice principale 230
A.3 Sous-matrice 231

B Compléments sur les graphes de 2-intervalles **233**
B.1 Introduction 233
B.2 Graphe d'arêtes 234
B.3 Partition 235
B.4 Coupe 237
B.5 Clique 238

C Description des modules du logiciel *orange* **239**

D Exemple d'exécution du programme orange **243**

Index **247**

Chapitre 1

Introduction

L'informatique est devenue aujourd'hui un apport fondamental à la biologie moléculaire. Les traitements informatiques sont naturellement utilisés pour le stockage ou la gestion des données, mais également pour l'interprétation de ces données. Le traitement informatique des séquences peut, par exemple, déceler la fonction biologique potentielle d'un gène par la recherche de critères spécifiques (signaux, structures secondaires ou tertiaires, ...) ou par la recherche de similitudes entre séquences.

Un aspect important de l'informatique en biologie moléculaire concerne donc les traitements systématiques que l'on peut effectuer sur les séquences afin de repérer ou de caractériser une fonctionnalité ou un élément biologique intéressant. Ces programmes représentent les traitements couramment utilisés dans l'analyse des séquences comme l'identification de phases codantes sur une molécule d'ADN ou la recherche de similitudes d'une séquence avec l'ensemble des séquences d'une base de données. De plus, devant la difficulté réelle de la plupart de ces problèmes, les algorithmes mis en œuvre se doivent d'être les plus efficaces possible.

Je propose dans ce travail un algorithme performant pour la recherche des *introns auto-catalytiques de groupe I* dans les grandes séquences génomiques. Cet algorithme est mis en œuvre dans un logiciel baptisé `orange` entièrement paramètrable qui est librement téléchargeable. Il est également utilisable directement sur Internet.

* *
*

Le lecteur d'une version finale d'un document est toujours plus à même de comprendre ce travail s'il a connaissance des motivations, des choix et des errements successifs de son auteur. C'est pourquoi je tiens à inclure ici une courte chronologie de ma thèse.

Le point de départ de mon travail fut un problème proposé par F. Lisacek concernant la prédiction des introns auto-catalytiques de groupe I [MW90] dans les séquences génomiques. La difficulté du problème reposait essentiellement sur la grande variabilité de taille et la faible conservation des éléments structuraux de ces introns. Un programme de prédiction (`citron`) existait déjà mais se heurtait à de

graves problèmes d'explosion combinatoire. Ce problème entraînait de nombreuses limitations et interdisait en particulier la recherche de ces introns dans les grandes séquences.

Après analyse du problème, je décidai de partir de l'algorithme existant. Mon objectif était double : supprimer les parties de `citron` qui conduisaient à l'explosion combinatoire lors de la recherche et permettre une plus grande souplesse de description. Il m'apparut rapidement cependant que je ne pourrais pas utiliser le code source de `citron` et qu'une réécriture complète du programme de prédiction était indispensable. J'entrepris alors une analyse fine du code source pour en dégager les éléments essentiels ainsi que les méthodes utilisées (il est toujours difficile de décrire complètement dans un article toutes les astuces utilisées lors de la mise en œuvre d'un algorithme). Je m'intéressai ensuite principalement à la phase de recherche des hélices ($P4$,$P4'$), l'opération la plus coûteuse en temps et en espace dans `citron`. Je croyais alors fermement que ce problème pouvait être résolu par un calcul préalable des automates de recherche associés. Les résultats furent pourtant décevants, les contraintes proposées dans [Lis95] s'avérèrent trop faibles pour parvenir à un résultat satisfaisant. Je décidai pourtant de poursuivre cette voie et écrivis une version complète du programme de prédiction des introns auto-catalytiques de groupe I. Les premiers tests furent totalement catastrophiques. Non seulement le programme de prédiction était beaucoup plus lent, mais la plupart des limitations de `citron` demeuraient en fait présentes. Une vaste opération d'optimisations fines et de chasse aux gaspis n'améliora pas de façon significative les performances du programme. J'estimai alors être parvenu dans une impasse et une refonte complète de l'algorithme de prédiction s'imposa. Je décidai en particulier d'abandonner complètement l'ordre de recherche des hélices de `citron`. De plus, des algorithmes plus élaborés et des structures de données plus adaptées devaient être utilisés. Le nouvel algorithme de prédiction (`orange`) ne conserve aujourd'hui de `citron` que l'idée d'ancrer la recherche initiale sur les segments $J67 - P7$ et $J87' - P7'$ et de ma version initiale du programme que très peu de code.

Dans la continuité de ce travail, je m'intéressai également à une méthode de classification automatique des introns auto-catalytiques de groupe I suivant la présence ou non d'un motif (arrangement des hélices dans la structure secondaire) dans la boucle de l'hélice ($P5$,$P5'$). Ce problème m'a semblé suffisamment général et important pour y consacrer un chapitre. Je commençai par un travail de recherche bibliographique sur ce sujet : recherche de motif (au sens large), calcul de structure secondaire d'ARN, problèmes de graphe, ... La lecture d'un article de W.T. Trotter de F. Harary [TH79] sur une généralisation des graphes d'intervalle m'amena à considérer le problème sous cet angle. Mes premiers résultats de complexité sur ce problème me persuadèrent ensuite rapidement de la difficulté du problème (tout du moins dans le cas le plus général). J'estimai alors qu'une étude approfondie des différents cas particuliers de ce problèmes était nécessaire et souhaitable afin d'identifier précisément les cas «*difficiles*». Malheureusement, je n'ai jusqu'à aujourd'hui pas été en mesure de donner la complexité de ce problème dans le cas «*le plus biologique*», à savoir le cas d'une structure secondaire sans pseudo-nœud. Il me parut également opportun d'étudier la complexité de la généralisation la «*plus naturelle de ce problème*», c'est-à-dire la recherche d'un ensemble de cardinalité maximale d'hélices disjointes dans un ensemble donné.

Parallèlement à ce travail de prédiction dans les grandes séquences génomiques, je m'intéressai à un problème sur les (0,1)-matrices rencontré (dans une version très simplifiée) lors de l'écriture de l'algorithme de recherche en place des occurrences d'un ensemble reconnaissable dans un texte. L'extrême simplicité du problème contrastait avec ma difficulté à y apporter une réponse; pour être tout à fait exact, j'ai tout d'abord cru que le problème était trivial même dans le cas général sans regarder plus avant et ce n'est que plus tard - par hasard ou par curiosité - que je me suis réellement intéressé au problème en constatant que ma première impression n'était finalement pas justifiée (la lecture de [BR91] guida mes premiers pas sur ce problème). L'étude de la complexité d'un nouveau problème est bien souvent un pari, certes raisonné, mais un pari tout de même. Aussi, après avoir tenté en vain de proposer un algorithme en temps polynomial pour ce problème, je décidai d'abandonner temporairement cette voie. J'entrepris alors un vaste et long travail de recherche bibliographique sur les problèmes de graphes et les problèmes NP-complets en particulier; [GJ79] et [Die00] me furent d'un grand secours. Cependant, je ne parvins pas à prouver que le problème était NP-complet. J'examinai alors des problèmes plus généraux dans l'espoir de trouver un algorithme en temps polynomial. Mais, toutes ces généralisation s'avérèrent NP-complètes : arrangement d'une famille de sous-ensembles, recherche de silhouette, minimisation de surface, *etc.* Parvenu au terme de ma thèse, le problème est, certes mieux défini, mais toujours ouvert quant à sa complexité. Paradoxalement, la majeure partie de mon travail est consacrée à l'étude de ce problème.

* *
*

Le problème central abordé dans ma thèse est le suivant : Pouvons-nous rendre inférieurement triangulaire une matrice carrée par permutation des lignes et des colonnes? Tout au long de ma thèse, cette question est formulée ainsi :

> *Étant donnée une* (0,1)*-matrice A d'ordre n, existe-t-il des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \llcorner_n$?*

l'inégalité étant entendue composante à composante. En fait, j'ai tout d'abord rencontré ce problème en tentant de répondre à la question suivante :

> *Pouvons-nous, au prix de deux indirections, calculer* «en place» *un système de n équations à n inconnues ?*

Le terme «*en place*» est utilisé au sens de «*sans utiliser une variable temporaire pour stocker des résultats intermédiaires*»

Ce problème est résoluble en temps polynomial si le permanent de A est non nul [BR91]. Plus précisément, il existe des matrices de permutation P et Q d'ordre n telles que $\mathbb{I}_n \leq PAQ^T \leq \llcorner_n$ si et seulement si le permanent de A est 1 où $\mathbb{I}_n$ désigne la matrice identité d'ordre n.. Si le permanent de A est supérieur à 1, la matrice n'est pas triangularisable par permutation des lignes et des colonnes. Rappelons que nous pouvons décider si $\mathrm{per}(A) > 0$ en calculant un couplage parfait dans un graphe

biparti avec n sommets et m arètes en temps $O(m\sqrt{n})$ [HK73] ou $O(n^{1.5}\sqrt{m/\log n})$ en utilisant des techniques plus récentes [ABMP91]. Le problème reste donc posé pour $\mathrm{per}(A) < 0$.

Pour l'étude de la complexité de ce problème, j'introduis la notion d'*arrangement linéaire triangulaire* d'un graphe. En termes matricielles, le problème associé s'énonce de la façon suivante:

> *Étant donnée une* (0,1)*-matrice A d'ordre n, existe-t-il une matrice de permutation P d'ordre n telle que $PAP^T \leq \triangle_n$?*

La matrice de permutation étant la même à gauche et à droite, il s'agit en fait d'un problème de renumérotation des sommets d'un graphe. J'établis que si ce problème restreint aux matrices d'adjacence des *graphes de séparation* [Gol80] est NP-complet, alors la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes est également un problème NP-complet. En effet, pour certains graphes de séparation de matrice d'adjacence A, il existe des matrices de permutation P et Q telles que $PAQ^T \leq \triangle_n$ si et seulement si il existe une matrice de permutation R telle que $RAR^T \leq \triangle_n$. Je montre ensuite que le problème de décision $RAR^T \leq \triangle_n$ est NP-complet dans le cas général et ceci même si A est la matrice d'adjacence d'un graphe planaire connexe de degré au plus 3. Ce résultat tranche avec la question «*symétrique*» $PAP^T \leq \triangle_n$ qui est, elle, résoluble en temps polynomial (il suffit en effet de vérifier en temps $\Theta(|V| + |E|)$ que le graphe orienté $G = (V,E)$ associé à la (0,1)-matrice A est acyclique[1] [CLR92]). Cependant, je n'ai pas été en mesure de déterminer la complexité du problème $PAP^T \leq \triangle_n$ lorsque A est la matrice d'adjacence d'un graphe de séparation.

Je me suis ensuite intéressé à la complexité du problème $PAQ^T \leq \triangle_n$ lorsqu'un paramètre K est introduit dans l'instance. Je montre que, étant donnés une (0,1)-matrice A d'ordre n et un entier positif K, décider s'il existe des matrices de permutation P et Q d'ordre n telles que:

$$PAQ^T \leq \begin{bmatrix} \triangle_K & \mathbf{0}_{K,n-K} \\ \mathbf{1}_{n-K,K} & \mathbf{1}_{n-K,n-K} \end{bmatrix}$$

est un problème NP-complet même si A est la matrice d'adjacence d'un graphe de séparation. Le cas $K = n$ est exactement notre problème initial.

J'aborde ensuite ce problème dans le cadre des sous-matrices dont la taille minimale est fixée:

> *Étant donnés une* (0,1)*-matrice A et un entier positif K, existe-t-il une sous-matrice B d'ordre K triangularisable par permutation des ligne et des colonnes?*

J'établis que si ce problème est NP-complet, alors la triangularisation d'une (0,1)-matrice carrée par permutation des lignes et des colonnes est également un problème NP-complet. Dans cette direction, je montre que étant donnés une (0,1)-matrice A

1. Il s'agit donc de calculer un tri topologique d'un graphe orienté.

de taille m par n et un entier positif K, décider s'il existe les matrices de permutation P d'ordre m et Q d'ordre n telles que :

$$PAQ^T = \begin{bmatrix} \triangle_K & * \\ * & * \end{bmatrix}$$

est un problème NP-complet. En relation avec ce dernier résultat, je présente les couplages uniques restreints [GHL] dans les graphes bipartis en particulier. Je montre que le problème de décision associé à la recherche d'un couplage unique restreint comportant un nombre maximal d'arêtes dans un graphe biparti est un problème NP-complet[2].

Pour être complet, ce travail se poursuit dans le cadre plus restreint des sous-matrices principales dont la taille minimale est fixée :

> *Étant donnés une* (0,1)*-matrice A et un entier positif K, existe-t-il une sous-matrice principale B d'ordre K triangularisable par permutation simultanée des ligne et des colonnes ?*

Je donne précisément les complexités de tous les problèmes abordés dans le cas des sous-matrices quelconques.

J'introduis ensuite la notion de *silhouette* d'une (0,1)-matrice. La silhouette d'une (0,1)-matrice $A = [a_{i,j}]$ d'ordre n, notée $\mathbb{S}_A$, est une application de $[\![n]\!]$ dans $[\![n+1]\!]$ telle que $\mathbb{S}_A(j) = i$ si i est le plus petit indice pour lequel $a_{i,j} = 1$ (par convention, $\mathbb{S}_A(j) = n+1$ si la colonne j ne contient que des éléments nuls). J'associe à cette définition le problème de décision suivant :

> *Étant données une* (0,1)*-matrice A d'ordre n et une application π de $[\![n]\!]$ dans $[\![n+1]\!]$, existe-t-il des matrices de permutation P et Q d'ordre n telles que $\pi(j) \leq \mathbb{S}_{PAQ^T}(j)$ pour $1 \leq j \leq n$?*

Ce problème est équivalent à la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes si π est une permutation de $[\![n]\!]$. Je montre que le problème sous sa forme générale est NP-complet ; le problème obtenu en imposant $P = Q$ est également NP-complet. Une extension naturelle de ce problème consiste à imposer pour chaque colonne exactement la hauteur maximale d'un élément non nul :

> *Étant données une* (0,1)*-matrice A d'ordre n et une application π de $[\![n]\!]$ dans $[\![n+1]\!]$, existe-t-il des matrices de permutation P et Q d'ordre n telles que $\pi(j) = \mathbb{S}_{PAQ^T}(j)$ pour $1 \leq j \leq n$?*

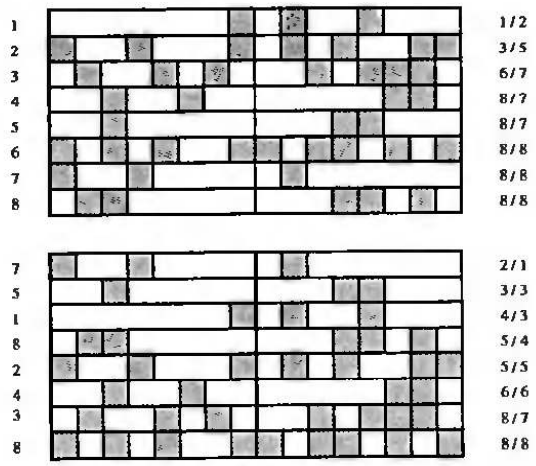
Je montre que ce problème est NP-complet ; le problème obtenu en imposant $P = Q$ est également NP-complet. Il faut remarquer cette fois-ci que le problème est résoluble en temps polynomial si π est une permutation de $[\![n]\!]$[3].

2. Le problème de décision associé à la recherche d'un couplage unique restreint comportant un nombre maximal de sommets dans un graphe biparti est un problème NP-complet [GHL].

3. Le problème est équivalent à la triangularisation par permutation des lignes et des colonnes d'une (0,1)-matrice dont le permanent est 1.

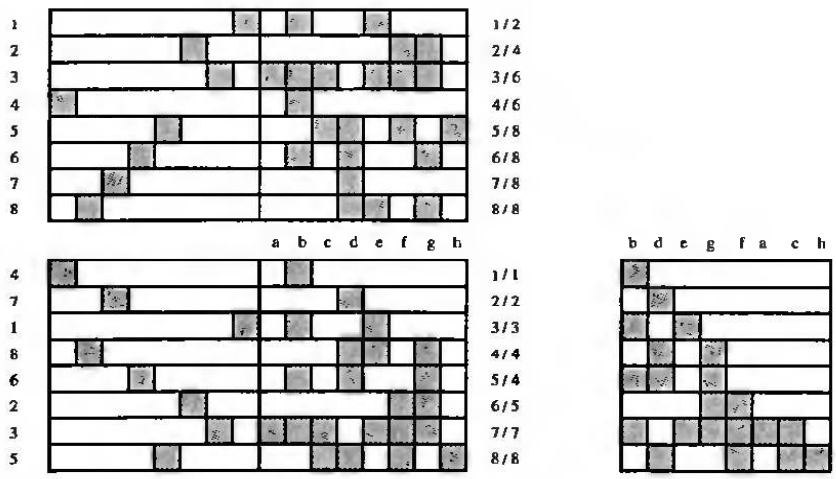
Un problème plus général concerne la compression d'une (0,1)-matrice par permutation des lignes. À ce titre, je présente la notion de surface d'une (0,1)-matrice qui est la mesure «*naturelle*» associée à sa silhouette. Intuitivement, la surface d'une matrice consiste à considérer la matrice comme un tableau et à compter pour chaque colonne le nombre de lignes à partir du premier élément non nul. Je montre que le problème de décision associé à la minimisation de surface d'une (0,1)-matrice par permutation des lignes[4] est un problème NP-complet dans le cas général et ceci même pour les matrices qui contiennent au plus deux éléments non nuls par colonne. Cependant les liens exacts entre ce problème et la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes restent en grande partie à établir.

Une nouvelle formulation du problème de la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes est obtenue en considérant un couple de (0,1)-matrices d'ordre n. Il s'agit dans ce cas de décider s'il existe une même permutation des lignes telle que le cardinal de l'union des k, $1 \leq k \leq n$, premières lignes de la matrice de gauche soit supérieur ou égal au cardinal de l'union des k premières lignes de la matrice de droite. Un exemple est donné ci-après.



Je montre que ce problème est NP-complet. Lorsque la matrice de gauche contient exactement un élément non nul par colonne, le problème est équivalent au problème de la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes. Il suffit en effet d'ordonner les colonnes de la matrice de droite suivant la position du premier élément non nul. Par exemple,

4. La surface d'une (0,1)-matrice est invariante par permutation des colonnes.



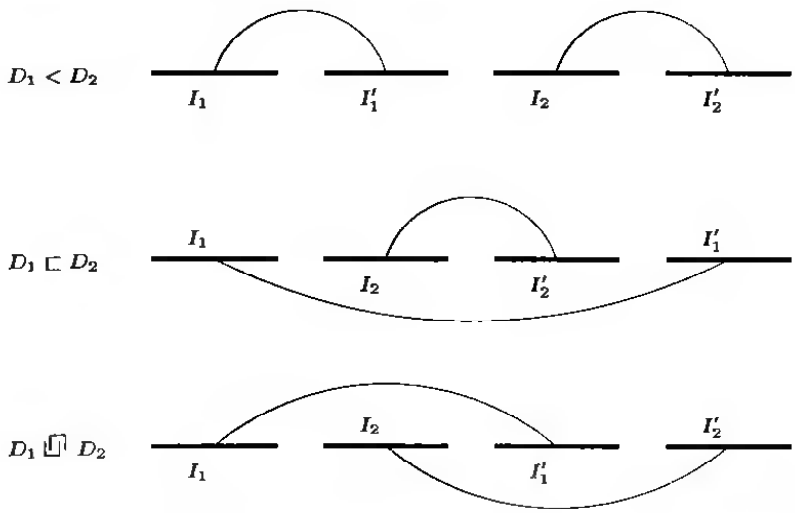
Je n'ai pas été en mesure de déterminer la complexité du problème dans ce cas précis[5].

* *

*

Motivé par le problème de la recherche d'un motif dans les structures secondaires d'ARN, je me suis intéressé aux familles de 2-intervalles et aux graphes d'intersection associés. Un 2-intervalle est un couple d'intervalles et un graphe de 2-intervalle est le graphe d'intersection d'une famille de 2-intervalles. Les graphes de 2-intervalles sont une généralisation des graphes d'intervalle [Gol80]. Ils ont été introduits par [Gol80] et étudiés dans [TH79] et [WS84]. Je me suis intéressé à différents problèmes de recherche de sous-familles de 2-intervalles comparables.

Étant donnés deux 2-intervalles $D_1 = (I_1, I_1')$ et $D_2 = (I_2, I_2')$, ils sont *comparables* s'ils sont dans l'une des trois configurations ci-après (ou symétriques) :



5. La complexité du problème obtenu lorsque chaque ligne de la matrice «*de gauche*» contient exactement un nombre borné d'éléments non nuls m'est également inconnue.

Un *modèle de comparaison* $\mathcal{R}$ permet de restreindre le nombre de ces configurations. Plus précisément, un modèle de comparaison est un sous-ensemble non vide de l'ensemble des relations $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$ tel que deux 2-intervalles sont $\mathcal{R}$-comparables si il sont comparables par l'une des relations de $\mathcal{R}$.

J'ai en fait étudié deux familles de problèmes sur les familles de 2-intervalles en relation avec les structures secondaires d'ARN :

1. calculer la plus grande sous-famille de 2-intervalles $\mathcal{R}$-comparables relativement à un modèle de comparaison $\mathcal{R}$ donné;
2. rechercher un motif respectant un modèle de comparaison $\mathcal{R}$ donné dans une famille de 2-intervalles.

Je montre en particulier les résultats suivants :

1. Le calcul d'un ensemble stable de cardinalité maximale restreint à la classe des graphes de 2-intervalles est un problème NP-complet. Plus généralement, la recherche d'une sous-famille de cardinalité maximale de 2-intervalles $\mathcal{R}$-comparables est un problème NP-complet pour les modèles de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsubset\!\!\sqsubset\}$ et $\mathcal{R} = \{\sqsubset, \sqsubset\!\!\sqsubset\}$.
2. La recherche d'un motif arbitraire dans une famille de 2-intervalles est un problème NP-complet pour les modèles de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsubset\!\!\sqsubset\}$ et $\mathcal{R} = \{\sqsubset, \sqsubset\!\!\sqsubset\}$[6].

En complément, je présente en annexe quelques problèmes classiques restreints à la classe des graphes de 2-intervalles. Les résultats s'appuient sur la remarque suivante : si un problème de décision sur les graphes est NP-complet restreint à la classe des graphes d'arêtes, alors il est également NP-complet restreint à la classe des graphes de 2-intervalles[7]. Il est alors facile de prouver qu'un problème est NP-complet restreint à la classe des graphes de 2-intervalles par un simple travail de recherche bibliographique. À titre d'exemple, je montre que les problèmes PARTITION EN CLIQUES, k-COLORIAGE[8] et ENSEMBLE DE COUPE STABLE sont NP-complets restreints à la classe des graphes de 2-intervalles.

* *

*

La dernière partie est consacrée à la prédiction des introns auto-catalytiques de groupe I [MW90] dans les séquence génomiques. Le but recherché est de maîtriser l'explosion combinatoire d'un algorithme existant (*citron*) [LDM94].

Je présente un nouvel algorithme *orange* dont je décris les différentes étapes. J'insiste en particulier sur les choix des structures de données et des algorithmes utilisés. Ce nouveau programme retrouve dans la majorité des cas les solutions prédites par *citron*. Il est cependant plus rapide et est complètement paramétrable *via* un fichier de configuration simple.

6. Le problème le «*plus biologique*» concerne évidemment le modèle de comparaison $\mathcal{R} = \{<, \sqsubset\}$. Je n'ai cependant pas été en mesure de déterminer la complexité du problème dans ce cas précis.

7. L'inverse n'est pas vrai en général.

8. nombre chromatique

Ce programme de prédiction est accessible en ligne sur internet.

* *

*

Ma thèse s'articule en six chapitres dont voici un brève description :

1. Rappels

Les notions fondamentales de la théorie des graphes y sont brièvement introduites. J'insiste en particulier sur les relations entre (0,1)-matrice et graphe.

2. Recherche en place des occurrences d'un ensemble reconnaissable

Je présente dans ce chapitre un algorithme pour rechercher toutes les occurrences d'un ensemble reconnaissable dans un texte. Je suppose que l'ensemble reconnaissable est donné sous forme d'un automate déterministe.

3. Triangularisation d'une (0,1)-matrice

J'aborde dans ce troisième chapitre le problème général de la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes. Je discute de l'importance des arrangements linéaires triangulaires pour ce problème et montre en particulier que décider s'il existe un arrangement linéaire triangulaire d'un graphe est un problème NP-complet.

J'examine ensuite le problème général dans le cas des sous-matrices en utilisant des techniques de couplages dans les graphes bipartis. Je montre que décider s'il existe un sous-graphe exact-triangulaire de taille K dans un graphe biparti est un problème NP-complet.

Enfin, j'introduis la notion d'arrangement linéaire des voisinages dans un graphe et discute des liens avec le problème général. Je montre ensuite que ce problème est NP-complet.

4. Généralisation et résultats complémentaires

Ce chapitre est le prolongement du précédent. Je présente plusieurs problèmes qui généralisent le problème général de la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes.

J'étudie dans un premier temps la complexité des problèmes de silhouette. Ce travail est prolongé par l'étude de la surface d'une (0,1)-matrice. Je montre en particulier que le problème de décision associé à la minimisation de surface d'une (0,1)-matrice par permutation des lignes est NP-complet.

Ensuite, je présente une généralisation du problème basée sur l'arrangement d'une famille de couples de sous-ensembles. Je montre que ce problème est NP-complet dans le cas général. Enfin, j'examine plusieurs problèmes de partition en «formes triangulaires» (inférieurement triangulaire, exactement triangulaire et supérieurement triangulaire) dans les (0,1)-matrices.

5. Structures secondaires d'ARN et famille de 2-intervalles

J'aborde dans ce chapitre différents problèmes liés aux familles de 2-intervalles en relation avec les structures secondaires d'ARN.

Je montre que la recherche d'un sous-famille de 2-intervalles de cardinalité maximale pour différents modèles de comparaison est un problème NP-complet. Je présente

ensuite différentes variantes de ce problème. Enfin, je montre que la recherche d'un motif dans une famille de 2-intervalles est un problème NP-complet pour différents modèles de comparaison.

6. Prédiction des introns auto-catalytiques de groupe I

Le sixième chapitre est consacré à la prédiction des introns auto-catalytiques de groupe I dans les séquences génomiques.

Après quelques brefs rappels sur la structure des ARN, je décris tout d'abord l'algorithme utilisé par le programme *citron* et je présente ensuite un nouvel algorithme et le programme *orange*. Pour conclure, je discute des résultats du programme orange ainsi que quelques évolutions possibles.

Chapitre 2

Rappels

Sommaire

2.1 Graphe 11
2.2 Matrice 16
2.3 Graphe et matrice 19

2.1 Graphe

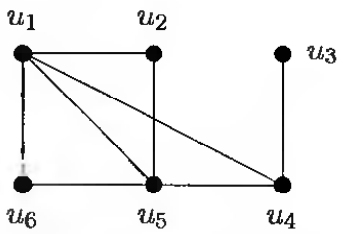
Nous rappelons quelques notions élémentaires sur les graphes [Har71, Bol79, GM79, Gol80, PS82, Jun99, Die00] et précisons les notations utilisées dans la suite.

Un *graphe* G est représenté par un ensemble fini[1] de sommets, généralement noté V, et par un ensemble de paires de sommets *non ordonnées* généralement noté E. Un élément $e = \{u,v\} \in E$ est appelé une *arête* dont les *extrémités* sont les sommets u et v. Nous disons que les sommets u et v sont *incidents* à l'arête e et que u et v sont *adjacents*. L'*ordre* d'un graphe $G = (V,E)$ désigne son nombre de sommets, c'est-à-dire $|V|$.

graphe
sommet
arête

Par exemple, le graphe suivant est un graphe d'ordre 6. Son ensemble de sommets V et son ensemble d'arêtes E sont donnés par :

$$V = \{u_1,u_2,u_3,u_4,u_5,u_6\}$$
$$E = \{\{u_1,u_2\},\{u_1,u_4\},\{u_1,u_5\},\{u_1,u_6\},\{u_2,u_5\},\{u_3,u_4\},\{u_4,u_5\},\{u_5,u_6\}\}$$



Une *boucle* dans un graphe $G = (V,E)$ est une arête de la forme $\{u,u\}$. Un *multigraphe* est un graphe où une arête peut être répétée dans E (E est dans ce cas un multi-ensemble). Un graphe est dit *simple* si :

boucle
multigraphe
graphe simple

1. Les graphes infinis sont également étudiés en théorie des graphes. Cependant, nous ne considérerons dans la suite que des graphes finis.

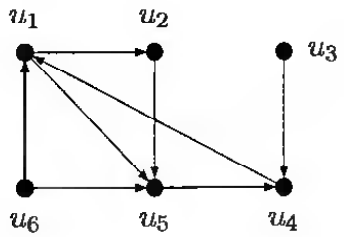
1. il est sans boucle;
2. il n'y a jamais plus d'une arête entre deux sommets quelconques.

Sauf mention contraire explicite, nous ne considérons dans la suite que des graphes (non orientés) simples.

graphe orienté Un *graphe orienté* est un graphe où toutes les arêtes sont orientées. Plus formellement, un graphe orienté D est une paire $D = (V,A)$ où V est un ensemble de sommets et A est un ensemble de paires ordonnées de sommets appelées *arcs*. Un arc orienté de u vers v est noté (u,v)[2].

arc

orientation Une *orientation* d'un graphe (non orienté) G est obtenue en orientant toutes les arêtes de G. Par exemple, le graphe orienté suivant est une orientation du graphe non orienté précédent.



2.1.1 Voisinage et degré

L'ensemble des sommets adjacents à un sommet u est noté $N(u)$:

$$N(u) = \{v \mid v \in V \text{ et } \{u,v\} \in E\}$$

degré L'ensemble $N(u)$ est également appelé le *voisinage* du sommet u. Le *degré* d'un sommet u, noté $\deg(u)$, est le nombre d'arêtes incidentes à u :

$$\deg(u) = |N(u)|$$

Un sommet de degré 0 est appelé un *sommet isolé*. Le degré maximum de G, noté $\Delta(G)$, est le degré maximum de l'un de ses sommets. De même, le degré minimum de G, noté $\delta(G)$, est le degré minimum de l'un de ses sommets.

$$\Delta(G) = \max_{u \in V} \deg(u)$$
$$\delta(G) = \min_{u \in V} \deg(u)$$

Dans le cas des graphes simples d'ordre n, nous avons toujours :

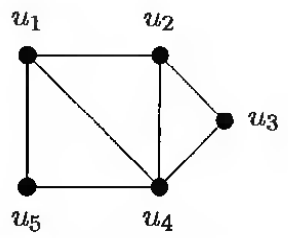
$$0 \leq \delta(G) \leq \Delta(G) \leq n - 1$$

degré sortant Si G est un graphe orienté, le *degré sortant* d'un sommet u, noté $d^+(u)$, est le
degré entrant nombre d'arcs sortants. De même, le *degré entrant*, noté $d^-(u)$, est le nombre d'arcs entrants.

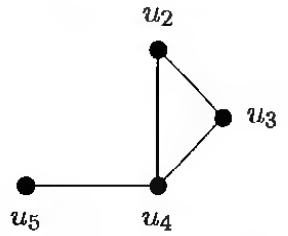
2.1.2 Sous-graphe

Soient $G = (V,E)$ un graphe et $V' \subseteq V$ un sous-ensemble de sommets. L'ensemble des arêtes $e \in E$ dont les sommets extrémités appartiennent à V' est noté $E|V'$. Le

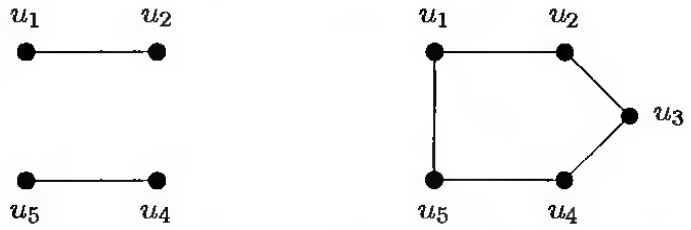
graphe $(V',E|V')$ est appelé le *sous-graphe induit* par V' et est noté $G|V'$. Par exemple, si G est le graphe suivant,



alors le sous-graphe induit $G|V'$ pour $V' = \{u_2,u_3,u_4,u_5\}$ est donné par :



Un graphe $G' = (V',E')$ est un *sous-graphe* de G si $V' \subseteq V$ et $E' \subseteq E$. Si $G' = (V',E')$ est un sous-graphe de G et $V = V'$, alors G' est appelé un *sous-graphe couvrant* de G. Par exemple, les deux graphes suivants sont des sous-graphes du graphe G précédent; le second est un sous-graphe couvrant de G. ***sous-graphe*** ***couvrant***



Soit $G = (V,E)$ un graphe. Pour tout $E' \subseteq E$, nous notons $V|E'$ l'ensemble des sommets incidents à au moins une arête de E' et $G[E']$ le sous-graphe $G' = (V|E',E')$.

2.1.3 Chemin, circuit et connexité

Soit $G = (V,E)$ un graphe d'ordre n. Un *chemin* P de longueur k est une séquence de k sommets : ***chemin***

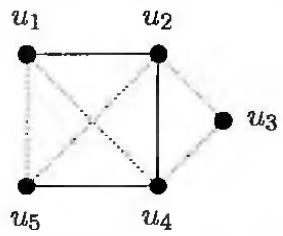
$$P = \{u_1,u_2,\ldots,u_k\}$$

telle que $\{u_i,u_{i+1}\} \in E$ pour $1 \leq i \leq k-1$. Les sommets u_1 et u_k sont appelés les *sommets extrémités* du chemin. Un *chemin élémentaire* est un chemin où aucun sommet n'est répété. Un chemin est *fermé* si $k > 1$ et $u_k = u_1$. Un chemin fermé où aucun sommet n'est répété (sauf u_1 et u_k) est appelé un *circuit*. ***circuit***

Un chemin élémentaire est *hamiltonien* s'il passe par tous les sommets du graphe. Un *circuit hamiltonien* est un circuit qui passe par tous les sommets du graphe. Par ***hamiltonien***

2. Certains auteurs préfèrent la notations $\overrightarrow{uv}$ pour l'arc de u vers v.

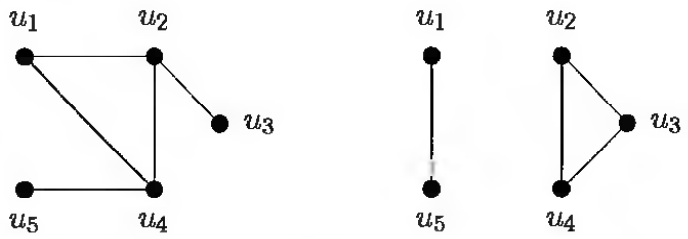
exemple, un circuit hamiltonien est représenté en pointillés sur le graphe suivant.



Deux sommets u_i et u_j d'un graphe G sont *connectés* s'il existe un chemin de sommets extrémités u_i et u_j. Si tous les sommets d'un graphe G sont deux à deux connectés, alors le graphe G est appelé un *graphe connexe*.

connexe

Par exemple, sur les graphes suivants, celui de gauche est connexe tandis que celui de droite ne l'est pas.



Pour tout sommet u, nous considérons $\{u\}$ comme un chemin trivial de longueur 0, de sorte que tout sommet est connecté avec lui même. Aussi, la connexion est une relation d'équivalence sur les sommets de G. Une classe d'équivalence de cette relation est appelée une *composante connexe* de G. Le graphe G est *connexe* si et seulement si son ensemble de sommets V est une classe d'équivalence. Les composantes qui ne contiennent qu'un seul sommet sont appelées les *sommets isolés*.

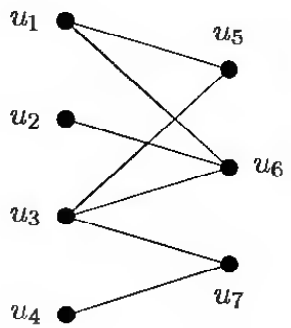
connexe

Un *point d'articulation* est un sommet dont la suppression sépare le graphe en parties non connexes. Un graphe sans point d'articulation est appelé un *graphe biconnexe*.

2.1.4 Quelques graphes particuliers

graphe biparti

Un graphe $G = (V,E)$ est un *graphe biparti* s'il existe une bipartition $V = V_1 \cup V_2$ telle que chaque arête est incidente à exactement un sommet de V_1 et un sommet de V_2. Lorsque la bipartition est connue, nous notons la plupart du temps le graphe biparti $G = (V_1 \cup V_2, E)$. En général, un graphe biparti est représenté en plaçant les sommets des ensembles V_1 et V_2 sur deux droites parallèles.



Un *graphe complet* est un graphe d'ordre n où tous les sommets sont adjacents deux à deux. Un graphe complet d'ordre n est noté K_n (le K est en l'honneur de Kuratowski, un pionnier de la théorie des graphes). Le *graphe biparti complet* $K_{m,n}$ est un graphe biparti de bipartition $V_1 \cup V_2$, $|V_1| = m$, $|V_2| = n$, tel que chaque sommet de V_1 est adjacent à tous les sommets de V_2, c'est-à-dire : ***biparti complet***

$$E = \{\{u,v\} \mid u \in V_1 \ \wedge \ v \in V_2\}.$$

Si tous les sommets d'un graphe G ont le même degré k, alors G est appelé un *graphe k-régulier*. Le graphe K_n est $(n-1)$-régulier, tandis que le graphe $K_{m,n}$ est régulier si et seulement si $m = n$ (il est dans ce cas n-régulier). Un *k-facteur* est un sous-graphe couvrant k-régulier. ***k-régulier*** ***k-facteur***

Un *arbre* $T = (V,E)$ est un graphe connexe sans cycle. Une *forêt* est un graphe acyclique[3]. Un sous-graphe couvrant connexe et sans cycle d'un graphe G est appelé un *arbre couvrant* de G. Un *arbre enraciné* est un arbre dans lequel l'un des sommet, généralement appelé la *racine* de l'arbre, se distingue des autres. ***arbre*** ***forêt*** ***arbre couvrant enraciné***

2.1.5 Définitions complémentaires

Deux graphes $G = (V,E)$ et $G' = (V',E')$ sont *isomorphes*, noté $G \cong G'$, s'il existe une bijection φ de V dans V' telle que $\{u,v\} \in E$ si et seulement si $\{\varphi(u),\varphi(v)\} \in E$ pour tout $u,v \in V$. Autrement dit, deux graphes sont isomorphes s'ils sont identiques modulo un renommage des sommets. ***graphes isomorphes***

Soit $G = (V,E)$ un graphe. Le *graphe complémentaire* de G est le graphe[4] $\overline{G} = (V,\overline{E})$ défini par : ***graphe complémentaire***

$$\overline{E} = \{\{u,v\} \mid u \in V, v \in V, u \neq v \text{ et } \{u,v\} \notin E\}$$

Un sous-ensemble $V' \subseteq V$ de q sommets est appelé une *q-clique* (ou plus simplement une clique) si le sous-graphe induit par V' est complet. Autrement dit, V' est une q-clique si $G|V' \cong K_q$. Un sommet est donc une 1-clique. Une clique V' est maximale s'il n'existe aucune clique V'' de G qui contient strictement V'. La cardinalité maximale d'une clique de G est notée $\omega(G)$. ***q-clique***

Une *couverture par cliques* de taille k est une partition des sommets de G : ***couverture par cliques***

$$V = V_1 \cup V_2 \cup \cdots \cup V_k$$

telle que V_i, $1 \leq i \leq k$, est une clique. La taille de la plus petite couverture par cliques de G est notée $\theta(G)$.

Un *ensemble stable* est un sous-ensemble de sommets $V' \subseteq V$ non connectés deux à deux. La cardinalité maximale d'un ensemble stable du graphe G est notée $\alpha(G)$. Remarquons qu'un sous-ensemble $V' \subseteq V$ est un ensemble stable dans V si et seulement si V' est une clique dans $\overline{G}$. ***ensemble stable***

Un *k-coloriage* de G est une partition des sommets de G : ***k-coloriage***

$$V = V_1 \cup V_2 \cup \cdots \cup V_k$$

telle que V_i, $1 \leq i \leq k$, est un ensemble stable. Le plus petit entier k pour lequel il existe un k-coloriage de G, noté $\chi(G)$, est appelé le *nombre chromatique* de G. ***nombre chromatique***

3. Une forêt est donc un ensemble d'arbres disjoints.

4. Le graphe complémentaire est parfois noté G^c.

2.2 Matrice

2.2.1 Introduction

matrice

Une *matrice* est un tableau d'éléments rectangulaire. De façon générale, nous notons les matrices par des lettres majuscules (A,B, $\cdots$) et la composante de la matrice située à la ligne i et à la colonne j par la lettre minuscule associée ($a_{i,j}$,$b_{i,j}$, $\cdots$). Par exemple,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.$$

Si A est une matrice de m lignes et n colonnes, nous disons que A est de taille m par n. Lorsque $m = n$, nous disons que A est une matrice carrée de taille n ou, plus simplement, que A est une matrice d'ordre n. ans la suite, nous appelons (0,1)-matrice une matrice dont toutes les composantes ont des valeurs booléennes.

transposée

La *transposée* d'une matrice A de taille m par n est la matrice notée A^T de taille n par m obtenue en échangeant les lignes et les colonnes de A. Par exemple,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \qquad A^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix}.$$

symétrique

Une matrice A carrée est dite *symétrique* si $A = A^T$.

complémentaire

Soit $A = [a_{i,j}]$ un (0,1)-matrice de taille m par n. La *matrice complémentaire* de A, notée $\overline{A} = [\overline{a}_{i,j}]$ est une (0,1)-matrice de même taille que A définie par $\overline{a}_{i,j} = 1$ si et seulement si $a_{i,j} = 0$ pour $1 \leq i \leq m$ et $1 \leq j \leq n$.

2.2.2 Quelques matrices carrées particulières

diagonale

Une *matrice diagonale* est une matrice dont toutes les composantes hors diagonale sont nulles:

$$\mathrm{diag}(a_{11},a_{22},\ldots,a_{nn}) = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}.$$

identité

La *matrice identité* d'ordre n, notée $\mathbb{I}_n$, est la (0,1)-matrice diagonale d'ordre n dont toutes les composantes situées sur la diagonale valent 1 :

$$\mathbb{I}_n = \mathrm{diag}(1,1,\ldots,1) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Nous utilisons également fréquemment une (0,1)-matrice, notée $\mathbb{C}$, définie par :

$$\mathbb{C}_n = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & \iddots & \vdots & \vdots \\ 1 & \cdots & 0 & 0 \end{bmatrix}.$$

Enfin, nous notons $\mathbf{1}_{m,n}$ (resp. $\mathbf{0}_{m,n}$) la (0,1)-matrice dont toutes les composantes valent 1 (resp. 0).

2.2.3 Matrice de permutation

Une *matrice de permutation* d'ordre n comporte exactement un élément non nul sur chaque ligne et colonne. Par exemple, la matrice P suivante est une matrice de permutation : ***permutation***

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Une matrice de permutation P d'ordre n satisfait les équations matricielles:

$$PP^T = P^T P = \mathbb{I}_n.$$

Par exemple, $\mathbb{I}_n$ et $\mathbb{C}_n$ sont des matrices de permutation d'ordre n.

Remarquons que l'action à gauche d'une matrice de permutation P sur une matrice A correspond à un réarrangement de ses lignes, tandis que l'action à droite de P sur A correspond à un réarrangement de ses colonnes.

2.2.4 Opérations et relations

Soient $A = [a_{i,j}]$ et $B = [a_{i,j}]$ deux matrices de taille m par n. Nous notons $A \leq B$ (resp. $A \geq B$) si et seulement si $a_{i,j} \leq b_{i,j}$ (resp. $a_{i,j} \geq b_{i,j}$) pour $1 \leq i \leq m$ et $1 \leq j \leq n$.

DÉFINITION 2.2.1 (ROTATIONS GAUCHE ET DROITE) *Soient $A = [a_{i,j}]$ une (0,1)-matrice de taille m par n. La* rotation gauche *(resp.* rotation droite*) de A, notée* rotg(A) *(resp.* rotd(A)*), est une (0,1)-matrice $B = [b_{i,j}]$ de taille n par m définie par $b_{n-j+1,i} = a_{i,j}$ (resp. $b_{j,n-i+1} = a_{i,j}$) pour tout $1 \leq i \leq m$ et $1 \leq j \leq n$.* ***rotation***

EXEMPLE 1 *Soit A la matrice d'ordre 4 définie par*

$$A = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}.$$

Alors,

$$\mathrm{rotg}(A) = \begin{bmatrix} d & h & l & p \\ c & g & k & o \\ b & f & j & n \\ a & e & i & m \end{bmatrix} \quad \textit{et} \quad \mathrm{rotd}(A) = \begin{bmatrix} m & i & e & a \\ n & j & f & b \\ o & k & g & c \\ p & l & h & d \end{bmatrix}.$$

REMARQUE 2.2.2 Soit A une (0,1) matrice de taille m par n. Alors, $\mathrm{rotg}(A) = (A\mathbb{C}_n)^T$ et $\mathrm{rotd}(A) = (\mathbb{C}_m A)^T$. ◇

2.2.5 Matrices triangulaires

Nous notons $\mathbb{L}_n$ et $\mathbb{U}_n$ les (0,1)-matrices d'ordre n définies par :

$$\mathbb{L}_n = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & & \cdots & a_{nn} \end{bmatrix}, \qquad a_{i,j} = 1 \text{ pour } 1 \leq j \leq i \leq n,$$

et

$$\mathbb{U}_n = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & & \cdots & a_{nn} \end{bmatrix}, \qquad a_{i,j} = 1 \text{ pour } 1 \leq j \leq i \leq n.$$

De plus, nous notons $\mathbb{D}_n$ et $\mathbb{F}_n$ les (0,1)-matrice d'ordre n définies par:

$$\mathbb{D}_n = \mathrm{rotd}(\mathbb{U}_n)$$
$$\mathbb{F}_n = \mathrm{rotd}(\mathbb{L}_n).$$

Une matrice $A = [a_{i,j}]$ est *triangulaire inférieure* si $a_{i,j} = 0$ pour $i < j$; elle est *triangulaire supérieure* si $a_{i,j} = 0$ pour $i > j$. En d'autres termes, A est triangulaire inférieure si $A \leq \mathbb{L}_n$ et triangulaire supérieure si $A \leq \mathbb{U}_n$.

REMARQUE 2.2.3 Soit A une (0,1)-matrice d'ordre n. Les conditions suivantes sont équivalentes :

1. $A \leq \mathbb{L}_n$;
2. $\mathrm{rotd}(A) \leq \mathbb{F}_n$;
3. $\mathrm{rotg}(A) \leq \mathbb{D}_n$;
4. $\mathrm{rotg}^2(A) \leq \mathbb{U}_n \quad (\mathrm{rotd}^2(A) \leq \mathbb{U}_n)$.

◇

inf-triangulaire
sup-triangulaire

DÉFINITION 2.2.4 (MATRICE INF-TRIANGULAIRE SUP-TRIANGULAIRE) *Soit A une (0,1)-matrice d'ordre n. Alors, la matrice A est dite* inf-triangulaire *si $A \leq \mathbb{T}$ pour $\mathbb{T} \in \{\mathbb{L}_n, \mathbb{D}_n, \mathbb{F}_n, \mathbb{U}_n\}$; la matrice A est dite* sup-triangulaire *si $A \geq \mathbb{T}$ pour $\mathbb{T} \in \{\mathbb{L}_n, \mathbb{D}_n, \mathbb{F}_n, \mathbb{U}_n\}$.*

2.2.6 Compléments

inverse

L'*inverse* d'une matrice A d'ordre n est une matrice (si elle existe) notée A^{-1} telle que $AA^{-1} = \mathbb{I}_n = A^{-1}A$. Une matrice sans inverse est dite *non inversible* ou *singulière*, sinon elle est dite *inversible* ou *non singulière*.

THÉORÈME 2.2.5 *Une matrice A d'ordre n est non inversible si et seulement si* $\det(A) = 0$.

Soit $A = [a_{i,j}]$ une (0,1)-matrice de taille m par n. Deux composantes non nulles $a_{i,j}$ et $a_{i',j'}$ de A sont *indépendantes* si $i \neq i'$ et $j \neq j'$.

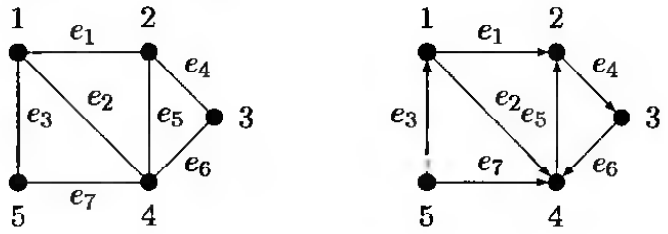
THÉORÈME 2.2.6 *Soit A une (0,1)-matrice de taille m par n. Le nombre minimal de lignes et de colonnes nécessaires pour couvrir tous le éléments non nuls de A est égal au nombre maximum de 1 indépendants dans A.*

Le nombre maximum de 1 indépendants dans A est noté $\rho(A)$. Peu de propriétés sur les (0,1)-matrices sont conservées par permutation arbitraire des lignes et des colonnes. Remarquons cependant que $\rho(A) = \rho(PAQ^T)$ où P et Q sont des matrices de permutation quelconques.

2.3 Graphe et matrice

2.3.1 Introduction

Nous utiliserons tout au long de cette section le graphe non orienté G et le graphe orienté G' définis ci-après (par soucis de clarté, nous identifions l'ensemble des n sommets aux entiers de 1 à 5.



2.3.2 Matrice d'adjacence

Soit $G = (V,E)$ un graphe d'ordre n simple (c'est à dire qu'il existe au plus une arête entre deux sommets quelconques et sans boucle). La *matrice d'adjacence*[5] de G est une (0,1)-matrice symétrique $A = [a_{i,j}]$ où chaque ligne correspond à un sommet de G, chaque colonne correspond à un sommet de G et $a_{i,j} = a_{j,i} = 1$ si et seulement si $\{i,j\} \in E$. Nous notons parfois $A(G)$ la matrice d'adjacence du graphe G. Puisque G est simple, nous avons toujours $a_{i,i} = 0$. ***matrice d'adjacence***

Si G est un graphe orienté, nous avons $a_{i,j} = 1$ si et seulement si $(u,v) \in E$. Remarquons que la matrice d'adjacence d'un graphe orienté n'est en général non symétrique.

Par exemple, les matrice d'adjacence A et A' des graphes G et G' définis en introduction sont données par :

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad A' = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

5. ou *matrice d'incidence sommets-sommets.*

REMARQUE 2.3.1 Soient G un graphe (orienté) et A sa (0,1)-matrice d'adjacence. La composante (i,j) de la matrice A^k est le nombre de chemins (orientés) du sommet i au sommet j.

irréductible

Une matrice A d'ordre n est dite *irréductible* s'il existe une matrice de permutation P d'ordre n telle que :

$$PAP^T = \begin{bmatrix} A_1 & 0 \\ A_{12} & A_2 \end{bmatrix}$$

où A_1 et A_2 sont des matrices carrées d'ordre au moins 1. Si la matrice A n'est pas réductible, alors elle est dite irréductible. Remarquons qu'une matrice d'ordre 1 est irréductible.

LEMME 2.3.2 *Soient G un graphe d'ordre n et A sa matrice d'adjacence. Le graphe G est connexe si et seulement si la matrice A est irréductible.*

L'isomorphisme entre deux graphes a une interprétation directe sur les matrices d'adjacence associées.

LEMME 2.3.3 *Soient G un graphe d'ordre n de matrice d'adjacence A et G' un graphe d'ordre n de matrice d'adjacence A'. Les graphes G et G' sont isomorphes ($G \cong G'$) si et seulement si il existe une matrice de permutation P d'ordre n telle que $PAP^T = A'$.*

2.3.3 Graphe biparti

Les graphes bipartis et les matrices d'adjacence associées satisfont un certain nombre de propriétés intéressantes [ADH98].

Ils sont dans un premier temps caractérisés par une forme particulière de leur matrice d'adjacence. Soient G un graphe d'ordre n. Le graphe G est biparti si et seulement si il existe une matrice de permutation P d'ordre n telle que :

$$PAP^T = \begin{bmatrix} \mathbf{0}_{n-p,n-p} & B \\ B^T & \mathbf{0}_{p,p} \end{bmatrix}$$

matrice d'adjacence réduite

où B est une (0,1)-matrice d'ordre au moins 1 appelée la *matrice d'adjacence réduite* de G. Remarquons que B est en général une matrice non symétrique.

2.3.4 Matrice d'incidence sommets-arcs

Soit $G = (V,E)$ un graphe orienté. La *matrice d'incidence sommets-arcs* de G est une $(-1,0,1)$-matrice A où chaque ligne correspond à un sommet de G, chaque colonne correspond à un arc de E et la colonne $e = (i,j) \in E$ a tous ses termes nuls sauf $a_{i,e} = 1$ et $a_{j,e} = -1$.

Par exemple la matrice d'incidence A sommets-arcs du graphe G' défini en introduction est donnée par :

$$A = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Une matrice carrée d'ordre n est dite *unimodulaire* si son déterminant est soit $+1$, soit -1. Une matrice A de taille m par n est dite *totalement unimodulaire* si et seulement si toutes les matrices carrées extraites de A sont unimodulaires. ***unimodulaire***

LEMME 2.3.4 ([GM79]) *La matrice d'incidence sommets-arcs d'un graphe orienté est totalement unimodulaire.*

2.3.5 Matrice d'incidence sommets-arêtes

Soit $G = (V,E)$ un graphe d'ordre n. La *matrice d'incidence sommets-arêtes* de G est une (0,1)-matrice A où chaque ligne correspond à un sommet i de G, chaque colonne correspond à une arête $e = \{i,j\}$ de E et la colonne e a tous ses éléments nuls sauf $a_{i,e}$ et $a_{j,e}$. Autrement dit, chaque colonne contient exactement deux éléments non nuls qui correspondent aux sommets extrémités de l'arête.

Par exemple le graphe d'incidence sommets-arêtes A du graphe G défini en introduction est donnée par:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Contrairement à la matrice d'incidence sommets-arcs, la matrice d'incidence sommets-arêtes n'est pas en général totalement unimodulaire. Par exemple la matrice A définie par :

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 2 & 2 \end{bmatrix}$$

a pour déterminant -2. En fait, il existe une caractérisation très précise des matrices d'incidence sommets-arêtes totalement unimodulaires.

LEMME 2.3.5 ([GM79]) *La matrice d'incidence sommets-arêtes d'un graphe G est totalement unimodulaire si et seulement si G est biparti.*

Chapitre 3

Localisation

Sommaire

3.1 Définitions **23**
3.2 Automate et recherche de motifs **23**
3.3 Représentation matricielle **27**
3.4 Résolution «*en place*» de système d'équations **31**
3.5 Occurrences d'un sous-ensemble reconnaissable **35**
3.6 Mise en œuvre **50**

Nous proposons dans ce chapitre un algorithme efficace pour calculer toutes les occurrences d'un sous-ensemble reconnaissable $X \subseteq \Sigma^*$ dans un mot.

Lorsque X est un ensemble fini, plusieurs algorithmes ont été proposés. Par exemple, l'algorithme de Aho et Corasick [AC75] basé sur une extension de l'algorithme de Knuth, Morris et Pratt [KMP77], algorithme de Commentz et Walter [CW79] basé sur une extension de l'algorithme de Boyer et Moore [BM77]. Fondamentalement, ces deux algorithmes utilisent un arbre à partir des mots de X où chaque transition est étiquetée par une lettre de l'alphabet. Le nombre de sommets de ces arbres est linéaire en la somme des longueurs des mots.

Nous proposons de travailler directement sur une structure qui est fondamentalement l'automate fini déterministe qui reconnaît X et où chaque état est étiqueté par un ensemble de positions. Ce faisant, nous réduisons le nombre d'états tout en ayant la possibilité de rechercher toutes les occurrences d'un sous-ensemble rationnel.

3.1 Définitions

3.2 Automate et recherche de motifs

3.2.1 Sous-ensemble reconnaissable

Étant donnés un sous-ensemble reconnaissable X et un mot u, le problème revient à chercher tous les préfixes de u qui appartiennent au langage Σ^*X. La méthode classique construit tout d'abord un automate reconnaissant Σ^*X, puis recherche les préfixes de u reconnus par cet automate [Aho90, CR94].

Nous notons e l'*expression rationnelle* $X = L(e)$. Il est possible de construire un *expression rationnelle*

automate déterministe pour le langage $\Sigma^* X$. Cependant, le nombre d'états de cet automate peut être exponentiel en fonction de la taille de l'expression rationnelle e. Par exemple, l'automate déterministe minimal qui reconnaît $(a+b)^*(a+b)^k$ possède 2^k états. Inversement, l'utilisation d'un automate non déterministe complique la recherche des occurrences. Aussi, une approche intermédiaire est souvent utilisée:

1. construction d'un automate non déterministe particulier qui reconnaît $X = L(e)$ à partir de e;
2. recherche des facteurs de u qui appartiennent à l'ensemble représenté par l'expression rationnelle e à l'aide de cet automate.

Nous décrivons tout d'abord succinctement la construction d'un automate non déterministe particulier à partir d'une expression rationnelle. La *taille* d'une expression rationnelle e, notée $|e|$, est le nombre de symboles dans e. Un automate est dit *normalisé* s'il vérifie les conditions suivantes:

normalisé

1. il existe un unique état initial et un unique état terminal, et ces deux états sont distincts;
2. aucune transition ne pointe sur l'état initial et aucune transition ne sort de l'état terminal;
3. tout état est soit l'origine d'exactement une transition étiquetée par une lettre $a \in \Sigma$, soit l'origine d'au plus deux transitions étiquetées par le mot vide ε.

THÉORÈME 3.2.1 ([THO68]) *Pour toute expression rationnelle e de taille m, il existe un automate normalisé reconnaissant $L(e)$ dont le nombre d'états est au plus $2m$.*

Remarquons que le nombre de transitions étiquetées par une lettre de Σ est au plus m et le nombre de transitions étiquetées par le mot vide ε est au plus $4m$. La preuve du théorème 3.2.1 est constructive et procède par induction sur la taille de e (voir [CH97] ou [BBC92] pour une preuve détaillée). Nous notons $\mathcal{A}(e)$ l'automate normalisé qui reconnaît $L(e)$.

La recherche des occurrences des facteurs de u qui appartiennent à l'ensemble représenté par l'expression rationnelle e se fait en deux étapes:

1. construction de l'automate normalisé $\mathcal{A}(e)$;
2. pour chaque préfixe de u, calcul des états de $\mathcal{A}(e)$ accessibles depuis l'état initial.

La première étape constitue le prétraitement, tandis que la seconde est la recherche proprement dite.

Le calcul de l'automate $\mathcal{A}(e)$ s'effectue en temps et en espace $O(m)$. La taille de l'automate $\mathcal{A}(e)$ est $O(m)$. Nous pouvons représenter $\mathcal{A}(e)$ par deux tables indicées par son ensemble d'états: la première table contient pour tout $q \in \mathcal{Q}$ l'unique état accessible par une lettre $a \in \Sigma$, tandis que la seconde contient pour tout $q \in \mathcal{Q}$ au plus deux états accessibles par le mot vide ε. L'étape de recherche peut être réalisée en temps $O(|\mathcal{Q}| \cdot |u|)$, où $\mathcal{Q}$ est l'ensemble des états de l'automate $\mathcal{A}(e)$. Puisque $|\mathcal{Q}| = O(|e|)$, l'algorithme est $O(|u| \cdot |e|)$ en temps.

La recherche des occurrences des facteurs de u dénotés par e consiste à simuler à l'aide de $\mathcal{A}(e)$ un automate déterministe qui reconnaît $\Sigma^* L(e)$. Pour cela, nous

conservons, pour chaque étape, l'ensemble des états accessibles depuis l'état initial par un chemin dont l'étiquette est la partie déjà lue du mot d'entrée.

De même que pour la déterminisation d'un automate avec ε-transitions, l'étape de recherche utilise la notion de *clôture* d'un ensemble d'états: si $\mathcal{Q}'$ est un ensemble d'états, sa clôture est l'ensemble des états accessibles depuis un état de $\mathcal{Q}'$ par un chemin étiqueté par le mot vide ε (algorithme `clôture`). À partir de la clôture d'un ensemble d'états, nous pouvons facilement calculer toutes les transitions étiquetées par une lettre $a \in \Sigma$ (algorithme `transitions`).L'algorithme de recherche des occurrences des facteurs de u dénotés par e utilise pour chaque lettre les fonctions `clôture` et `transitions` pour conserver l'ensemble des états accessibles depuis l'état initial par le préfixe courant. *clôture*

Algorithm 1: Clôture de $\mathcal{Q}'$.

```
clôture
Entrée: Un sous-ensemble Q' ⊂ Q.
Sortie: La clôture de Q'.
début
1  T ← Q'
2  R ← Q'
3  tant que T ≠ ∅ faire
4      choisir q dans T
5      pour q' tel que q' = q · ε faire
6          si q' ∉ R alors
7              R ← R ∪ {q'}
8              T ← T ∪ {q'}
9  retourner T
fin
```

Algorithm 2: Transitions de $\mathcal{Q}'$ par a.

```
transitions
Entrée: Un sous-ensemble Q' ⊂ Q, et une lettre a ∈ Σ..
Sortie: Ensemble des états accessibles de Q' par a.
début
1  R ← ∅
2  pour q ≠ Q' faire
3      si il existe q' ∈ Q tel que q' = q · a alors
4          T ← R ∪ {q'}
5  retourner R
fin
```

Comme nous l'avons vu, il est toujours possible de déterminiser l'automate qui reconnaît $\Sigma^* L(e)$. La recherche des occurrences se simplifie puisqu'il n'existe à tout moment qu'un seul état courant: *i.e.* chaque lettre du mot u induit au plus une

Algorithm 3: Calcul des préfixes d'un mot u qui appartiennent à $\Sigma^* L(e)$, où e est une expression rationnelle.

```
recherche-occurrences
Entrée: Une expression rationnelle e, et un mot u de longueur n.
Sortie: Les préfixes de u qui appartiennent à Σ*L(e).
début
1  construction de l'automate A(e)
2  C ← clôture(q_-)
3  pour i de 1 à n faire
4      S ← clôture(transitions(S,u_i))
5      si q_+ ∈ S alors
6          afficher i
fin
```

transition. Plusieurs techniques ont été proposées pour réduire la taille de l'automate [ASU86]. Par exemple, il est possible de faire une *évaluation paresseuse* des transitions: les transitions sont calculées à la demande et conservées dans un système de cache. Si le cache est plein, quelques (ou toutes) transitions sont effacées. M. Mohri [Moh97] utilise une *fonction d'erreur* pour construire efficacement un automate déterministe qui reconnaît $\Sigma^* X$ à partir d'un automate déterministe qui reconnaît X.

3.2.2 Ensemble fini de mots

Lorsque X est un ensemble fini de mots, la recherche est dite *multiple*. De nombreux algorithmes efficaces ont été proposés pour la recherche multiple. Nous notons $|X| = \sum_{v \in X} |v|$.

L'algorithme le plus connu pour la recherche d'un ensemble de mots est celui de Aho-Corasick [AC75]. Cet algorithme est basé sur une extension de l'algorithme de Knuth-Morris-Pratt [KMP77] à la recherche multiple. Il consiste à construire un automate $AC(X)$ à partir de tous les mots de X, ainsi qu'une fonction d'échec f. La structure sous-jacente de l'automate $AC(X)$ est un arbre: chaque chemin depuis l'état initial est étiqueté par un préfixe d'un mot de X. Toutes les feuilles sont terminales, et un sommet interne est terminal lorsqu'un mot de X est un facteur propre d'un autre mot de X. La fonction d'échec est utilisée pendant la recherche lorsqu'une transition n'existe pas depuis un état q donné: si le chemin de l'état initial à q est étiqueté par v, elle envoie q sur l'état correspondant au plus long suffixe propre de v qui soit préfixe d'un mot de X. La construction de l'automate $AC(X)$ et la calcul de la fonction d'échec f peuvent s'effectuer en temps $O(|X|)$. La recherche des occurrences des mots de X dans un mot u s'effectue en temps $O(|u|)$. Il est possible de compléter le fonction de transition de l'automate $AC(X)$, l'étape de recherche des occurrences est ensuite linéaire. Une optimisation de l'algorithme de Aho-Corasick est proposée dans [AYS84].

Une autre méthode consiste à combiner l'algorithme de Aho-corasick avec l'ap-

proche Boyer-Moore [BM77]. C'est par exemple le cas de l'algorithme de Commentz-Walter [CW79]. Il consiste à construire un automate $CW(X)$ du type Aho-Corasick pour les mots inversés de X. La construction de l'automate de recherche $CW(X)$ peut s'effectuer en temps $O(|X|)$. La complexité de la recherche des occurrences de X dans un mot u est $\theta(|u| \cdot |X|)$ en temps. Cependant, elle reste plus efficace en pratique que l'algorithme de Aho-Corasick lorsque les motifs sont courts [Aho90]. L'algorithme de Fan-Su [FS93] utilise également une approche Boyer-Moore. Une approche Boyer-Moore-Horspool est utilisée dans l'algorithme de Wu-Mamber [WM94].

Un algorithme plus efficace que celui de Commentz-Walter est obtenu en combinant l'automate de Aho-Corasick et un DAWG [CCG+99]. Étant donné un mot u, la construction d'un arbre des suffixes T sur u est $O(|u|)$ en temps et en espace; la recherche des occurrences des mots de X dans u s'effectue ensuite en temps $O(|x|)$. Cependant, cette approche est beaucoup moins rapide en pratique [Gus97].

3.3 Représentation matricielle

3.3.1 Introduction

Un automate fini $\mathcal{A} = \langle \Sigma, \mathcal{Q}, \mathcal{Q}_-, \mathcal{Q}^+, \delta \rangle$ est *standard* s'il possède un unique état initial et aucune transition ne pointe vers cet état.

PROPRIÉTÉ 3.3.1 *Si un langage est reconnaissable, alors il existe un automate fini standard qui le reconnaît.*

Il suffit d'ajouter un nouvel état initial qui devient l'unique état initial de l'automate duquel, pour chacune des transitions qui partaient des états initiaux, part une transition de même étiquette.

Soit $X \in \Sigma^*$ un sous-ensemble reconnaissable. Dans la suite, nous notons toujours $\mathcal{A}$ l'automate non nécessairement minimal déterministe standard qui reconnaît X. Nous souhaitons utiliser cet automate pour trouver toutes les occurrences de X dans un mot.

Nous commençons par une présentation intuitive de notre construction. Étant donnés une position i dans le mot u et un état q de l'automate déterministe $\mathcal{A}$, nous considérons l'ensemble des préfixes du sous-ensemble reconnaissable X qui sont des suffixes de $u_0 u_1 \cdots u_i$ et qui envoient dans $\mathcal{A}$ l'état initial sur l'état q. Nous sauvegardons toutes les positions de ces occurrences (la position de leur première lettre) dans l'entrée q d'un tableau Occ indicé par l'ensemble des états de l'automate.

Plus précisément, après la lecture de la lettre u_i, $0 \leq i \leq m-1$, l'entrée indicée par q du tableau Occ contient les valeurs suivantes :

$$\mathrm{Occ}[q] = \{j \mid 0 \leq j \leq i \text{ et } q_- \cdot u_j u_{j+1} \cdots u_i = q\}.$$

Supposons maintenant que nous disposions d'autant de tableaux Occ qu'il y a de lettres dans le mot u. Nous notons Occ_i, $0 \leq i < |u|$, le tableau Occ associé à la

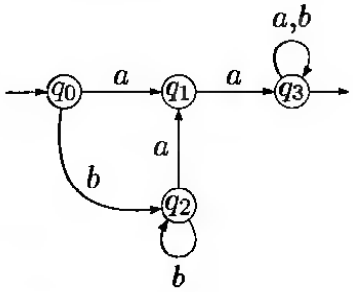
lettre u_i. Nous initialisons le tableau Occ_0 de la façon suivante :

$$\mathrm{Occ}_0[q_-] = \{1\}$$
$$\forall q \in \mathcal{Q} \setminus \{q_-\}, \qquad \mathrm{Occ}_0[q] = \begin{cases} \{0\} & \text{si } q_- \cdot u_0 = q \\ \emptyset & \text{sinon} \end{cases}$$

Le calcul du tableau Occ_{i+1}, $i \geq 0$, à partir du tableau Occ_i est donné par :

$$\mathrm{Occ}_{i+1}[q_-] = \{i+2\}$$
$$\forall q \in \mathcal{Q} \setminus \{q_-\}, \qquad \mathrm{Occ}_{i+1}[q] = \bigcup_{p \cdot u_{i+1} = q} \mathrm{Occ}_i[p]$$

EXEMPLE 2 *Considérons l'automate défini ci-après.*



Pour le mot $u = abaa$, les tableaux Occ_i, $0 \leq i < 4$, sont donnés par :

	q_0	q_1	q_2	q_3
Occ_0	$\{1\}$	$\{0\}$	$\emptyset$	$\emptyset$
Occ_1	$\{2\}$	$\emptyset$	$\{1\}$	$\emptyset$
Occ_2	$\{3\}$	$\{1,2\}$	$\emptyset$	$\emptyset$
Occ_3	$\{4\}$	$\{3\}$	$\emptyset$	$\{1,2\}$

◇

Remarquons que pour tout état $q \in \mathcal{Q}$, l'entrée $\mathrm{Occ}_{i+1}[q]$ contient l'ensemble vide s'il n'existe pas un état $p \in \mathcal{Q}$ tel que $p \cdot u_{i+1} = q$. Puisque le calcul du tableau Occ_{i+1}, $i \geq 0$, n'utilise que le tableau Occ_i, deux tableaux Occ suffisent.

Dans la suite, nous proposons un algorithme qui n'utilise qu'un seul tableau Occ. Autrement dit, nous recherchons une relation de mise à jour de la forme :

$$\mathrm{Occ}[q_-] = \{i+2\}$$
$$\forall q \in \mathcal{Q} \setminus \{q_-\}, \qquad \mathrm{Occ}[q] = \bigcup_{\substack{p \in \mathcal{Q} \\ p \cdot u_{i+1} = q}} \mathrm{Occ}[p]$$

3.3.2 Semi-anneau

DÉFINITION 3.3.2 (SEMI-ANNEAU) *Un* semi-anneau $\mathbb{K}$ *est un ensemble muni de deux opérations : une* addition, *notée* $\oplus$, *et une* multiplication, *notée* $\otimes$, *qui satisfont les axiomes suivants.*

1. $\mathbb{K}$ *est un* monoïde commutatif *pour* $\oplus$ *dont l'élément neutre, appelé le* zéro *de* $\mathbb{K}$, *est noté* $0_{\mathbb{K}}$ *(ou 0),*
2. $\mathbb{K}$ *est un* monoïde *(non nécessairement commutatif) pour* $\otimes$ *dont l'élément neutre, appelé* l'unité de $\mathbb{K}$*), est noté* $1_{\mathbb{K}}$ *(ou 1),*
3. $\otimes$ *est distributive à gauche et à droite par rapport à* $\oplus$ *:*

$$\forall\, i,j,k \in \mathbb{K}, \quad i \otimes (j \oplus k) = (i \otimes j) \oplus (i \otimes k) \quad \textit{et} \quad (i \oplus j) \otimes k = (i \otimes k) \oplus (j \otimes k)$$

4. *L'élément neutre pour* $\oplus$ *est un zéro pour* $\otimes$ *:*

$$\forall\, i \in \mathbb{K}, \quad i \otimes 0 = 0 \otimes i = 0$$

Un semi-anneau S est *idempotent* si $a \oplus a = a$ pour tout $a \in S$. Il est *commutatif* si $\otimes$ est une opération commutative.

EXEMPLE 3 1. *Le semi-anneau de Boole,* $\mathbb{B} = \{0,1\}$, *est entièrement défini par les axiomes et par l'équation* $1 \oplus 1 = 1$.

2. *L'ensemble* $\mathbb{N}$ *des entiers positifs ou nul, muni des opérations d'addition et de multiplication ordinaires, est un semi-anneau.*

Soit $\mathcal{A} = (\mathcal{Q},\Sigma,\delta,q_-,\mathcal{Q}_+)$ un automate fini déterministe. Dans la suite, $\mathbb{K}$ sera toujours le semi-anneau sur les parties de $\mathbb{N}$ où l'addition $\oplus$ est l'union ensembliste et la multiplication $\otimes$ est l'*addition des parties* :

$$X \oplus Y = \{x + y \mid x \in X \text{ et } y \in Y\}$$

Par convention, $X \otimes \emptyset = \emptyset$ pour tout $X \subseteq \mathcal{P}(\mathbb{N})$.

Un automate déterministe étant sous-entendu, nous considérons le morphisme μ de Σ dans l'ensemble des matrices d'ordre n à coefficients dans le semi-anneau $\mathbb{K}$ défini par :

$$\mu(a)_{p,q} = \begin{cases} \{1\} & \text{si } p = q = q_- \\ \{0\} & \text{si } p \cdot a = q,\ q \neq q_- \\ \emptyset & \text{sinon.} \end{cases}$$

Le morphisme μ est étendu en un morphisme de Σ^* dans l'ensemble des matrices d'ordre n à coefficients dans le semi-anneau $\mathbb{K}$. Par abus de notations, nous continuons à noter μ ce morphisme :

$$\forall u,v \in \Sigma^*, \quad \mu(uv) = \mu(u)\mu(v)$$

EXEMPLE 4 *Soit $\mathcal{A}$ l'automate de l'exemple 2. Alors, les matrices successives pour le mot $u = abaa$ sont (par convention, les lignes et les colonnes sont indicées dans l'ordre q_0, q_1, q_2 et q_3) :*

$$\mu(a) = \begin{bmatrix} \{1\} & \{0\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{0\} \\ \emptyset & \{0\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{0\} \end{bmatrix} \qquad \mu(ab) = \begin{bmatrix} \{2\} & \emptyset & \{1\} & \emptyset \\ \emptyset & \emptyset & \emptyset & \{0\} \\ \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{0\} \end{bmatrix}$$

$$\mu(aba) = \begin{bmatrix} \{3\} & \{1,2\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{0\} \\ \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{0\} \end{bmatrix} \qquad \mu(abaa) = \begin{bmatrix} \{4\} & \{3\} & \emptyset & \{1,2\} \\ \emptyset & \emptyset & \emptyset & \{0\} \\ \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{0\} \end{bmatrix}.$$

Nous interprétons la première ligne de la dernière matrice en remarquant que les suffixes de abaa sont : 1 en position 4 (q_0 sur q_0), a en position 3 (Q_0 sur Q_1) et aa et baa en position 2 et 1 respectivement (q_0 sur q_4). ◇

LEMME 3.3.3 $\forall u \in \Sigma^+, \quad \mu(u)_{q_-,q_-} = \{|u|\}$

PROPOSITION 3.3.4 *Pour tout $q \in \mathcal{Q} \setminus \{q_-\}$ et $u \in \Sigma^+$, $\mu(u)_{q_-,p}$ est l'ensemble des positions de départ des suffixes de u qui sont l'étiquette d'un chemin de q_- à p dans $\mathcal{A}$.*

PREUVE

Par induction sur la longueur de u. La proposition est vraie pour $|u| = 1$. Supposons que la proposition soit vraie pour tous les mots de longueur au plus m. Soient $u \in \Sigma^m$, et $a \in \Sigma$. Alors,

$$\begin{aligned} \mu(ua)_{q_-,p} &= \bigoplus_{q \in \mathcal{Q}} \mu(u)_{q_-,q} \otimes \mu(a)_{q,p} \\ &= \bigcup_{q \in \mathcal{Q}} \mu(u)_{q_-,q} \otimes \mu(a)_{q,p} \end{aligned}$$

Or,

$$\forall q \in \mathcal{Q}, \quad \mu(u)_{q_-,q} \otimes \mu(a)_{q,p} = \begin{cases} \mu(u)_{q_-,q} \otimes \{0\} = \mu(u)_{q_-,q} & \text{si } q \cdot a = p \\ \mu(u)_{q_-,q} \otimes \emptyset = \emptyset & \text{sinon} \end{cases}$$

Alors,

$$\mu(ua)_{q_-,p} = \bigcup_{\substack{q \in \mathcal{Q} \\ q \cdot a = p}} \mu(u)_{q_-,q}.$$

Par conséquent, $\mu(u)_{q_-,p}$ est l'ensemble des positions de départ des suffixes de u qui sont l'étiquette d'un chemin de q_- à p dans $\mathcal{A}$. $\boxed{3.3.4}$

EXEMPLE 5 *Nous reprenons l'automate de l'exemple 2. Puisque l'état q_3 est terminal et $\mu(abaa)_{q_0,q_3} = \{1,2\}$, les facteurs $u_1u_2u_3 = baa$ et $u_2u_3 = aa$ appartiennent au langage reconnu par l'automate $\mathcal{A}$.*

3.4 Résolution «*en place*» de système d'équations

3.4.1 Système d'affectations

Nous présentons ici la notion de «*système d'affectations*». Intuitivement, un système d'affectations est un ensemble de calculs sur un ensemble de variables. Par exemple, si $A = [a_{i,j}]$ est une matrice d'ordre n et $x = [x_i]$, $y = [y_i]$ sont deux vecteurs colonnes d'ordre n, l'équation matricielle suivante : ***système d'affectations***

$$Ax = y$$

est un système d'affectations. En effet, en notant $X = \{x_i \mid 1 \leq i \leq n\}$ et $Y = \{y_i \mid 1 \leq i \leq n\}$, la variable y_i, $1 \leq i \leq n$, reçoit le calcul $f_i(X_i)$, où f_i est une fonction linéaire et X_i est entièrement déterminé par les coefficients $a_{i,j}$, $1 \leq j \leq n$.

DÉFINITION 3.4.1 (SYSTÈME D'AFFECTATIONS) *Un* système d'affectations S *est un ensemble :*

$$S = \{y_i \leftarrow f_i(X_i) \mid 1 \leq i \leq n\}$$

où $X = \{x_1, x_2, \ldots, x_n\}$ *et* $Y = \{y_1, y_2, \ldots, y_n\}$ *sont deux ensembles de variables,* $X_i \subseteq X$ *pour tout* $1 \leq i \leq n$, *et* f_i, $1 \leq i \leq n$, *est une fonction quelconque. Le système d'affectations* S *est* disjoint *si les ensembles de variables* X *et* Y *sont disjoints,* i.e. $X \cap Y = \emptyset$.

DÉFINITION 3.4.2 (SYSTÈME D'AFFECTATIONS ORDONNÉ) *Un système d'affectations* S *est* ordonné *si l'ordre des affectations est imposé.*

Si un système d'affectations disjoint n'est pas ordonné, nous disons qu'il est *simultané* ou *parallèle*, *i.e.* tout se passe comme si toutes les affectations étaient effectuées en même temps. Précisons maintenant les notations. Soit $S = \{y_i \leftarrow f_i(X_i) \mid 1 \leq i \leq n\}$ un système d'affectations. Dans la suite, nous notons

$$\begin{cases} y_1 \leftarrow f_1(X_1) \\ y_2 \leftarrow f_2(X_2) \\ \quad\vdots \\ y_n \leftarrow f_n(X_n) \end{cases}$$

si aucun ordre n'est imposé, et

$$\begin{array}{c} 1.\ y_1 \leftarrow f_1(X_1) \\ 2.\ y_2 \leftarrow f_2(X_2) \\ \vdots \\ n.\ y_n \leftarrow f_n(X_n) \end{array}$$

si l'ordre des affectations est imposé, *i.e.* S est un système d'affectations ordonné. De plus, afin de différencier *variable* et *valeur d'une variable*, nous notons $\langle v \rangle$ la valeur de la variable $v \in X \cup Y$.

Nous pouvons associer une représentation matricielle (booléenne) à tout système d'affectations disjoint.

DÉFINITION 3.4.3 (MATRICE DE DÉPENDANCE) *Soit* $S = \{y_i \leftarrow f_i(X_i) \mid 1 \leq i \leq n\}$ *un système d'affectations disjoint. La* (0,1)-matrice de dépendance *de* S, *notée* $A_S = [a_{i,j}]$ *(ou plus simplement* A *s'il n'y a pas d'ambiguïté), est définie par* $a_{i,j} = 1$ *si et seulement si* $x_j \in X_i$.

Autrement dit, $a_{i,j} = 1$, $1 \leq i \leq n$ et $1 \leq j \leq n$, si et seulement si la variable x_j est un paramètre de la fonction f_i. Chaque ligne de la matrice A correspond à une affectation et chaque colonne à une variable. Aussi, la matrice A est la matrice d'adjacence réduite d'un graphe biparti $G = (Y \cup X, E)$ appelé *graphe biparti de dépendance*. Une arête $\{y_i, x_j\} \in E$ indique que x_j est une variable nécessaire au calcul de la fonction f_i, *i.e.* x_j est un paramètre de la fonction f_i. En d'autres termes, le degré d'un sommet $y_i \in Y$ est le nombre de paramètres de la fonction f_i et le degré d'un sommet x_j est le nombre d'équations où la variable x_j apparaît en paramètre.

3.4.2 Calcul en place d'un système d'affectations disjoint

Nous commençons par une présentation intuitive. Soit S le système d'affectations disjoint non-ordonné défini par :

$$\begin{cases} y_1 \leftarrow f_1(x_2, x_3) \\ y_2 \leftarrow f_2(x_1, x_2, x_3) \\ y_3 \leftarrow f_3(x_2) \end{cases}$$

Considérons maintenant le système d'affectation ordonné S' défini par :

$$\begin{aligned} &1)\ x_1 \leftarrow f_2(x_1, x_2, x_3) \\ &2)\ x_3 \leftarrow f_1(x_2, x_3) \\ &3)\ x_2 \leftarrow f_3(x_2) \end{aligned}$$

Remarquons alors que pour toute valeur initiale des variables x_1, x_2 et x_3, après exécution du système d'affectations ordonné S', nous avons $\langle y_1 \rangle = \langle x_3 \rangle$, $\langle y_2 \rangle = \langle x_1 \rangle$ et $\langle y_3 \rangle = \langle x_2 \rangle$.

Aussi, au prix d'une *indirection* supplémentaire, nous pouvons calculer le système d'affectation S sans utiliser aucune variable de l'ensemble Y. Il suffit ensuite d'utiliser les ordres (supposés connus) des variables et des affectations pour retrouver les résultats du calcul.

Évidemment, ce n'est pas toujours possible. Par exemple, le système d'affectations S'' défini par :

$$\begin{cases} y_1 \leftarrow f_1(x_3) \\ y_2 \leftarrow f_2(x_1, x_2, x_3) \\ y_3 \leftarrow f_3(x_1, x_2, x_3) \end{cases}$$

n'est pas calculable en place. En effet, chaque variable x_i, $1 \leq i \leq n$, est utilisée au moins deux fois en argument dans les fonctions du système d'affectations.

Nous définissons dans un premier temps plus formellement la notion de *calcul en place* d'un système d'affectations disjoint. Ensuite, nous donnons une condition nécessaire et suffisante pour qu'un système d'affectations disjoint soit calculable en place.

DÉFINITION 3.4.4 (SYSTÈME D'AFFECTATIONS DISJOINT CALCULABLE EN PLACE) *Un système d'affectations disjoint* $S = \{y_i \leftarrow f_i(X_i) \mid 1 \leq i \leq n\}$ *est* calculable en place *s'il existe deux permutations* π *et* φ *de* $[\![n]\!]$ *telles qu'une exécution du système d'affectations ordonné disjoint défini par :*

$$\begin{aligned} &1)\ x_{\pi(1)} \leftarrow f_{\varphi(1)}(X_{\varphi(1)}) \\ &2)\ x_{\pi(2)} \leftarrow f_{\varphi(2)}(X_{\varphi(2)}) \\ &\qquad\vdots \\ &n)\ x_{\pi(n)} \leftarrow f_{\varphi(n)}(X_{\varphi(n)}) \end{aligned}$$

vérifie $\langle y_{\varphi(i)} \rangle = \langle x_{\pi(i)} \rangle$ *pour* $1 \leq i \leq n$.

En d'autres termes, si un système d'affectations disjoint est calculable en place, alors après chaque affectation de la forme :

$$i)\ x_{\pi(i)} \leftarrow f_{\varphi(i)}(X_{\varphi(i)})$$

pour $1 \leq i \leq n$, la variable $x_{\pi(i)}$ n'est plus utilisée dans les $n-i$ affectations suivantes.

Rappelons que la (0,1)-matrice ∇_n est la matrice triangulaire supérieure pleine d'ordre n, *i.e.* $\nabla_{i,j} = 1$ si et seulement si $j \geq i$ pour $1 \leq i,j \leq n$. De plus, étant données deux (0,1)-matrices $A = [a_{i,j}]$ et $B = [b_{i,j}]$ de taille m par n, alors nous notons $A \leq B$ si et seulement si $a_{i,j} \leq b_{i,j}$ pour $1 \leq i,j \leq n$.

PROPOSITION 3.4.5 *Soient* $S = \{y_i \leftarrow f_i(X_i) \mid 1 \leq i \leq n\}$ *un système d'affectations disjoint et* A *la* (0,1)*-matrice de dépendance de* S*. Alors,* S *est calculable en place si et seulement si il existe des matrices de permutation* P *et* Q *d'ordre* n *telles que* $PAQ^T \leq \nabla_n$.

PREUVE

Supposons que le système d'affectations S soit calculable en place. Alors, il existe les permutations π et φ de $[\![n]\!]$ telles que dans le nouveau système d'affectations S' défini par :

$$S' = \{x_{\pi(i)} \leftarrow f_{\varphi(i)}(X_{\varphi(i)}) \mid 1 \leq i \leq n\}$$

la fonction $f_{\varphi(j)}$ ne contient aucune occurrence de la variable $x_{\varphi(i)}$ en paramètre pour tout $1 \leq i < j \leq n$. Soit P (resp. Q) la matrice de permutation d'ordre n associée à φ (resp. π). La condition précédente est équivalente à $PAQ^T \leq \nabla_n$.

Inversement, supposons qu'il existe les matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \nabla_n$. Soient φ (resp. π) la permutation de $[\![n]\!]$ associée à P (resp. Q). Alors, le système d'affectations S' défini par :

$$S' = \{x_{\pi(i)} \leftarrow f_{\varphi(i)}(X_{\varphi(i)}) \mid 1 \leq i \leq n\}$$

est calculable en place. En effet, la fonction $f_{\varphi(j)}$ ne contient aucune occurrence de la variable $x_{\varphi(i)}$ en paramètre pour tout $1 \leq i < j \leq n$ $\boxed{3.4.5}$

La proposition précédente se traduit immédiatement sur le graphe biparti de dépendance du système d'affectations disjoint S.

COROLLAIRE 3.4.6 *Soient $S = \{y_i \leftarrow f_i(X_i) \mid 1 \leq i \leq n\}$ un système d'affectations disjoint et $G = (Y \cup X, E)$ le graphe biparti de dépendance de S. Alors, S est calculable en place si et seulement si il existe des permutations π et φ de $[\![n]\!]$ telles que si $\{y_i, x_j\} \in E$ alors $\pi(i) \leq \varphi(j)$.*

EXEMPLE 6 *Soit S le système d'affectations défini par :*

$$\begin{cases} y_1 \leftarrow f_1(x_2) \\ y_2 \leftarrow f_2(x_3) \\ y_3 \leftarrow f_3(x_1, x_2, x_3, x_4) \\ y_4 \leftarrow f_4(x_2, x_3) \end{cases}$$

La matrice de dépendance A est donnée par :

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Le système est calculable en place pour les permutations π et φ de $[\![n]\!]$ définies par :

$$\begin{array}{llll} \pi(1) = 1 & \pi(2) = 4 & \pi(3) = 3 & \pi(4) = 2 \\ \varphi(1) = 3 & \varphi(2) = 4 & \varphi(3) = 2 & \varphi(4) = 1 \end{array}$$

Notons P la matrice de permutation ligne associée à φ, et Q^T la matrice de permutation colonne associée à π :

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad Q^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

En effet, nous avons :

$$PA = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \textit{puis} \qquad PAQ^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Nous vérifions $PAQ^T \leq \nabla_4$. Le système se réécrit en :

$$\begin{array}{l} 1)\ x_{\pi(1)=1} \leftarrow f_{\varphi(1)}(X_{\varphi(1)}) = f_3(x_1, x_2, x_3, x_4) \\ 2)\ x_{\pi(2)=4} \leftarrow f_{\varphi(2)}(X_{\varphi(2)}) = f_4(x_2, x_3) \\ 3)\ x_{\pi(3)=3} \leftarrow f_{\varphi(3)}(X_{\varphi(3)}) = f_2(x_3) \\ 4)\ x_{\pi(4)=2} \leftarrow f_{\varphi(4)}(X_{\varphi(4)}) = f_1(x_2) \end{array}$$

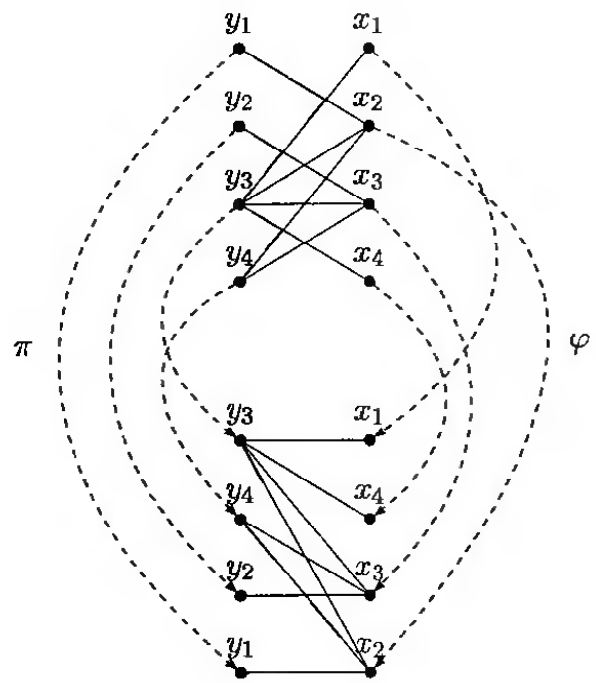


FIG. 3.1 – *En haut, le graphe biparti de dépendance du système d'affectations de l'exemple 6. En bas, le même graphe permuté par les permutations π et φ de $[\![n]\!]$. Remarquons que dans ce dernier, les pentes de toutes les arêtes sont négatives ou nulles par suite de l'inégalité matricielle $PAQ^T \leq \nabla_n$.*

Les graphes bipartis de dépendance et de dépendance permuté sont donnés en figure 3.1. La figure 3.2 détaille les 4 affectations en place du système permuté.

◇

3.5 Recherche des occurrences d'un sous-ensemble reconnaissable dans un mot

3.5.1 Introduction

Nous avons vu (proposition 3.4.5) qu'un système d'affectations est calculable en place si et seulement si il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \nabla_n$, où A est la (0,1)-matrice de dépendance du système d'affectations. Nous pouvons appliquer ce résultat à la recherche de toutes les occurrences d'un sous-ensemble reconnaissable $X \subseteq \Sigma^*$ dans un mot $u \in \Sigma^*$.

Nous allons montrer que, étant donnée une lettre $a \in \Sigma$, les transitions définies par a peuvent être appliquées en place par un système d'affectations ordonnées de

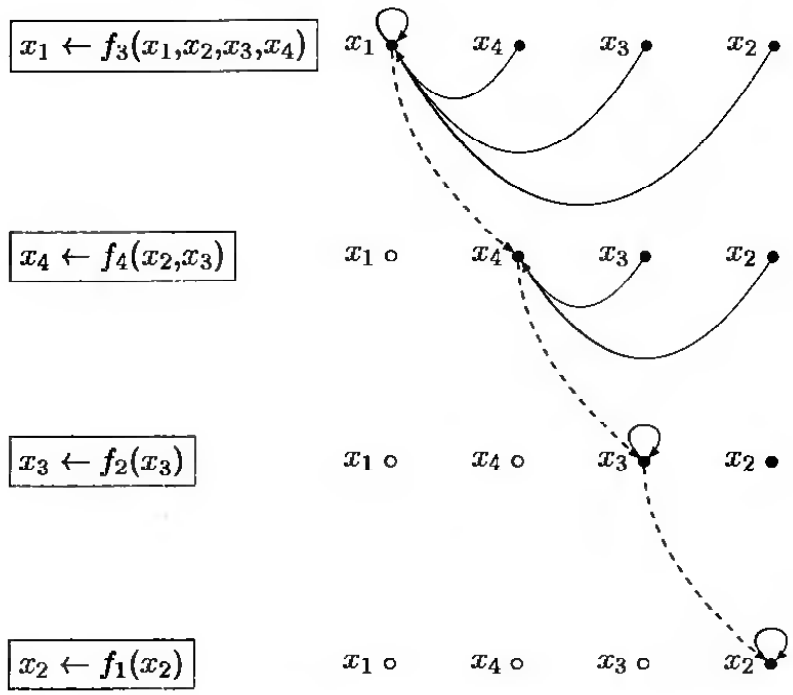


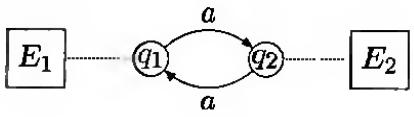
FIG. 3.2 – *Détails du calcul en place du système d'affectations de l'exemple 6. Le calcul se décompose en quatre pas de calcul. Un arc continu d'un sommet x_j vers un sommet x_i signifie que la variable x_j est utilisée pour le calcul stocké dans la variable x_i. Remarquons qu'une fois un calcul stocké dans une variable, cette dernière n'est plus utilisée par la suite (sommets de couleur claire). Par exemple, la variable x_4 calculée au second pas n'est plus utilisée dans les deux pas suivants. Les arcs pointillés représente l'ordonnancement des variables.*

la forme :

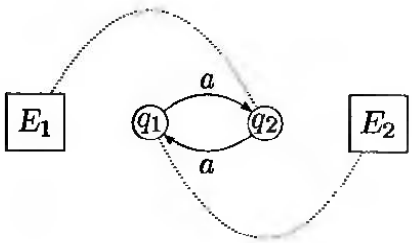
$$\forall i,\ 1 \leq i \leq |\mathcal{Q}|, \qquad x_{\pi(i)} = f_{\varphi(i)}(X_{\varphi(i)}).$$

Les variables x_i, $1 \leq i \leq |\mathcal{Q}|$, représenteront des ensembles de positions dans un mot et les fonctions f_i, $1 \leq i \leq |\mathcal{Q}|$, décrivent l'action de la lettre a sur l'automate.

Nous commençons par une présentation intuitive de l'algorithme. Soient $u \in \Sigma^*$ un mot et i une position dans u. Notons $a = u_{i+1}$ et considérons l'action de a sur les deux états ci-dessous. Les éléments E_1 et E_2 représentent des ensembles de positions dans u. L'ensemble E_i, $1 \leq i \leq 2$, est l'ensemble des positions de départ des suffixes du $u_0u_1 \cdots u_i$ qui sont l'étiquette d'un chemin de l'état initial à q_i.



Nous considérons plus particulièrement l'état q_1. Lorsque la lettre $a = u_{i+1}$ est lue, l'ensemble des positions de départ des suffixes du $u_0u_1 \cdots u_{i+1}$ qui sont l'étiquette d'un chemin de l'état initial à q_1 est exactement l'ensemble des positions de départ des suffixes du $u_0u_1 \cdots u_i$ qui sont l'étiquette d'un chemin de l'état initial à q_2. La situation est symétrique pour l'état q_2. Il suffit donc de recopier l'ensemble E_1 dans E_2 et inversement. Ces recopies peuvent être évitées en associant directement l'ensemble E_1 à l'état q_2 et l'ensemble E_2 à l'état q_1 :



Autrement dit, nous associons à chaque état non plus un ensemble (de positions), mais l'indice d'un ensemble. En d'autres termes, nous introduisons un niveau d'indirection entre les états et les ensembles. Les recopies se limitent dans ce cas à un simple échange d'indices. Tout n'est pas si simple s'il existe plusieurs transitions vers un même état par une même lettre comme nous le verrons dans la suite.

Nous introduisons dans un premier temps les définitions nécessaires. Nous associons en particulier à chaque lettre $a \in \Sigma$ une application $\overline{a}$. Nous montrons ensuite que cette application est toujours une bijection de l'ensemble des états dans lui même. Enfin, nous expliquons comment utiliser l'approche décrite ci-dessus dans le cas de la recherche des occurrences d'un sous-ensemble reconnaissable dans un mot.

3.5.2 Définitions

Nous notons $\mathrm{Dom}(a)$ et $\mathrm{Im}(a)$, $a \in \Sigma$, le *domaine* et l'*image* de l'action de a sur l'automate $\mathcal{A}$. Plus précisément,

$$\forall a \in \Sigma, \quad \begin{cases} \mathrm{Dom}(a) = \{q \mid q \in \mathcal{Q} \text{ et } \exists p \subset \mathcal{Q}, \quad q \cdot a = p\} \\ \mathrm{Im}(a) = \{q \mid q \in \mathcal{Q} \text{ et } \exists p \in \mathcal{Q}, \quad p \cdot a = q\} \end{cases}$$

Nous supposons donné un ordre total (arbitraire), noté $\leq$, sur l'ensemble des états de l'automate. Dans la suite, cet ordre sera parfois implicitement donné par les états eux-mêmes : $q_i \leq q_j$ si et seulement si $i \leq j$. Définissons l'ensemble $\min(a)$ pour $a \in \Sigma$ des états «*minimaux*» pour l'ordre $\leq$:

$$\min(a) = \{q \mid q \in \mathcal{Q} \text{ et } \forall p \in \mathcal{Q}, \quad q \cdot a = p \cdot a \Rightarrow q \leq p\}.$$

Autrement dit,

$$\forall a \in \Sigma, \quad \min(a) = \bigcup_{p \in \mathrm{Im}(a)} \min\, p \cdot a^{-1}.$$

où $p \cdot a^{-1}$ est l'ensemble des états q tels que $q \cdot a = p$. Il est immédiat par la définition précédente qu'un état minimal appartient toujours au domaine de l'action a, *i.e.* $\min(a) \subseteq \mathrm{Dom}(a)$ pour $a \in \Sigma$.

LEMME 3.5.1 *Soient $a \in \Sigma$ et $q \in \mathrm{Dom}(a) \setminus \mathrm{Im}(a)$. Alors,il existe un entier positif k tel que $q \cdot a^k \notin \min(a)$.*

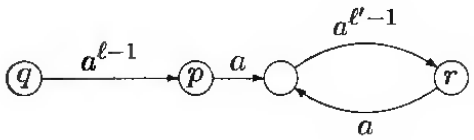
PREUVE

Puisque $q \in \mathrm{Dom}(a) \setminus \mathrm{Im}(a)$, il existe un état $p \in \mathrm{Im}(a)$ tel que $q \cdot a = p$. Supposons par l'absurde que pour tout entier $k \geq 1$ et $q \cdot a^k$ définie, nous ayons $q \cdot a^k \in \min(a)$. Envisageons les deux cas suivants :

1. Il existe un plus petit entier $\ell > 1$ pour lequel $q \cdot a^\ell$ n'est pas définie. Notons $p = q \cdot a^{\ell-1}$. Alors, nous avons $p \notin \mathrm{Dom}(a)$ et par suite $p \notin \min(a)$ car $\min(a) \subseteq \mathrm{Dom}(a)$ pour $a \in \Sigma$, ce qui est une contradiction.
2. Pour tout $k \geq 1$, $q \cdot a^k$ est définie. Remarquons que $q \neq q \cdot a^k$ pour $k \geq 1$ car $q \in \mathrm{Dom}(a) \setminus \mathrm{Im}(a)$. Notons $\mathcal{Q}'$ le sous-ensemble d'états défini par :

   $$\mathcal{Q}' = \{p \mid \exists k,k' \geq 1, \quad p = q \cdot a^k = q \cdot a^{k+k'}\}.$$

   Par hypothèse, tous les états de $\mathcal{Q}'$ sont minimaux et $q \notin \mathcal{Q}'$. Soient ℓ le plus petit entier positif pour lequel nous ayons $q \cdot a^\ell \in \mathcal{Q}'$ et ℓ' un entier positif tel que $q \cdot a^\ell = q \cdot a^{\ell+\ell'}$. Notons $p = q \cdot a^{\ell-1}$ (nous avons $p = q$ si $\ell = 1$) et $r = q \cdot a^{\ell+\ell'-1}$. Alors, $p \neq r$ car $p \notin \mathcal{Q}'$ et $r \in \mathcal{Q}'$. De plus, $p \cdot a = r \cdot a$. Mais, au plus un des deux états p et r appartient à $\min(a)$, ce qui est une contradiction.



3.5.1

À chaque lettre $a \in \Sigma$, nous associons maintenant une application $\overline{a}$ de l'ensemble des états dans lui-même définie par :

$$\forall a \in \Sigma, \forall q \in \mathcal{Q}, \quad q \cdot \overline{a} = \begin{cases} q & \text{si } q \notin \mathrm{Dom}(a) \cup \mathrm{Im}(a), \\ p & \text{si } q \in \mathrm{Im}(a) \text{ et } p = \min\{p' \mid p' \cdot a = q\}, \\ q \cdot a^k & \text{sinon, où } k = \min\{\ell \mid q \cdot a^\ell \notin \min(a)\}. \end{cases}$$

L'application $\overline{a}$ pour $a \in \Sigma$ est bien définie. En effet, il suffit de vérifier en utilisant le lemme 3.5.1 que pour tout $q \in \mathrm{Dom}(a) \setminus \mathrm{Im}(a)$, il existe un entier positif k tel que $q \cdot a^k \notin \min(a)$.

PROPRIÉTÉ 3.5.2 $\forall a \in \Sigma, \forall q \in \mathrm{Im}(a), \quad q \cdot \overline{a} \in \min(a)$

Nous étendons l'action de $\overline{a}$ pour $a \in \Sigma$ à l'ensemble Σ^* par *anti-homomorphisme* en posant :

$$\forall q \in \mathcal{Q}, \forall u,v \in \Sigma^*, \quad q \cdot \overline{uv} = q \cdot \overline{v}\,\overline{u}.$$

3.5.3 Propriétés

Un *automate à groupe* est un automate réversible, déterministe et complet. Autrement dit, l'action de chaque lettre est une bijection de l'ensemble des états dans lui-même. Dans le cas des automates à groupe, pour tout $a \in \Sigma$ et $q \in \mathcal{Q}$, nous avons $q \in \mathrm{Im}(a)$. En reprenant la définition de l'application $\overline{a}$ pour $a \in \Sigma$, il vient :

$$\forall a \in \Sigma, \forall q \in \mathcal{Q}, \quad q \cdot \overline{a} = \min\{p \mid p \cdot a = q\}.$$

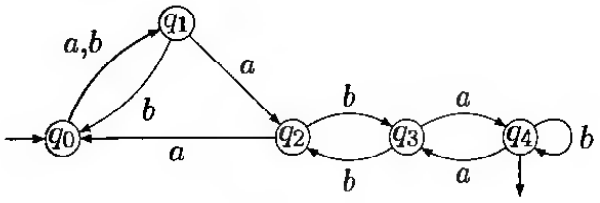
Mais puisque l'action de a pour $a \in \Sigma$ est une bijection de l'ensemble des états dans lui-même, il existe un unique état p tel que $p \cdot a = q$. La proposition suivante est alors immédiate.

PROPOSITION 3.5.3 *Soit $\mathcal{A}$ un automate à groupe. Alors,*

$$\forall a \in \Sigma, \forall p,q \in \mathcal{Q}, \quad p \cdot a = q \quad \textit{si et seulement si} \quad q \cdot \overline{a} = p.$$

Autrement dit, si $\mathcal{A}$ est un automate à groupe, l'application $\overline{a}$ pour $a \in \Sigma$ est la bijection inverse de l'action de a.

EXEMPLE 7 *Soit $\mathcal{A}$ l'automate à groupe défini par :*



Les applications $\overline{a}$ et $\overline{b}$ sont obtenues en inversant le sens de toutes les flèches.

Nous venons de voir que, dans le cas des automates à groupe, $\overline{a}$ pour $a \in \Sigma$ est une bijection de l'ensemble des états dans lui-même (en fait, $\overline{a}$ est la bijection inverse de l'action de a). Nous allons montrer que pour tout automate déterministe $\mathcal{A}$, l'application $\overline{u}$ pour $u \in \Sigma^*$ définit *toujours* une bijection de l'ensemble des états dans lui-même.

PROPOSITION 3.5.4 *Pour tout $u \in \Sigma^*$, l'application $\overline{u}$ est une bijection de l'ensemble des états dans lui-même.*

PREUVE

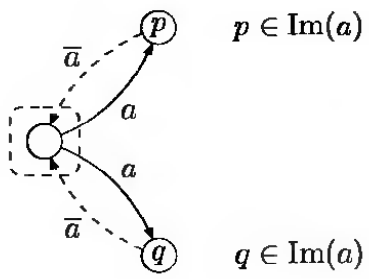
Il suffit de montrer que la proposition est vraie pour toutes les lettres de l'alphabet Σ. Puisqu'une application d'un ensemble fini dans lui même est surjective si et seulement si elle est injective, il suffit de montrer que l'application $\overline{a}$ pour $a \in \Sigma$ est injective.

Soit $a \in \Sigma$. Remarquons dans un premier temps que:

1. $\forall q \in \mathrm{Dom}(a) \cup \mathrm{Im}(a), \quad q \cdot \overline{a} \in \mathrm{Dom}(a) \cup \mathrm{Im}(a)$;
2. $\forall q \in \mathcal{Q} \setminus (\mathrm{Dom}(a) \cup \mathrm{Im}(a)), \quad q \cdot \overline{a} = q$.

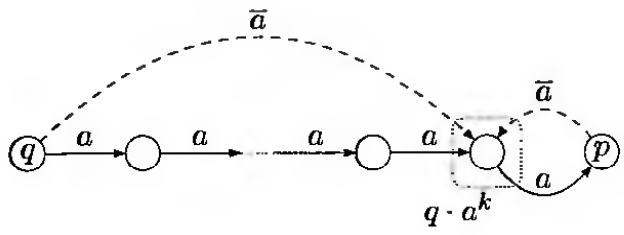
Soient $p,q \in \mathrm{Dom}(a) \cup \mathrm{Im}(a)$, $p \neq q$, et supposons $p \cdot \overline{a} = q \cdot \overline{a}$.

1. Si $p,q \in \mathrm{Im}(a)$, alors $\min\{p' \mid p' \cdot a = p\} = \min\{q' \mid q' \cdot a = q\}$. C'est une contradiction puisqu'il existe au plus une transition depuis un même état pour une lettre donnée.

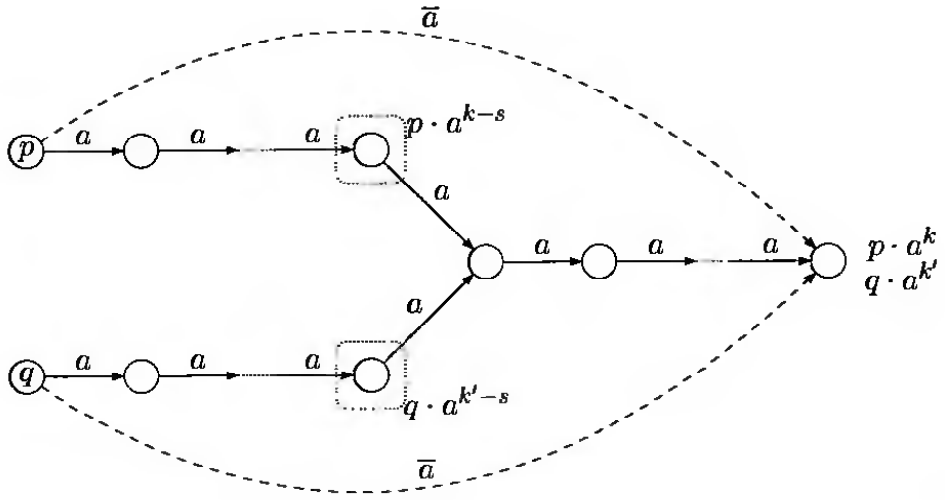


2. Si $p \in \mathrm{Im}(a)$ et $q \in \mathrm{Dom}(a) \setminus \mathrm{Im}(a)$, alors $p \cdot \overline{a} = \min\{p' \mid p' \cdot a = p\}$ et $q \cdot \overline{a} = q \cdot a^k$ où k est le plus petit entier ℓ tel que $q \cdot a^\ell \notin \min(a)$. C'est une

contradiction puisque nous avons $p \cdot \overline{a} \in \min(a)$ et $q \cdot \overline{a} \notin \min(a)$..



3. Si $p,q \in \mathrm{Dom}(a) \setminus \mathrm{Im}(a)$, alors $p \cdot \overline{a} = p \cdot a^k$ où k est le plus petit entier ℓ tel que $p \cdot a^\ell \notin \min(a)$ et $q \cdot \overline{a} = q \cdot a^{k'}$ où k' est le plus petit entier ℓ' tel que $q \cdot a^{\ell'} \notin \min(a)$. Soit s le plus petit entier positif tel que $p \cdot a^{k-s} \neq q \cdot a^{k'-s}$. Alors, au plus un des deux états $p \cdot a^{k-s}$ et $q \cdot a^{k'-s}$ appartient à $\min(a)$, ce qui est une contradiction.

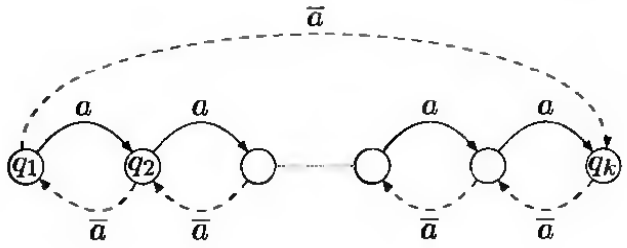


3.5.4

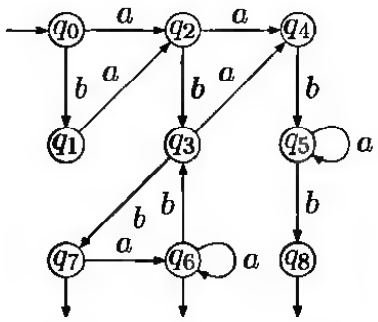
COROLLAIRE 3.5.5 *Soient $a \in \Sigma$ et $\mathcal{Q}' \subseteq \mathcal{Q}$ tels que la restriction de l'action de a à $\mathcal{Q}'$ soit une bijection. Alors, la restriction de $\overline{a}$ à $\mathcal{Q}'$ est la bijection inverse de la restriction de l'action de a à $\mathcal{Q}'$ si et seulement si $\mathcal{Q}' \subseteq \min(a)$.*

Nous avons vu que l'application $\overline{a}$ pour $a \in \Sigma$ est une bijection de l'ensemble des états dans lui même, *i.e.* une permutation. Aussi, $\overline{a}$ pour $a \in \Sigma$ se décompose en un produit de cycles. En fait, les calculs de la forme $q \cdot a^k$ où $k = \min\{\ell \mid q \cdot a^\ell \notin \min(a)\}$ sont justement les processus de «*fermeture des cycles*». En effet, soit $\{q_i \mid 1 \leq i \leq k\} \subseteq \mathcal{Q}$ un sous-ensemble d'états tel que $q_i \cdot a = q_{i+1}$ pour $1 \leq i \leq k-1$ et supposons $\{q_i \mid 1 \leq i \leq k\} \subseteq \min(a)$. Alors, $q_{i+1} \cdot \overline{a} = q_i$ pour $1 \leq i \leq k-1$. Le chemin étiqueté

par $\overline{a}$ est «*fermé*» par $q.\overline{a} = q_1 \cdot a^{k-1} = q_k$ comme illustré sur l'automate ci-dessous.



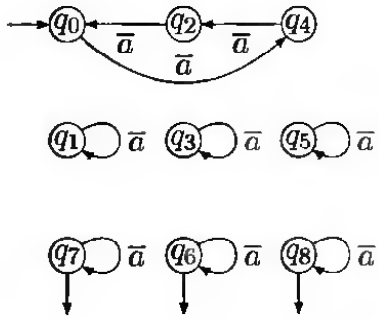
EXEMPLE 8 *Soit $\mathcal{A}$ l'automate défini par :*



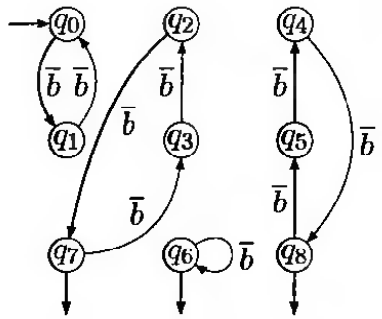
En supposant $q_i \leq q_j$ si et seulement si $i \leq j$, les ensembles $\min(a)$ *et* $\min(b)$ *sont définis par :*

$$\min(a) = \{q_0, q_2, q_5, q_6\}$$
$$\min(b) = \{q_0, q_2, q_3, q_4, q_5\}$$

La bijection $\overline{a}$ de l'ensemble des états dans lui-même est définie par :



et la bijection $\overline{b}$ de l'ensemble des états dans lui-même est définie par :



◇

3.5.4 Calcul en place

Rappelons que notre but est, pour toute lettre $a \in \Sigma$, d'appliquer les transitions définies par a par un système d'affectations ordonnées de la forme :

$$\forall i,\ 1 \leq i \leq |\mathcal{Q}|, \qquad x_{\pi(i)} = f_{\varphi(i)}(X_{\varphi(i)})$$

où les variables x_i, $1 \leq i \leq |\mathcal{Q}|$, représentent des ensembles de positions dans un mot et les fonctions f_i, $1 \leq i \leq |\mathcal{Q}|$, décrivent l'action de la lettre a sur l'automate.

Introduisons tout d'abord quelques notations et définitions. Étant donnée une bijection $\overline{a}$ de l'ensemble des états dans lui-même, nous notons $P_{\overline{a}} = [p_{i,j}]$ la matrice de permutation d'ordre n définie par :

$$\forall q,r \in \mathcal{Q}, \qquad p_{q,r} = 1 \quad \text{si et seulement si} \quad r \cdot \overline{a} = q.$$

Pour tout $a \in \Sigma$, nous notons $M_a = [m_{p,q}]$, $p,q \in \mathcal{Q}$, la (0,1)-matrice d'ordre $|\mathcal{Q}|$ définie par $m_{p,q} = 1$ si et seulement si $q \cdot a = p$. Cette représentation décrit complètement les transitions étiquetées par a, *i.e.* chaque élément non nul $m_{p,q}$ de M_a code une transition de l'état q vers p étiquetée par a. Aussi, puisque $\mathcal{A}$ est déterministe, chaque colonne de la matrice M_a contient au plus un élément non nul. En d'autres termes,

$$\forall a \in \Sigma,\ \forall p \in \mathrm{Im}(a),\ \exists q \in \mathcal{Q}, \qquad m_{p,q} = 1 \tag{3.1}$$
$$\forall a \in \Sigma,\ \forall p \notin \mathrm{Im}(a),\ \forall q \in \mathcal{Q}, \qquad m_{p,q} = 0 \tag{3.2}$$

et

$$\forall a \in \Sigma,\ \forall q \in \mathrm{Dom}(a),\ \exists! p \in \mathcal{Q}, \quad m_{p,q} = 1 \tag{3.3}$$
$$\forall a \in \Sigma,\ \forall q \notin \mathrm{Dom}(a),\ \forall p \in \mathcal{Q}, \quad m_{p,q} = 0 \tag{3.4}$$

LEMME 3.5.6 *Soit $A = [a_{i,j}]$ une (0,1)-matrice d'ordre n, $n > 1$, comportant m éléments non nuls avec $1 \leq m \leq n+1$. Alors, il existe une ligne indicée par i, $1 \leq i \leq n$, qui contient au moins un élément non nul et la matrice de taille $n-1$ par n obtenue en supprimant la ligne i a une colonne qui ne contient que des 0.*

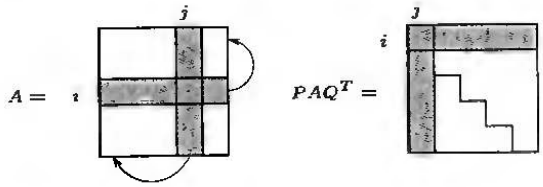
PREUVE

Soit i, $1 \leq i \leq n$, l'indice d'une ligne de A qui contient le plus grand nombre d'éléments non nuls. Cette ligne contient au moins un élément non nul car $m \geq 1$. Soit A' la matrice de taille $n-1$ par n obtenue en supprimant la ligne i. Puisque la ligne indicée par i contient au moins un élément non nul, la matrice A' contient au plus $m-1 = n$ éléments non nuls. Supposons que toutes les colonnes de la matrice A' contiennent au moins un élément non nul. Alors, la matrice A' contient exactement n éléments non nuls (un par colonne). Alors, il existe une ligne de A' qui contient au moins deux éléments non nuls car A' a $n-1$ lignes. Par suite, la ligne indicée par i contenait un unique élément non nul. C'est une contradiction puisque i est l'indice d'une ligne qui contient le plus grand nombre d'éléments non nuls dans A. $\boxed{3.5.6}$

PROPOSITION 3.5.7 *Soit A une (0,1)-matrice d'ordre n pour $n > 1$ comportant au plus $n+1$ éléments non nuls. Alors, il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \nabla_n$.*

PREUVE

Si $A = \mathbf{0}_{n,n}$, alors nous avons $A \leq \nabla_n$. Sinon, en appliquant le lemme 3.5.6, il existe une ligne indicée par i qui contient au moins un élément non nul et la matrice obtenue en supprimant la ligne indicée par i a une colonne qui ne contient que des 0. Notons j l'indice de la colonne de A' qui ne contient que des 0. Soient P et Q les matrices de permutation d'ordre n qui placent l'élément $a_{i,j}$ dans le coin supérieur gauche.



La sous-matrice obtenue en supprimant la première ligne et la première colonne est d'ordre $n-1$ et contient au plus n éléments non nuls. Nous concluons par induction puisque la proposition est vérifiée pour $n = 2$. $\boxed{3.5.7}$

La matrice M_a pour $a \in \Sigma$ contient au plus $|Q|$ éléments non nuls. En appliquant la proposition 3.5.7, nous savons qu'il existe des matrices de permutation P et Q d'ordre $|Q|$ telles que $PM_aQ^T \leq \nabla_{|Q|}$ pour $a \in \Sigma$. Autrement dit, il existe les bijections σ et σ' de l'ensemble des états dans lui-même telles que $P_\sigma M_a Q^T_{\sigma'} \leq \nabla_{|Q|}$. En fait, nous allons montrer que la bijection $\overline{a}$ pour $a \in \Sigma$ permet toujours de s'affranchir d'une permutation des colonnes.

PROPOSITION 3.5.8 $\forall a \in \Sigma, \quad P_{\overline{a}} M_a \leq \nabla_{|Q|}$

PREUVE

Notons $B = [b_{p,q}]$, $p,q \in \mathcal{Q}$, la matrice $P_{\overline{a}}M_a$. En utilisant (3.1) et (3.2), il suffit de montrer que :

$$\forall p \in \mathrm{Im}(a),\ \forall p \in \mathcal{Q},\quad p \cdot \overline{a} > q \ \Rightarrow\ b_{p\cdot\overline{a},q} = 0.$$

Supposons par l'absurde que $P_{\overline{a}}M_a \not\leq \diagdown_{|\mathcal{Q}|}$. Alors, il existe les états $p \in \mathrm{Im}(a)$ et $q \in \mathrm{Dom}(a)$ tels que $q \cdot a = p$ et $p \cdot \overline{a} > q$. Notons $q' = p \cdot \overline{a}$. Par suite, il vient $q' \cdot a = q \cdot a = p$ et $q' > q$. C'est une contradiction puisque $q' = \min\{p' \mid p' \cdot a = p\}$. $\boxed{3.5.8}$

EXEMPLE 9 *Reprenons l'exemple de l'automate de l'exemple 2 pour la lettre a. Alors,*

$$q_0 \cdot \overline{a} = q_3 \qquad q_1 \cdot \overline{a} = q_0 \qquad q_2 \cdot \overline{a} = q_2 \qquad q_3 \cdot \overline{a} = q_1$$

Nous vérifions

$$P_aM_a = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \leq \diagdown_4.$$

◇

3.5.5 Application à la recherche des occurrences

Nous appliquons les résultats précédents à la recherche de toutes les occurrences d'un sous-ensemble reconnaissable $X \subseteq \Sigma^*$ dans un mot $u \in \Sigma^*$. Nous supposons donné un automate déterministe $\mathcal{A}$ qui reconnaît X. De plus, nous supposons qu'il n'existe aucune transition dans $\mathcal{A}$ qui pointe vers l'état initial.

Soient $a \in \Sigma$ et $M_a = [m_{p,q}]$ la (0,1)-matrice associée. Rappelons que $m_{p,q} = 1$ si et seulement si $q \cdot a = p$. Nous avons montré précédemment que pour toute lettre $a \in \Sigma$, la bijection $\overline{a}$ vérifie $P_{\overline{a}}M_a \leq \diagdown_{|\mathcal{Q}|}$.

Rappelons que pour tout $p,q \in \mathcal{Q}$, $\mu(u)_{p,q}$ est l'ensemble des positions de départ des suffixes de u qui sont l'étiquette d'un chemin de p à q. Dans l'algorithme `occ-ss-rec`, nous notons i la position courante dans un mot u. De plus, nous notons σ la permutation $\overline{u_{0,i}}$. Nous supposons que pour $q \in \mathcal{Q}$, la valeur de $\mu(u)_{q_-,q}$ est stockée dans le registre $\mathtt{Reg}_{q\cdot\sigma}$. L'algorithme parcourt le mot u séquentiellement. Pour chaque lettre u_i, il effectue les opérations suivantes :

1. mise à jour (réécriture) du registre associé à l'état initial;
2. calcul des registres associés aux autres états de l'automate;
3. mise à jour de la permutation courante;
4. calcul des occurrences, *i.e.* les positions de départ des suffixes de u_i qui sont dans le sous-ensemble reconnaissable X.

Initialement, tous les registres sont vides ($\mathtt{Reg}_q = \emptyset$ pour $q \in \mathcal{Q}$). De plus, le registre $\mathtt{Reg}_q$ est associé à l'état q pour $q \in \mathcal{Q}$ car $q \cdot \sigma = q$.

Algorithm 4: Calcul de toutes les occurrences d'un sous-ensemble reconnaissable dans un mot.

```
occ-ss-rec
Entrée: Un automate déterministe A, et un mot u ∈ Σ* de longueur m.
Sortie: L'ensemble de tous les facteurs de u qui sont reconnus par A.
début
    /* initialisation des registres et de la permutation courante */
1   pour q ∈ Q faire
2       Reg_q ← ∅
3       q · σ = q
    /* boucle principale */
4   pour i de 0 à m − 1 faire
        /* nous notons a = u_i la lettre courante */
        /* traitement de l'état initial */
5       Reg_{q_⊥·σ} ← {i + 1}

        /* Parcours des états de l'automate A */
6       pour q ∈ Q suivant l'ordre ≤ faire
7           si q ∈ min(a) alors
8               pour p ∈ Q avec q ≠ p tel que q · a = p · a faire
9                   Reg_{q·σ} ← Reg_{q·σ} ∪ Reg_{p·σ}
10          sinon
11              Reg_{q·σ} ← ∅
        /* mise à jour de la permutation courante */
12      σ ← ā σ
        /* calcul des occurrences */
13      pour q ∈ Q⁺ faire
14          pour j ∈ Reg_{q·σ} faire
15              afficher une occurrence du facteur u_j u_{j+1} ··· u_i
fin
```

REMARQUE 3.5.9 Soient $a \in \Sigma$ et $p,q \in \mathcal{Q}$. Considérons une itération quelconque de l'algorithme. Si $q \cdot a = p \cdot a$ et $q \in \min(a)$, alors l'état q est considéré avant l'état p dans la boucle de l'algorithme. Il suffit pour cela de remarquer que les états de l'automate sont considérés suivant l'ordre arbitraire $\leq$. Autrement dit, nous sommes assurés que le registre associé à l'état p n'a pas été initialisé (ligne 11) avant que celui-ci n'ait été utilisé (ligne 9) pour calculer la valeur du registre associé à l'état q.

PROPOSITION 3.5.10 *L'algorithme* `occ-ss-rec` *calcule tous les facteurs d'un mot u qui sont l'étiquette d'un chemin de l'état initial à un état terminal.*

PREUVE

Il suffit de vérifier l'invariant suivant par induction sur i :

$$\forall q \in \mathcal{Q}, \forall i, 0 \leq i \leq m-1, \qquad \mu(u_0 u_1 \ldots u_i)_{q_-,q} = \mathsf{Reg}_{q \cdot \overline{u_i} \cdots \overline{u_1}\,\overline{u_0}}.$$

Rappelons dans un premier temps que $\overline{u_i} \ldots \overline{u_1}\,\overline{u_0} = \overline{u_0 u_1 \ldots u_i}$.

Soit $q \in \mathcal{Q}$ un état quelconque. Supposons tout d'abord $q \notin \mathrm{Im}(u_{i+1})$. Alors, nous avons $\mu(u_0 u_1 \cdots u_{i+1})_{q_-,q} = \emptyset$. Considérons maintenant les deux cas suivants :

1. Si $q \notin \mathrm{Dom}(u_{i+1})$, alors $q \cdot \overline{u_{i+1}} = q$. De plus, $\mathsf{Reg}_{q \cdot \overline{u_0 u_1 \ldots u_i}} = \emptyset$ (ligne 11 de l'algorithme). D'où $\mathsf{Reg}_{q \cdot \overline{u_0 u_1 \ldots u_{i+1}}} = \emptyset$.
2. Si $q \in \mathrm{Dom}(u_{i+1})$, alors $q \cdot \overline{u_{i+1}} = q \cdot u_{i+1}^k$ où $k = \min\{\ell \mid q \cdot u_{i+1}^\ell \notin \min(u_{i+1})\}$. Notons $r = q \cdot \overline{u_{i+1}}$. Mais, $\mathsf{Reg}_{r \cdot \overline{u_0 u_1 \ldots u_i}} = \emptyset$ car $r \notin \min(u_{i+1})$. Par suite, nous avons $\mathsf{Reg}_{q \cdot \overline{u_0 u_1 u_{i+1}}} = \emptyset$.

Nous supposons dorénavant $q \in \mathrm{Im}(u_{i+1})$. Alors, nous avons :

$$\mu(u_0 u_1 \cdots u_{i+1})_{q_-,q} = \bigcup_{p \cdot u_{i+1} = q} \mu(u_0 u_1 \cdots u_i)_{q_-,p}$$

Notons $r = q \cdot \overline{u_{i+1}}$. Après les lignes 6-11 de l'algorithme, nous avons :

$$\mathsf{Reg}_{r \cdot \overline{u_0 u_1 \ldots u_{i+1}}} = \bigcup_{p \cdot u_{i+1} = q} \mathsf{Reg}_{q \cdot \overline{u_0 u_1 \ldots u_i}}$$

Or, $r = q \cdot \overline{u_{i+1}}$. D'où,

$$\mathsf{Reg}_{q \cdot \overline{u_0 u_1 \ldots u_{i+1}}} = \bigcup_{p \cdot u_{i+1} = q} \mathsf{Reg}_{q \cdot \overline{u_0 u_1 \ldots u_i}}$$

Par hypothèse, la valeur de l'élément

$$\mu(u_0 u_1 \cdots u_i)_{q_-,p}, \qquad \text{pour } p \in \mathcal{Q} \text{ et } p \cdot u_{i+1} = q$$

est stockée dans le registre

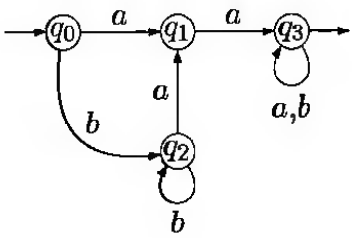
$$\mathsf{Reg}_{p \cdot \overline{u_0 u_1 \ldots u_i}} \qquad \text{pour } p \in \mathcal{Q} \text{ et } p \cdot u_{i+1} = q.$$

Par conséquent, nous avons

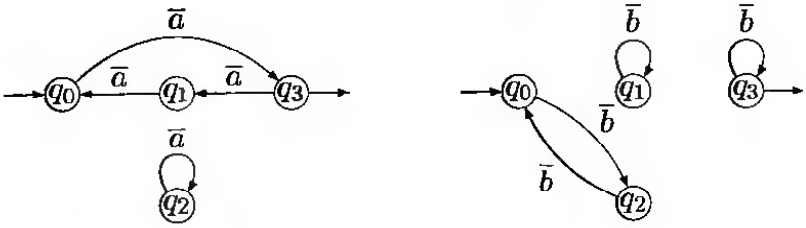
$$\mu(u_0 u_1 \ldots u_{i+1})_{q_-,q} = \mathsf{Reg}_{q \cdot \overline{u_0 u_1 \ldots u_{i+1}}}$$

qui est le résultat recherché. 3.5.10

EXEMPLE 10 *Soit $\mathcal{A}$ l'automate défini par :*



Les bijections $\overline{a}$ et $\overline{b}$ de l'ensemble des états dans lui-même sont données ci-après (par convention, $q_i \leq q_j$ si $i \leq j$) :

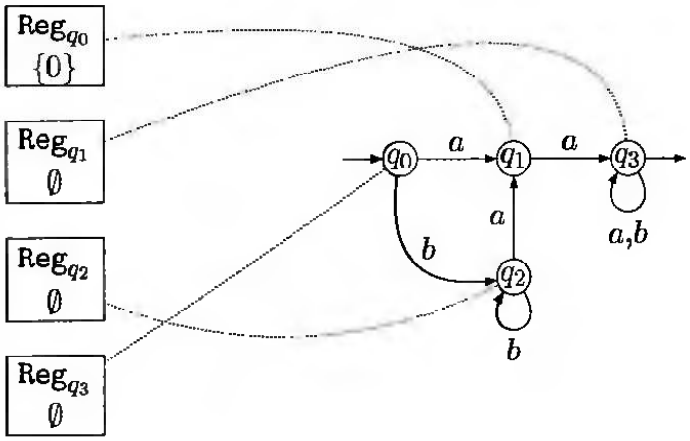


Nous détaillons l'algorithme sur l'automate $\mathcal{A}$ pour le mot $u = abba$. Pour chaque lettre, nous indiquons :

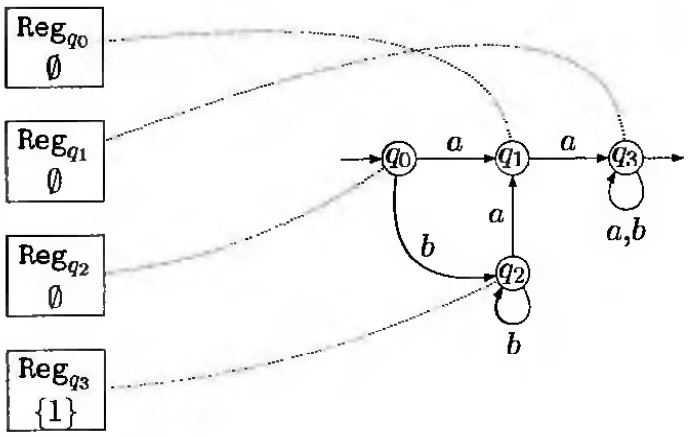
1. *Les valeurs de tous les registres.*
2. *La permutation courante σ (sous forme de cycles).*

Les arêtes en pointillés indiquent l'état associé. Par exemple, après la lecture de la première lettre a, le registre Reg_{q_0} est associé à l'état q_1, le registre Reg_{q_1} est associé à l'état q_3, le registre Reg_{q_2} est associé à l'état q_2 et le registre Reg_{q_3} est associé à l'état q_0,

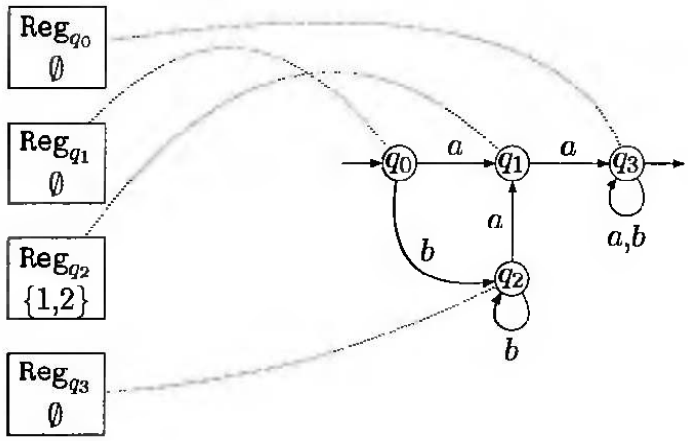
$\sigma = \overline{a} = (0\,3\,1)\,(2)$



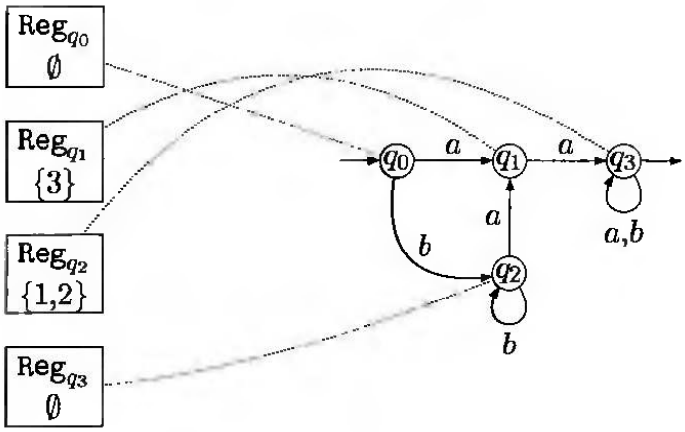
$\sigma = \overline{ab} = (0\,2\,3\,1)$



$\sigma = \overline{aba} = (0\,1\,2\,3)$



$\sigma = \overline{abaa} = (0)\,(1)\,(2\,3)$



Le registre $\mathtt{Reg}_{q_2}$ *est associé à l'état* q_3. *Puisque* $\mathtt{Reg}_{q_2} = \{1,2\} \neq \emptyset$ *et* q_3 *est terminal, les facteurs* $u_1u_2u_3 = baa$ *et* $u_2u_3 = aa$ *sont affichés.* ◇

3.6 Mise en œuvre

3.6.1 Calcul de $\overline{a}$ pour $a \in \Sigma$

Les bijections $\overline{a}$ pour $a \in \Sigma$ ne dépendent que de la lettre a. Nous pouvons donc les calculer une fois pour toutes avant l'étape de recherche.

Le précalcul de la bijection $\overline{a}$ pour $a \in \Sigma$ est donné ci-après par l'algorithme `calcul-bijection`. L'algorithme se décompose en deux étapes. Dans une première étape, nous parcourons tous les états de l'automate suivant l'ordre arbitraire $\leq$. Si $q \cdot \overline{a}$ n'est pas définie lorsque l'état q est visité, celui-ci est ajouté à l'ensemble P. Dans une seconde boucle, nous calculons (si nécessaire) $q \cdot \overline{a}$ pour tous les états de P.

Algorithm 5: Calcul de la bijection $\overline{a}$ pour $a \in \Sigma$.

```
calcul-bijection
Entrée: Un automate déterministe A muni d'un ordre total ≤ sur les états,
        et une lettre a ∈ Σ
Sortie: La bijection ā de l'ensemble des états Q dans lui-même
début
    /* initialisation */
1   P ← ∅
    /* boucle L1 */
2   pour q ∈ Q suivant l'ordre ≤ faire
3       si il existe p ∈ Q tel que q · a = p alors
4           si p · ā n'est pas définie alors
5               p · ā = q
6               q · a_min = p
7       si q · ā n'est pas définie alors
8           P ← P ∪ {q}
    /* boucle L2 */
9   tant que P ≠ ∅ faire
10      retirer un état q de P
11      si q · ā n'est pas définie alors
12          si q · a_min n'est pas définie alors
13              q · ā = q
14          sinon
15              p ← q
16              tant que p · a_min est définie faire
17                  p ← p · a_min
18              q · ā ← p
19  retourner ā
fin
```

Nous utilisons dans l'algorithme `calcul-bijection` une injection partielle, notée $a_{\min}$, définie pour tous les états minimaux. En effet, pour tout $q \in \mathcal{Q}$, elle vérifie $q \cdot a_{\min} = p \;\Rightarrow\; p \cdot \overline{a} = q$. Elle est construite dans la boucle L_1 et utilisée dans la boucle L_2.

LEMME 3.6.1 *Après la boucle L_1, pour tout $q \in \mathcal{Q}$, les conditions suivantes sont équivalentes :*

1. *il existe $p \in \min(a)$ tel que $p \cdot a = q$ et $p \leq q$;*
2. *$q \notin P$.*

PREUVE

($1 \Rightarrow 2$) S'il existe $p \in \min(a)$ tel que $p \cdot a = q$, alors $p \cdot a\overline{a} = p$. Puisque $p \leq q$ et les états de l'automate sont examinés dans la boucle L_1 suivant l'ordre $\leq$, alors $q \cdot \overline{a}$ est déjà définie (lignes 7-8) lors de l'examen de l'état q. Par suite, $q \notin P$.

($2 \Rightarrow 1$) Si $q \notin P$, alors $q \cdot \overline{a}$ était déjà définie lors de l'examen de l'état q. Aussi, il existe $p \in \min(a)$ avec $p \leq q$ tel que $p \cdot a\overline{a} = p$ et $p \cdot a = q$. $\boxed{3.6.1}$

À la fin de la boucle L_1, $q \cdot \overline{a}$ est définie pour tous les états q pour lesquels il existe $p \in \mathcal{Q}$ tel que $p \cdot a = q$ et ceci même si $q \leq p$. Autrement dit, $q \cdot \overline{a}$ est définie à la fin de la boucle L_1 si et seulement si $q \in \mathrm{Im}(a)$. Remarquons qu'à la fin de la boucle L_1, nous pouvons avoir à la fois $q \cdot \overline{a}$ définie et $q \in P$. C'est le cas pour tous les états q tels que $q < \min\{p \mid p \cdot a = q\}$.

PROPOSITION 3.6.2 *Pour tout $q \in \mathcal{Q}$, l'algorithme `calcul-bijection` calcule*

$$q \cdot \overline{a} = \begin{cases} q & \textit{si } q \notin \mathrm{Dom}(a) \cup \mathrm{Im}(a) \\ p & \textit{si } q \in \mathrm{Im}(a) \textit{ et } p = \min\{p' \mid p' \cdot a = q\} \\ q \cdot a^k & \textit{sinon, où } k = \min\{\ell \mid q \cdot a^\ell \notin \min(a)\} \end{cases}$$

PREUVE

Examinons les trois cas suivants :

1. Si $q \notin \mathrm{Dom}(a) \cup \mathrm{Im}(a)$, alors $q \cdot \overline{a}$ n'est pas définie à la fin de la boucle L_1 et par suite $q \in P$. Après la boucle L_2, nous avons $q \cdot \overline{a} = q$ car q n'est pas un état minimal.
2. Si $q \in \mathrm{Im}(a)$, alors il existe un état $p \in \min(a)$ tel que $p \cdot a\overline{a} = p$. Par suite, $q \cdot \overline{a} = p$ dès la fin de la boucle L_1. Remarquons cependant que nous pouvons avoir $q \in P$. Le test ligne 11 de l'algorithme écarte ces états.
3. Si $q \in \mathrm{Dom}(a) \setminus \mathrm{Im}(a)$, alors $q \cdot \overline{a}$ n'est pas définie à la fin de la boucle L_1, et par suite $q \in P$. Aussi, $q \cdot \overline{a}$ est calculée dans la boucle L_2. Si $q \cdot a_{\min}$ n'est pas définie, alors $q \notin \min(a)$. En conséquence, $q \cdot \overline{a} = q = q \cdot a^0$. Inversement, si $q \cdot a_{\min}$ est définie, alors $q \in \min(a)$. Nous calculons lignes 15-18 de l'algorithme le plus entier k tel que $q \cdot a^k \notin \min(a)$ en utilisant l'injection $a_{\min}$.

$\boxed{3.6.2}$

3.6.2 Complexité

Étant donnés un sous-ensemble reconnaissable $X \subseteq \Sigma^*$ et un mot $u \in \Sigma^*$, nous notons :

1. n le nombre d'états de l'automate «*minimal*» (sans transitions entrantes pour l'état initial) qui reconnaît X,
2. $m = |u|$.

Nous étudions tout d'abord la complexité du calcul des bijections $\overline{a}$ pour $a \in \Sigma$. Soient p et q deux états de l'automate. Rappelons que si $q \cdot a_{\min} = p$, alors $p \cdot \overline{a} = q$. Autrement dit, $q \cdot a_{\min}$ est définie à la fin de la boucle L_1 si et seulement si $q \in \min(a)$.

LEMME 3.6.3 *Soit $q \in Q$. Si $q \cdot a_{\min}$ est définie à la fin de la boucle L_1, alors q est visité (lignes 16-17) au plus une fois dans la boucle L_2.*

PREUVE

Supposons que q soit visité (lignes 16-17) au moins deux fois dans la boucle L_2. Alors il existe les états $p,p' \in P$ tels que $p \cdot a_{\min} = p' \cdot a_{\min} = q$. Une contradiction puisqu'au plus un deux deux états q et q' est minimal. $\boxed{3.6.3}$

PROPOSITION 3.6.4 *L'algorithme* `calcul-bijection` *s'exécute en temps $O(n)$.*

PREUVE

La boucle L_1 parcourt tous les états de l'automate. Nous effectuons à chaque itération un nombre constant d'opérations. Par conséquent, la boucle L_1 est en temps $O(n)$.

L'ensemble P contient au plus n états. Pour chaque état q de P, la boucle L_2 n'est effective que si $q \cdot \overline{a}$ n'est pas définie. Si $q \cdot a_{\min}$ n'est pas définie, nous effectuons une unique opération en temps constant. Si $q \cdot a_{\min}$ est définie, nous utilisons l'injection partielle $a_{\min}$ pour calculer $q \cdot \overline{a}$. Mais ce calcul n'utilise que des états minimaux. Or, nous savons (lemme 3.6.3) que chaque état minimal est visité au plus une fois dans la boucle L_2. Par conséquent, la boucle L_2 est également en temps $O(n)$. $\boxed{3.6.4}$

Par conséquent, puisque l'algorithme `calcul-bijection` est appelé pour chaque lettre de l'alphabet, la complexité totale du calcul des bijections $\overline{a}$ pour $a \in \Sigma$ est en temps $O(n\,|\Sigma|)$.

Nous nous intéressons maintenant à la complexité de l'algorithme de recherche `occ-ss-rec`. L'algorithme de recherche nécessite un espace de travail $O(m)$ pour les registres. C'est par exemple le cas si l'automate $\mathcal{A}$ est dénoté par l'expression rationnelle a^*b et $u = a^{m-1}b$ puisque tous les suffixes propres de u sont reconnus.

Nous pouvons représenter les registres par des listes chaînées [CLR92]. Remarquons maintenant (ligne 9) que si nous avons une affectation de la forme :

$$\mathsf{Reg}_{q\cdot\sigma} \leftarrow \mathsf{Reg}_{q\cdot\sigma} \cup \mathsf{Reg}_{p\cdot\sigma}$$

alors q est un état minimal, p n'est pas un état minimal et $q \cdot a = p \cdot a$. Aussi, le registre associé à l'état p sera initialisé (ligne 11) lorsque ce dernier sera visité :

$$\mathsf{Reg}_{p\cdot\sigma} \leftarrow \emptyset$$

En conséquence, nous pouvons directement concaténer les listes chaînées $\mathtt{Reg}_{q\cdot\sigma}$ et $\mathtt{Reg}_{p\cdot\sigma}$. Cette opération s'effectue en temps constant par une simple manipulation de pointeurs.

L'initialisation (lignes 1-3) est en temps $O(n)$. Détaillons la complexité de la boucle principale.

1. Le traitement de l'état initial est en temps constant puisque c'est une simple réécriture.
2. Le parcours des états de l'automate ainsi que le calcul des registres est en temps $O(n)$.
3. La mise à jour de la permutation courante σ est en temps $O(n)$.

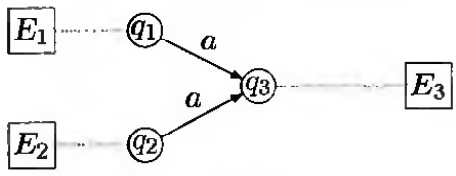
Nous regroupons ces remarques dans la proposition suivante (k est le nombre d'occurrences des mots du sous-ensemble reconnaissable X dans le mot u).

PROPOSITION 3.6.5 *L'algorithme* `occ-ss-rec` *s'exécute en temps* $O(mn + k)$.

3.6.3 Conclusion

Nous avons présenté dans ce chapitre un algorithme pour la recherche de toutes les occurrences d'un sous-ensemble reconnaissable X dans un mot. Nous avons toujours supposé que X était donné par un automate déterministe standard, *i.e.* un automate déterministe pour lequel aucune transition ne pointe sur l'unique état initial. Notre algorithme utilise un niveau d'indirection supplémentaire entre les états et les ensembles de positions.

Remarquons qu'il est toujours possible d'associer à chaque état q un ensemble de positions E_q et d'effectuer des copies ou des effacements d'ensembles pour simuler les transitions de l'automate. Par exemple, considérons l'automate suivant (nous représentons par des carrés les ensembles de positions associés aux états de l'automate).

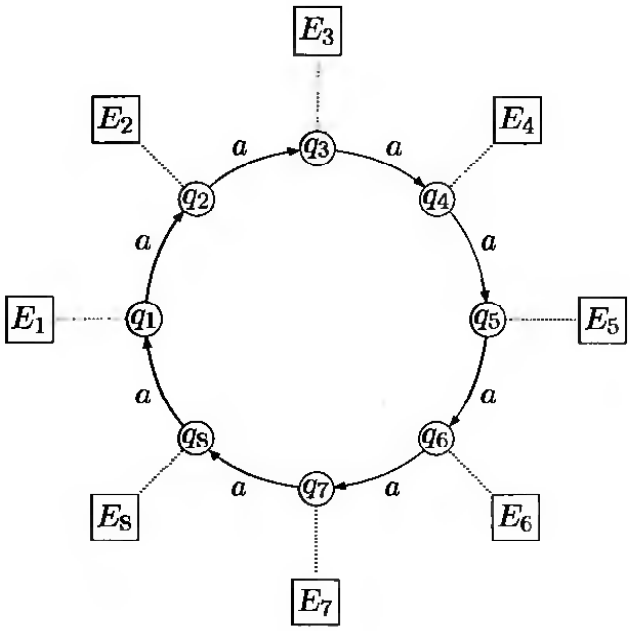


Pour appliquer les transitions étiquetées par la lettre a dans cet automate, nous devons successivement :

1. effacer le contenu de l'ensemble E_3;
2. copier le contenu de l'ensemble E_1 dans l'ensemble E_3;
3. copier le contenu de l'ensemble E_2 dans l'ensemble E_3;
4. effacer le contenu de l'ensemble E_1;
5. effacer le contenu de l'ensemble E_2.

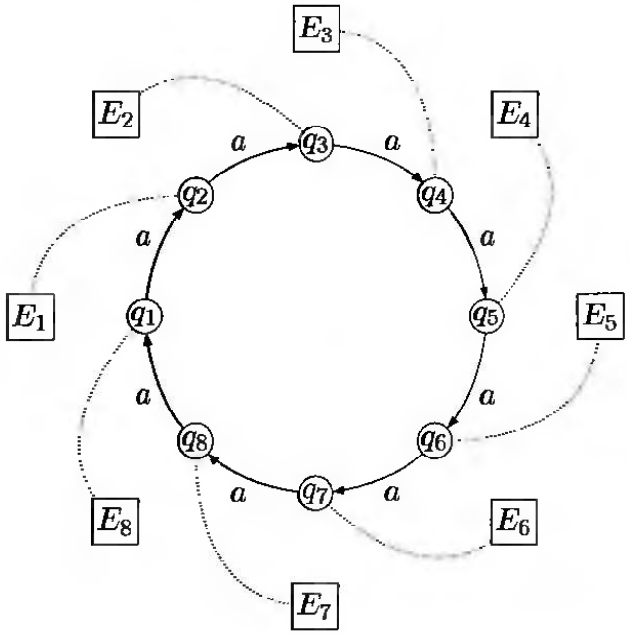
Les opérations se compliquent très rapidement dès qu'une nouvelle transition est ajoutée. La situation est encore pire lorsque l'automate contient de «*grands*» cycles

étiquetés par une même lettre. C'est avant tout dans ce cas particulier que notre algorithme est préférable et surtout beaucoup plus simple à mettre en œuvre. Prenons l'exemple de l'automate suivant.



Une mise à jour (copies + effacement) des ensembles de positions associés aux états nécessite pour chaque transition étiquetée par la lettre a de sauver au moins un des deux ensembles. Autrement dit, nous devons nécessairement utiliser une variable supplémentaire pour stocker un ensemble de positions - ou éventuellement stocker l'indice d'un ensemble de positions. Au contraire, en utilisant l'algorithme `occ-ss-rec` que nous avons proposé, l'application de *toutes* les transitions étiquetées par la lettre a se limite à un simple décalage circulaire des ensembles associés aux états. Cette opération s'effectue très rapidement en opérant uniquement sur le niveau

d'indirection entre les états de l'automate et les ensembles de positions.



Lorsque le nombre d'états de l'automate considéré est petit, nous pouvons utiliser notre algorithme et construire «*à la volée*» un automate dont les états sont indicés par des permutations distinctes de l'ensemble des états de l'automate initial, et il existe une transition d'un état p indicé par une permutation σ_p vers un état q indicé par une permutation σ_q qui est étiquetée par une lettre $a \in \Sigma$ si l'application de la permutation $\overline{a}$ à σ_p correspond à σ_q, *i.e.* $\sigma_q = \overline{a}\ \sigma_p$. Ce nouvel automate - contruit au fur et à mesure de la lecture du mot d'entrée - nous permet donc d'éviter le calcul de la permutation courante à chaque itération lorsque celui-ci a déjà été effectué une fois. Remarquons qu'en pratique, cette construction utilisée tel quel n'est cependant plus utilisable - sans un système de cache[1] - dès que le nombre d'états de l'automate dépasse 6 ou 7 pour un alphabet de taille 4.

1. Nous avons utilisé pour nos tests un algorithme de cache simple qui, lorsque cela est nécessaire, ne conserve à l'intérieur du cache que les états les plus récemment visités pour permettre l'introduction de nouveaux états dans celui-ci.

Chapitre 4

Triangularisation d'une (0,1)-matrice

Sommaire

4.1	Introduction	57
4.2	Famille de sous-ensembles	60
4.3	Permanent non nul	65
4.4	Arrangement linéaire triangulaire	70
4.5	Sous-matrice et couplage	86
4.6	Sous-matrice principale	99
4.7	Arrangement linéaire à cumul croissant	108
4.8	Conclusion et problèmes ouverts	113

Ce chapitre et le suivant sont consacrés à l'étude du problème de la triangularisation d'une (0,1)*-matrice par permutation des lignes et des colonnes. Étant donnée une* (0,1)*-matrice A d'ordre n, il s'agit de décider si nous pouvons réordonner ses lignes et ses colonnes (indépendamment) de sorte que tous les éléments non nuls apparaissent dans la partie inférieure triangulaire. Nous avons déjà vu dans le chapitre précédent qu'il existe pour ce problème un algorithme en temps polynomial si la* (0,1)*-matrice A contient au plus $n+1$ éléments non nuls. Nous consacrons l'intégralité de ce chapitre à l'étude du cas général.*

4.1 Introduction

Nous avons étudié dans le chapitre précédent le problème de la mise sous forme triangulaire d'une (0,1)-matrice par permutation des ligne et des colonnes. Nous avons prouvé que cela est toujours possible si la matrice contient au plus $n+1$ éléments non nuls où n est l'ordre de la matrice (cf. proposition 3.5.7). Nous consacrons ce chapitre et une large partie du suivant à l'étude de ce problème dans le cas général. En d'autres termes, nous étudions le problème de décision MATRICE INF-TRIANGULAIRE défini ci-après.

PROBLÈME 4.1.1 (MATRICE INF-TRIANGULAIRE)
DONNÉES : Une (0,1)-matrice A d'ordre n.
QUESTION : Existe-t-il des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \llcorner_n$?

En d'autres termes, étant donnée une (0,1)-matrice A d'ordre n, pouvons-nous permuter ses lignes et ses colonnes afin d'obtenir une matrice inférieurement triangulaire ? Nous nous proposons d'en étudier la complexité dans le cas général. L'approche naïve consiste à considérer les $n!$ matrices de permutation P différentes, ainsi que les $n!$ matrices de permutation Q différentes, et de vérifier à chaque fois si la matrice PAQ^T obtenue est inférieurement triangulaire, *i.e.* $PAQ^T \leq \llcorner_n$. Ceci nous donne un algorithme en temps $O(n!\,n!)$ (nous verrons cependant dans la suite qu'il suffit en fait de considérer uniquement $n!$ permutations des lignes). Nous devons préciser notre question : pouvons-nous calculer (si elles existent) des matrices de permutation P et Q en un «*temps raisonnable*» ? Il nous faut également préciser ce que nous appelons un «*temps raisonnable*» : pouvons nous calculer des matrices de permutation P et Q en un temps proportionnel en un polynôme en n ? Cette question constitue le fil directeur de ce chapitre et du suivant :

QUESTION 1 *Existe-t-il un algorithme en temps polynomial pour le problème* MATRICE INF-TRIANGULAIRE *?*

Il est important de remarquer dans un premier temps que l'orientation de la matrice triangulaire considérée (en l'occurrence ici $\llcorner_n$) est sans importance sur la complexité du problème.

PROPOSITION 4.1.2 *Soit A une (0,1)-matrice d'ordre n. Les conditions suivantes sont équivalentes :*

1. *il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \llcorner_n$;*
2. *il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$;*
3. *il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangledown_n$;*
4. *il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \lrcorner_n$.*

Nous avons déjà rencontré ce problème dans le chapitre précédent dans le cas particulier où la matrice A contient au plus $n+1$ éléments non nuls. Nous avons montré dans ce cas qu'il est toujours possible de calculer deux matrices de permutation P et Q telles que $PAQ^T \leq \llcorner_n$.

tri topologique

Il convient également de remarquer que le problème obtenu en imposant $P = Q$ est un problème bien connu appelé *tri topologique* sur un graphe orienté [CLR92]. En effet, pour $P = Q$, le problème consiste à décider si le graphe orienté associé à la matrice A est acyclique. Par contre, il est important de noter dès à présent que le problème $PAP^T \leq \triangle_n$ n'est pas un tri topologique d'un graphe orienté (nous montrons que ce problème est NP-complet : problème 4.4.6 page 74.).

Le présent chapitre est organisé comme suit : Nous présentons dans un premier temps quelques rappels et définitions sur les *familles de sous-ensembles* et montrons le lien avec le problème MATRICE INF-TRIANGULAIRE. Nous montrons ensuite

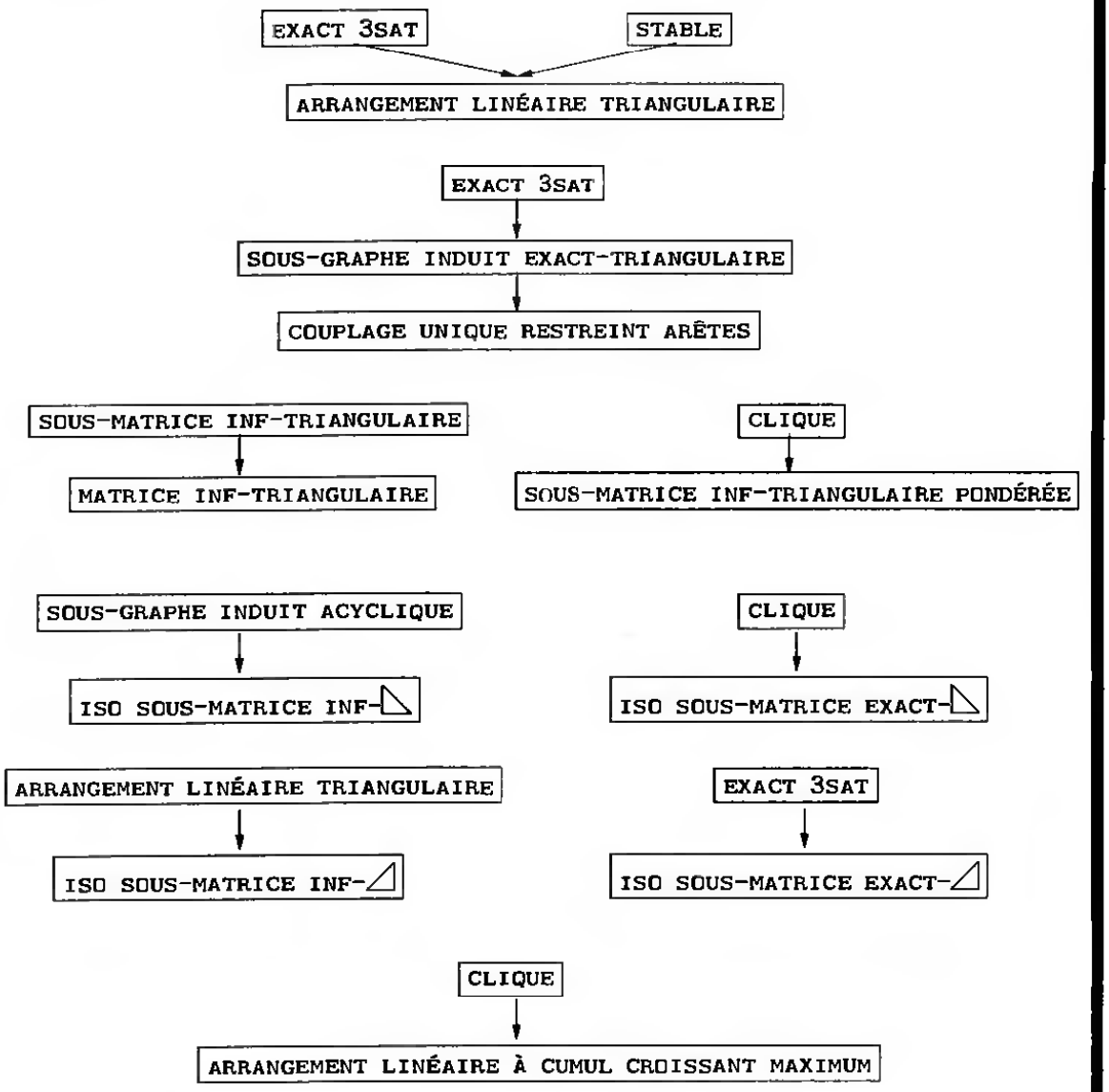


FIG. 4.1 – *Schéma des réductions polynomiales présentées dans ce chapitre.*

que ce problème est résoluble en temps polynomial lorsque le *permanent* de la matrice A est non nul. Nous présentons ensuite un problème sur les graphes appelé ARRANGEMENT LINÉAIRE TRIANGULAIRE en remarquant que si ce dernier est résoluble en temps polynomial, alors le problème MATRICE INF-TRIANGULAIRE est également résoluble en temps polynomial. Malheureusement, nous montrons que ce problème est NP-complet dans le cas général. Nous étudions ensuite différents problèmes de couplages dans les graphes bipartis et insistons en particulier sur les relations avec le problème MATRICE INF-TRIANGULAIRE. Cette étude est poursuivie dans le cas des sous-matrices principales. Enfin, nous donnons une interprétation du problème MATRICE INF-TRIANGULAIRE en termes de «*voisinages*» dans un graphe non orienté.

4.2 Famille de sous-ensembles

4.2.1 Définitions et notations

famille de sous-ensembles

Soit $X = \{x_1,x_2,\ldots,x_n\}$ un ensemble non vide de n éléments. Nous appelons X un n-ensemble. Dans la suite, nous désignons par *famille de sous-ensembles* une famille $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ de m sous-ensembles non nécessairement distincts de X [BR91]. La relation entre *famille de sous-ensembles* et (0,1)-matrice s'établit de la façon suivante en interprétant chaque ligne - ou colonne - comme un ensemble caractéristique :

- $a_{i,j} = 1$ si $x_j \in S_i$;
- $a_{i,j} = 0$ si $x_j \notin S_i$.

matrice d'incidence

La (0,1)-matrice $A = [a_{i,j}]$, de taille m par n est la *matrice d'incidence* de la famille de sous-ensembles $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ du n-ensemble X. Les éléments non nuls de la ligne i représentent les éléments du sous-ensemble S_i, tandis que les éléments non nuls de la colonne j représentent les occurrences de x_j parmi les sous-ensembles.

isomorphes

Soient $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ une famille de sous-ensembles d'un n-ensemble X et $\mathcal{F}' = \{S'_1,S'_2,\ldots,S'_m\}$ une famille de sous-ensembles d'un n-ensemble X'. Les familles $\mathcal{F}$ et $\mathcal{F}'$ sont *isomorphes* s'il est possible de renommer les sous-ensembles $S_1,S_2,\ldots,S_m$ et les éléments du n-ensemble X de telle façon que la famille obtenue coïncide avec la famille $\mathcal{F}' = \{S'_1,S'_2,\ldots,S'_m\}$ du n-ensemble X'.

L'isomorphisme de familles de sous-ensembles a une interprétation directe en termes de matrices d'incidence associées aux familles de sous-ensembles. Soient A la (0,1)-matrice d'incidence d'une famille $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ de sous-ensembles d'un n-ensemble X et A' la (0,1)-matrice d'incidence d'une famille $\mathcal{F}' = \{S'_1,S'_2,\ldots,S'_m\}$ de sous-ensembles d'un n-ensemble X'. Alors, les familles $\mathcal{F}$ et $\mathcal{F}'$ sont isomorphes si et seulement si il existe des matrices de permutation P d'ordre m et Q d'ordre n telles que :

$$PAQ^T = A'.$$

Si deux familles de sous-ensembles sont isomorphes, alors les matrices d'incidence associées de taille m par n satisfont un certain nombre des conditions nécessaires. Par exemple, les sommes des lignes doivent être les mêmes pour les deux matrices. De façon équivalente, les sommes des colonnes doivent être les mêmes pour les deux matrices. Remarquons également que $\rho(A) = \rho(B)$ [1]. Néanmoins, même si deux familles de sous-ensembles satisfont un certain nombre de conditions nécessaires, déterminer si elles sont isomorphes peut rester un problème ouvert [BR91].

La matrice d'incidence associée à une famille de m sous-ensembles d'un n-ensemble permet de représenter de façon compacte les intersections des sous-ensembles. Considérons en effet l'équation matricielle :

$$U = AA^T.$$

matrice d'intersection

La matrice $U = [u_{i,j}]$ est une matrice symétrique d'ordre m à valeurs entières positives appelée *matrice d'intersection* de A. La composante $u_{i,j}$ représente le cardinal

1. Rappelons que $\rho(A)$ est le nombre maximal d'éléments non nuls indépendants dans A.

de l'ensemble $S_i \cap S_j$. Autrement dit, nous avons :

$$u_{i,j} = |X_i \cap X_j|, \quad 1 \leq i,j \leq m.$$

En particulier, la diagonale principale de la matrice U représente les cardinaux des ensembles $S_1,S_2,\ldots,S_m$. Décider si une matrice symétrique d'ordre m à valeurs positives est la matrice d'intersection d'une famille de sous-ensembles est un problème NP-complet [GJ79] [2] et ceci même si $u_{i,i} = 3$ pour tout $1 \leq i \leq m$, c'est-à-dire même si chaque sous-ensemble contient exactement trois éléments non nuls. Cependant, si $u_{i,i} = 2$ pour tout $1 \leq i \leq m$, alors le problème est résoluble en temps polynomial [GJ79].

Il est également possible d'inverser l'ordre de la multiplication des matrices A et A^T. Considérons l'équation matricielle suivante :

$$S = A^T A.$$

La matrice $S = [s_{i,j}]$ est une matrice symétrique d'ordre n à valeurs entières positives appelée *matrice des occurrences* de A. La composante $s_{i,j}$ représente le nombre de fois où les éléments x_i et x_j apparaissent dans les sous-ensembles $S_1,S_2,\ldots,S_m$. La diagonale principale de S représente les occurrences de chacun des n éléments dans les m sous-ensembles.

matrice des occurrences

4.2.2 Système de représentants

Soient $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ une famille de sous-ensembles d'un n-ensemble X et $D = (x_1,x_2,\ldots,x_m)$ une séquence ordonnée de m éléments distincts de X. Si $x_i \in S_i$ pour tout $1 \leq i \leq m$, alors l'élément x_i *représente* l'ensemble S_i. Nous disons dans ce cas que la famille $\mathcal{F}$ possède un *système de représentants distincts*, abrégé SRD [3]. Remarquons que la définition d'un SRD impose que $x_i \neq x_j$ si $i \neq j$, mais les ensembles X_i et X_j peuvent ne pas être distincts.

SRD

En 1935, P. Hall a donné une condition nécessaire et suffisante pour qu'une famille de sous-ensembles possède un SRD.

THÉORÈME 4.2.1 *Soit $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ une famille de sous-ensembles d'un n-ensemble X. La famille $\mathcal{F}$ possède un SRD si et seulement si l'union $X_{i_1} \cup X_{i_2} \cup \ldots \cup X_{i_k}$ contient au moins k éléments pour tout $1 \leq k \leq m$ et pour tout k-sous-ensemble $\{i_1,i_2,\ldots,i_k\}$ des entiers $1,2,\ldots,m$.*

Une preuve détaillée de ce théorème est donnée, par exemple, dans [BR91]. En fait, calculer un système de représentants distincts se ramène à la recherche d'un flot maximum dans un graphe. En effet, ce problème est équivalent à la recherche d'un *couplage de cardinalité maximale* dans un graphe biparti $G = (V_1 \cup V_2,E)$. L'algorithme de Ford et Fulkerson [FF62] permet de calculer ce couplage dans un temps polynomial en $|V_1| + |V_2|$ et $|E|$. L'astuce consiste à construire un réseau de transport $G' = (V',E')$ de la façon suivante. Soient s et t deux sommets distincts tels que $s \notin V_1 \cup V_2$ et $t \notin V_1 \cup V_2$. le sommet s est appelé la *source* et t le *puits*. Le réseau de transport G' est entièrement défini par :

- $V' = V_1 \cup V_2 \cup \{s,t\}$;

2. La preuve est attribuée dans [GJ79] à Chvátal.
3. SDR en anglais.

– $E' = E \cup \{\{s,u\} \mid u \in V_1\} \cup \{\{v,t\} \mid v \in V_2\}$.

Pour compléter la construction, nous affectons une capacité unitaire à chaque arête de E'.

Le théorème suivant relie le nombre de SRD distincts d'une famille de sous-ensembles au permanent de la matrice d'incidence associée.

THÉORÈME 4.2.2 *Soient $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ une famille de sous-ensembles d'un n-ensemble X, $m \leq n$, et A la matrice d'incidence associée. Alors, le nombre de SRD distincts pour la famille $\mathcal{F}$ est donné par* $\mathrm{per}(A)$.

Autrement dit, nous pouvons décider en temps polynomial si $\mathrm{per}(A) > 0$ car nous pouvons calculer un SRD pour une famille de sous-ensembles en temps polynomial. Cependant, nous verrons par la suite que calculer le nombre de SRD distincts (et donc la valeur de $\mathrm{per}(A)$) est un problème beaucoup plus difficile.

4.2.3 Propriété de croissance

Nous allons dans cette sous-section établir un lien entre la notion de «*famille de sous-ensembles*» et le problème de la «*triangularisation d'une* (0,1)*-matrice*» par permutation des lignes et des colonnes. Pour cela, nous introduisons dans un premier temps une propriété, appelée *propriété de croissance*, sur une famille de sous-ensembles d'un ensemble fini. Nous montrons ensuite qu'une famille de sous-ensembles possède la propriété de croissance si et seulement si la matrice d'incidence associée est triangularisable par permutation des lignes et des colonnes. Enfin, nous donnons une *forme normalisée* pour les matrices triangularisables par permutation des lignes et des colonnes.

propriété de croissance

DÉFINITION 4.2.3 (PROPRIÉTÉ DE CROISSANCE) *Une famille $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ de sous-ensembles d'un n-ensemble possède la* propriété de croissance *(resp.* propriété de croissance stricte*) s'il existe une bijection π de $\mathcal{F}$ dans $[\![m]\!]$ telle que :*

$$\forall i,\ 1 \leq i \leq \min(m,n), \quad \mathrm{card}\Big(\bigcup_{\pi(S)\leq i} S\Big) \leq i \quad (\textit{resp.} = i).$$

EXEMPLE 11 *Soit $\mathcal{F} = \{S_1,S_2,S_3,S_4\}$ une famille de sous-ensembles de l'ensemble $X = \{x_1,x_2,x_3,x_4\}$ définie par $S_1 = \{x_1,x_2,x_3\}$, $S_2 = \{x_3\}$, $S_3 = \{x_2,x_3,x_4\}$ et $S_4 = \{x_2\}$. La famille $\mathcal{F}$ possède la* propriété de croissance *car la bijection π de $\mathcal{F}$ dans $[\![4]\!]$ définie par $\pi(S_1) = 3$, $\pi(S_2) = 2$, $\pi(S_3) = 4$ et $\pi(S_4) = 1$ vérifie :*

$$\mathrm{card}\Big(\bigcup_{\pi(S)\leq 1} S = \{x_2\}\Big) \leq 1 \qquad \mathrm{card}\Big(\bigcup_{\pi(S)\leq 2} S = \{x_2,x_3\}\Big) \leq 2$$
$$\mathrm{card}\Big(\bigcup_{\pi(S)\leq 3} S = \{x_1,x_2,x_3\}\Big) \leq 3 \quad \mathrm{card}\Big(\bigcup_{\pi(S)\leq 4} S = \{x_1,x_2,x_3,x_4\}\Big) \leq 4.$$

REMARQUE 4.2.4 Soit $\mathcal{F}$ une famille de m sous-ensembles non nécessairement distincts d'un n-ensemble X. Si $\mathcal{F}$ possède la propriété de croissance par une bijection π, alors les conditions suivantes sont satisfaites :

1. $\pi(S) \geq |S|$ pour tout $S \in \mathcal{F}$;
2. $\mathrm{card}(\{S \mid |S| \geq i\}) \leq n - i + 1$ pour tout $1 \leq i \leq n$.

En effet, pour tout $S \in \mathcal{F}$, nous avons :

$$|S| \leq \text{card}\Big(\bigcup_{\pi(S')\leq\pi(S)} S'\Big) \leq \pi(S)$$

Supposons que la seconde condition ne soit pas vérifiée. Alors, il existe un entier k, $1 < k \leq n$, tel que :

$$\text{card}(\{S \mid |S| \geq k\}) > n - k + 1.$$

Par conséquent, il existe $S \in \mathcal{F}$ tel que $|S| \geq k$ et $\pi(S) \leq k - 1$. Ce qui est en contradiction avec la première condition. ◇

La propriété de croissance est liée au problème de la mise sous forme triangulaire d'une (0,1)-matrice par permutations des lignes et des colonnes. En effet, en reprenant l'exemple 11, nous pouvons représenter la famille $\mathcal{F}$ ainsi que la bijection π de la façon suivante :

$$\begin{array}{lllll} S_4 = \{ & & x_2 & & \} \\ S_2 = \{ & & & x_3 & \} \\ S_1 = \{ & x_1, & x_2, & x_3 & \} \\ S_3 = \{ & x_1, & x_2, & x_3, & x_4 \}. \end{array}$$

En ordonnant ensuite les colonnes par hauteur maximale d'un élément non nul, nous obtenons la représentation ci-après :

$$\begin{array}{lllll} S_4 = \{ & x_2 & & & \} \\ S_2 = \{ & & x_3 & & \} \\ S_1 = \{ & x_2, & x_3, & x_1 & \} \\ S_3 = \{ & x_2, & x_3, & x_1, & x_4 \}. \end{array}$$

En fait cette représentation intuitive d'une famille de sous-ensembles n'est autre que la matrice d'incidence de cette dernière dont les lignes et les colonnes ont été permutées. Nous pouvons formaliser cette remarque par le lemme suivant :

PROPOSITION 4.2.5 *Soient $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ une famille de sous-ensembles d'un n-ensemble X et $A = [a_{i,j}]$ la (0,1)-matrice d'incidence d'ordre n associée à $\mathcal{F}$. Il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$ si et seulement si la famille $\mathcal{F}$ possède la propriété de croissance.*

La bijection π qui confère à une famille de sous-ensembles la propriété de croissance n'est pas nécessairement unique. Par exemple, en reprenant l'exemple 11, nous pouvons vérifier que la bijection $\pi'(1) = 3$, $\pi'(2) = 1$, $\pi'(3) = 4$ et $\pi'(4) = 2$ satisfait également les conditions. Nous allons cependant donner une «*forme normalisée*» (ce résultat technique est une commodité qui sera utilisée dans la suite).

PROPOSITION 4.2.6 *Soit $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ une famille de sous-ensembles d'un n-ensemble X. Si $\mathcal{F}$ possède la propriété de croissance par une bijection π de $\mathcal{F}$ dans $[\![n]\!]$, alors nous pouvons toujours supposer (pour un coût polynomial) que la bijection π satisfait également les deux conditions suivantes :*

1. $\forall S_i, S_j \in \mathcal{F}, \quad S_i \subset S_j \quad \Rightarrow \quad \pi(S_i) < \pi(S_j)$;

2. $\forall S_i,S_j,S_k \in \mathcal{F}, \qquad \pi(S_i) < \pi(S_j) < \pi(S_k) \;\; et \;\; S_i = S_k \quad \Rightarrow \quad S_i = S_j.$

Preuve

Nous allons montrer qu'il existe une bijection qui confère à la famille $\mathcal{F}$ la propriété de croissance qui satisfait de plus les deux conditions demandées. Dans une première étape, nous construisons une nouvelle bijection qui respecte l'ordre d'inclusion (condition 1). Ensuite, nous montrons que cette nouvelle bijection peut être modifiée afin de satisfaire également la seconde condition.

Notons Π l'ensemble des bijections de $\mathcal{F}$ dans $[\![n]\!]$. Pour toute bijection $\pi \in \Pi$, considérons l'ensemble $A(\pi)$ de paires ordonnées défini par :

$$A(\pi) = \{(S_i,S_j) \mid \pi(S_i) < \pi(S_j) \;\text{ et }\; S_j \subset S_i\}.$$

Si $A(\pi) = \emptyset$, alors la condition 1 est satisfaite. Soit $(S_i,S_j) \in A(\pi)$ une paire ordonnée. Nous pouvons toujours supposer $S_j \not\subset S_k$ et $S_k \not\subset S_j$ pour tout S_k tel que $\pi(S_i) < \pi(S_k) < \pi(S_j)$. Considérons maintenant la bijection $\pi' \in \Pi$ obtenue à partir de π par transposition des éléments S_i et S_j. En notant $i = \pi(S_i)$ et $j = \pi(S_j)$ pour simplifier l'écriture, il vient :

$$\begin{aligned}
\forall \ell,\, 1 \le \ell < i \;\text{ ou }\; j \le \ell \le m, \qquad & \bigcup_{\pi'(S)\le\ell} S = \bigcup_{\pi(S)\le\ell} S \\
\forall \ell,\, 1 \le \ell < j, \qquad & \bigcup_{\pi'(S)\le\ell} S = \Big(\bigcup_{\pi'(S)<i} S\Big) \cup S_j \cup \Big(\bigcup_{i<\pi'(S)\le\ell} S\Big) \\
& \subseteq \Big(\bigcup_{\pi(S)<i} S\Big) \cup S_i \cup \Big(\bigcup_{i<\pi(S)\le\ell} S\Big) \\
& = \bigcup_{\pi(S)\le\ell} S.
\end{aligned}$$

Aussi, la bijection π' confère à la famille $\mathcal{F}$ la propriété de croissance. Montrons que $A(\pi') \subset A(\pi)$. En effet, supposons qu'il existe une paire ordonnée $(S_k,S_\ell) \in A(\pi')$ telle que $(S_k,S_\ell) \notin A(\pi)$. Alors, $S_\ell \subset S_k$, $\pi'(S_k) < \pi'(S_\ell)$ et $\pi(S_k) > \pi(S_\ell)$. Remarquons dans un premier temps que par construction de la bijection π', $(S_k,S_\ell) \neq (S_i,S_j)$. Envisageons les deux autres cas possibles :

1. Si $S_\ell = S_i$ et $\pi(S_i) < \pi(S_k) < \pi(S_j)$, alors $S_j \subset S_k$; ce qui est une contradiction car $S_j \not\subset S_k$.
2. Si $S_k = S_j$ et $\pi(S_i) < \pi(S_\ell) < \pi(S_j)$, alors $S_\ell \subset S_j$; ce qui est une contradiction car $S_\ell \not\subset S_j$.

En conséquence, $A(\pi') \subset A(\pi)$. Aussi, nous pouvons appliquer l'algorithme décrit ci-dessus à la bijection π' elle-même tant que $A(\pi') \neq \emptyset$. Nous supposons dorénavant que la bijection π qui confère à la famille $\mathcal{F}$ la propriété de croissance satisfait également l'ordre d'inclusion (condition 1).

Il reste à montrer que la seconde condition peut également être satisfaite. Soient S_i, S_j et S_k trois sous-ensembles de $\mathcal{F}$ tels que $S_i = S_j \neq S_k$ et $\pi(S_i)+1 = \pi(S_k) < \pi(S_j)$ (s'ils n'existent pas, la seconde condition est vérifiée). Considérons la bijection

$\pi' \in \Pi$ définie par :

$$\begin{cases} \forall S \in \mathcal{F}, 1 \leq \pi'(S) \leq \pi'(S_i), \quad \pi''(S) = \pi'(S) \\ \forall S \in \mathcal{F}, \pi'(S_i) < \pi'(S) < \pi'(S_j), \quad \pi''(S) = \pi'(S) + 1 \\ \pi''(S_j) = \pi'(S_k) \\ \forall S \in \mathcal{F}, \pi'(S_j) < \pi'(S) \leq m, \quad \pi''(S) = \pi'(S). \end{cases}$$

En notant $k = \pi(S_k)$ et $j = \pi(S_j)$ pour simplifier l'écriture, il vient :

$$\begin{aligned} \forall \ell, 1 \leq \ell \leq i \vee j \leq \ell \leq n, \qquad & \bigcup_{\pi'(S) \leq \ell} S = \bigcup_{\pi(S) \leq \ell} S \\ \ell = k(= i+1), \qquad & \bigcup_{\pi'(S) \leq \ell} S = (\bigcup_{\pi'(S) \leq i} S) \cup S_j \\ & \subseteq (\bigcup_{\pi(S) \leq i} S) \cup S_k \\ & = \bigcup_{\pi(S) \leq \ell} S \\ \forall \ell, k < \ell < j, \qquad & \bigcup_{\pi'(S) \leq \ell} S = (\bigcup_{\pi'(S) \leq i} S) \cup S_j \cup (\bigcup_{k < \pi'(S) \leq \ell} S) \\ & = (\bigcup_{\pi(S) \leq i} S) \cup S_j \cup (\bigcup_{k \leq \pi(S) \leq \ell - 1} S) \\ & \subseteq \bigcup_{\pi(S) \leq \ell} S. \end{aligned}$$

Aussi, cette nouvelle bijection π' confère à la famille $\mathcal{F}$ la propriété de croissance et respecte toujours l'ordre d'inclusion. En appliquant un argument identique à celui utilisé lors de la construction de la bijection respectant l'ordre d'inclusion, nous concluons qu'il existe une bijection π qui confère à la famille $\mathcal{F}$ la propriété de croissance et qui satisfait également les deux conditions énoncées. □

La proposition précédente nous permet toujours de supposer (pour un coût polynomial) que :

- un sous-ensemble S_i strictement inclus dans un sous-ensemble S_j est placé avant ce dernier par la bijection π;
- les sous-ensembles égaux apparaissent consécutivement dans la bijection π.

4.3 Permanent non nul

4.3.1 Introduction

Nous revenons maintenant au problème concernant la triangularisation d'une (0,1)-matrice par permutations des lignes et des colonnes. Plus formellement, étant donnée une (0,1)-matrice A d'ordre n, existe-t-il des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$? Nous nous intéressons dans cette section au cas particulier où le permanent de la matrice considérée est non nul.

Rappelons que le *permanent* d'une (0,1)-matrice $A = [a_{i,j}]$ d'ordre n est défini *permanent*

par :

$$\mathrm{per}(A) = \sum_{\pi \in \mathcal{S}_n} \prod_{i=1}^{n} a_{i,\pi(i)}.$$

À la différence du calcul du *déterminant* d'une matrice, tous les termes ont un signe positif. Une des principales raisons de l'intérêt du calcul du permanent est qu'une (0,1)-matrice $A = [a_{i,j}]$ d'ordre n peut être considérée comme la *matrice d'adjacence réduite* d'un graphe biparti $G = (X \cup Y, E)$, où l'ensemble $X = \{x_1, x_2, \cdots, x_n\}$ indice les lignes de A, l'ensemble $Y = \{y_1, y_2, \cdots, y_n\}$ indice les colonnes de A et $a_{i,j} = 1$ s'il existe une arête entre x_i et y_j. La valeur de $\mathrm{per}(A)$ est exactement le nombre de couplages parfaits dans G.

Malgré la similarité des définitions, permanent et déterminant sont deux notions radicalement différentes du point de vue de la complexité du calcul. En effet, bien qu'il existe des algorithmes en temps polynomiaux pour le calcul du déterminant, le meilleur algorithme pour l'évaluation du permanent d'une matrice d'ordre n nécessite $O(n2^n)$ opérations arithmétiques. En fait, L.G. Valiant [Val79] a montré que le calcul du permanent d'une (0,1)-matrice (ou, de façon équivalente, le nombre de couplages parfaits distincts dans un graphe biparti) est un problème #P-complet. Une implication forte de ce résultat est que si $\mathsf{P} \neq \mathsf{NP}$, alors il n'existe aucun algorithme en temps polynomial pour calculer $\mathrm{per}(A)$.

REMARQUE 4.3.1 Le permanent de A est invariant par permutation des lignes et des colonnes. Autrement dit, $\mathrm{per}(PAQ^T) = \mathrm{per}(A)$ pour toutes les matrices de permutation P et Q d'ordre n. ◇

REMARQUE 4.3.2 Si A est une (0,1)-matrice d'ordre n telle que $\mathrm{per}(A) > 0$, alors il existe des matrices de permutation P et Q d'ordre n telle que la matrice PAQ^T a des 1 sur tous les éléments de sa diagonale principale, *i.e.* $PAQ^T \geq \mathbb{I}_n$. En effet, il suffit de calculer un couplage parfait dans le graphe biparti associé. ◇

REMARQUE 4.3.3 Soit A une (0,1)-matrice d'ordre n. Si $\mathrm{per}(A) = 1$, alors il existe une unique matrice de permutation R d'ordre n telle que $R \leq A$. ◇

Le théorème suivant caractérise complètement les (0,1)-matrices d'ordre n dont le permanent est 1.

THÉORÈME 4.3.4 ([BRU66]) *Soit A une (0,1)-matrice d'ordre n. Alors,* $\mathrm{per}(A) = 1$ *si et seulement si il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \llcorner_n$ et la matrice PAQ^T a des 1 sur tous les éléments de sa diagonale principale,* i.e. $PAQ^T \geq \mathbb{I}_n$.

Remarquons que nous pouvons vérifier en temps polynomial si $\mathrm{per}(A) > 0$ en recherchant un couplage parfait dans le graphe biparti associé [HK73].

COROLLAIRE 4.3.5 *Soit A une (0,1)-matrice d'ordre n. S'il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \llcorner_n$, alors* $\mathrm{per}(A) \leq 1$.

4.3.2 Algorithme

Le théorème 4.3.4 nous permet de donner un algorithme pour calculer (si elles existent) des matrices de permutation P et Q telles que $PAQ^T \leq \triangle_n$. Étant donnée une (0,1)-matrice A, l'algorithme `calcul-mat-`PQ^T recherche à chaque itération une ligne qui contient un unique élément non nul. Cette ligne et la colonne correspondante sont ensuite supprimées. Remarquons tout de suite que s'il existe une ligne qui ne contient aucun élément non nul, alors $\text{per}(A) = 0$.

Algorithm 6:

`calcul-mat-`PQ^T

Entrée: Une (0,1)-matrice A d'ordre n avec $\text{per}(A) > 0$

Sortie: Deux matrices de permutation P et Q^T d'ordre n (si elles existent) telles que $PAQ^T \leq \triangle_n$

début
- **pour** *k de 1 à n* **faire**
  - **si** *il existe une ligne i de A qui contient un unique élément non nul* **alors**
    - Soit j la colonne de l'unique élément non nul de la ligne i.
    - Supprimer la ligne d'indice i et la colonne d'indice j.
  - **sinon**
    - Retourner un échec (car $\text{per}(A) > 1$).
- La matrice de permutation P (resp. Q^T) d'ordre n est calculée à partir de l'ordre d'élimination des lignes (resp. colonnes) dans la boucle. Retourner les matrices de permutation P et Q^T.

fin

Un exemple d'exécution de l'algorithme `calcul-mat-`PQT pour une (0,1)-matrice dont le permanent est exactement 1 est donné en figure 4.2.

THÉORÈME 4.3.6 ([BR91]) *Soit A une (0,1)-matrice d'ordre n. Si A a au moins un élément non nul sur chaque ligne, alors A contient une sous-matrice principale qui est une matrice de permutation.*

Nous prouvons maintenant la validité de l'algorithme `calcul-mat-`PQT.

PROPOSITION 4.3.7 *L'algorithme `calcul-mat-`PQ^T calcule les matrices de permutation P et Q^T d'ordre n telles que $PAQ^T \leq \triangle_n$ si $\text{per}(A) = 1$ et retourne un échec si $\text{per}(A) > 1$.*

PREUVE

Par induction sur la taille de la (0,1)-matrice (voir théorème 4.3.4 [Bru66]). Il suffit de montrer qu'à chaque itération il existe une ligne qui contient un unique élément non nul si $per(A) = 1$.

Le résultat est immédiat si $n = 1$. Si $\text{per}(A) > 0$, alors il existe au moins une suite $(j_1, j_2, \ldots, j_n)$ d'indices telle que $a_{i,j_i} = 1$ pour $1 \leq i \leq j$. Notons $D = [d_{i,j}]$ la matrice de permutation définie par $d_{i,i_j} = 1$ pour $1 \leq i \leq n$. Supposons que chaque ligne de

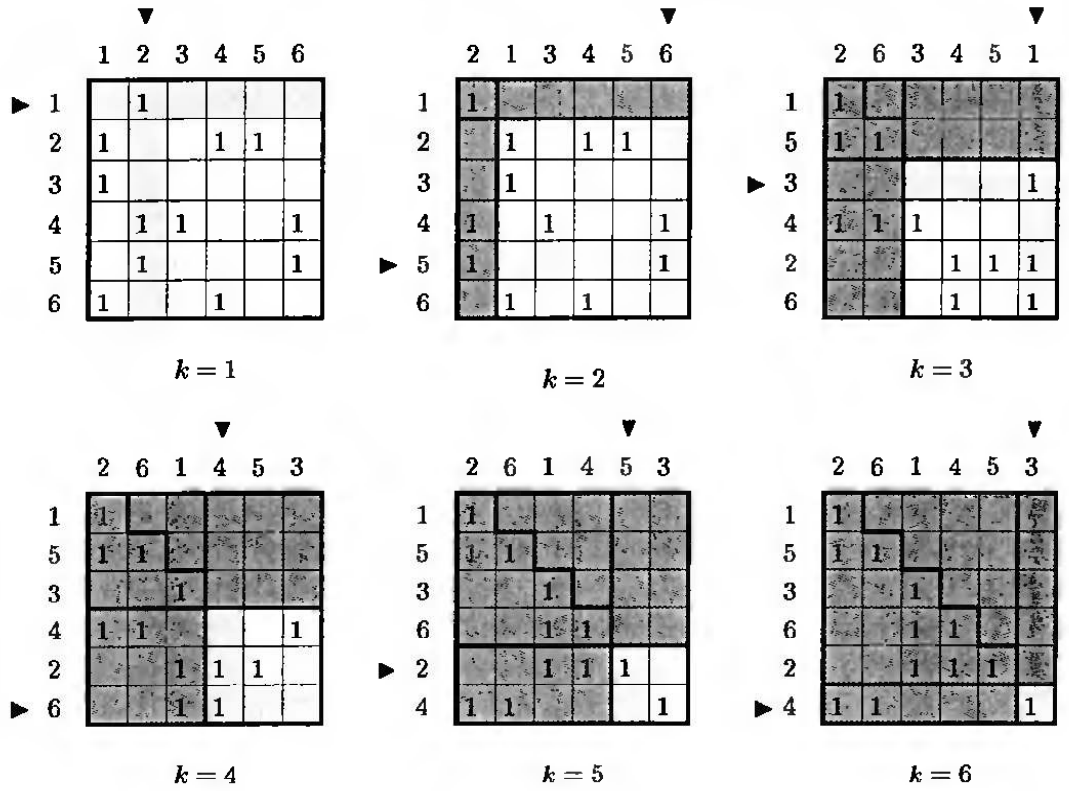


FIG. 4.2 – *Exemple de calcul des permutations P et Q^T d'ordre 6 retournées par l'algorithme* `calcul-mat-PQ`^{T}. *Par souci de clarté, les éléments nuls ne sont pas représentés dans la matrice. Les curseurs ▶ et ▼ indiquent à chaque itération la ligne et la colonne qui sont sélectionnées. La partie grisée représente les lignes et les colonnes déjà traitées aux itérations précédentes.*

la matrice A contienne au moins deux éléments non nuls. Alors, la matrice $A - D$ contient au moins un élément non nul sur chaque ligne, et, en utilisant le théorème 4.3.6, elle contient une sous-matrice principale qui est une matrice de permutation. Par suite, $\mathrm{per}(A) > 1$, et par le théorème 4.3.4, il n'existe pas des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$. Sinon, la matrice A possède une ligne i qui contient un unique élément non nul en colonne j. Il suffit alors de permuter les lignes 1 et i, les colonnes 1 et j, et d'appliquer l'hypothèse d'induction sur la matrice d'ordre $n-1$ obtenue en supprimant la première ligne et la première colonne. $\boxed{4.3.7}$

Nous allons maintenant montrer que l'algorithme ***calcul-mat-PQ***^{T} est en temps $O(n^2)$, où n est l'ordre de la (0,1)-matrice A. Soit $f(n)$ la complexité du calcul lorsqu'il est réussi. Nous supposons disposer d'un tableau `T` de n places où `T[`i`]`, $1 \leq i \leq n$, est le nombre d'éléments non nuls de la ligne i. La recherche d'un indice i, $1 \leq i \leq n$, vérifiant `T[i]` $= 1$ se fait en temps $O(n)$, ainsi que les échanges lignes et colonnes. La mise à jour après échange du tableau `T` est également $O(n)$ en temps. Donc, avant d'appliquer l'algorithme ***calcul-mat-PQ*** à la sous-matrice d'ordre $n-1$, le coût des opérations est cn où c est une constante indépendante de n. Nous obtenons donc la récurrence $f(n) = cn + f(n-1)$ qui admet une solution quadratique comme énoncé.

REMARQUE 4.3.8 Si $\mathrm{per}(A) > 1$, alors il existe au moins une matrice de permutation R d'ordre n telle que $R \leq A$. Dans le cas particulier où $\mathrm{per}(A) = 1$, cette matrice est donnée par P^TQ où P et Q sont deux matrices de permutation d'ordre n telles que $PAQ^T \leq \triangle_n$. En effet, dans un premier temps, $\mathbb{I}_n \leq PAQ^T$ car la matrice PAQ^T a des 1 sur tous les éléments de la diagonale principale. Par suite, nous avons $P^TQ = P^T\mathbb{I}_nQ \leq A$. ◇

Nous avons dit (section 4.2.1) que deux familles $\mathcal{F}$ et $\mathcal{F}'$ de m sous-sensembles d'un n-ensemble X sont isomorphes s'il existe des matrices de permutation P d'ordre m et Q d'ordre n telles que :

$$PAQ^T = A',$$

où A et A' sont respectivement les matrices d'incidence associées aux familles $\mathcal{F}$ et $\mathcal{F}'$. Décider s'il existe des matrices P et Q telles que $PAQ^T = A'$ est un problème ouvert [BR91]. Nous supposons dans la suite $m = n$. Nous allons montrer que dans le cas particulier où $\mathrm{per}(A) = \mathrm{per}(A') = 1$, $A \leq \triangle_n$ et $A' \leq \triangle_n$, le problème est équivalent à décider s'il existe une matrice de permutation R d'ordre n telle que :

$$RAR^T = A'.$$

REMARQUE 4.3.9 Si $PQ^T \leq \triangle_n$, alors $P = Q$. ◇

PROPOSITION 4.3.10 *Soient A et A' deux (0,1)-matrices d'ordre n telles que* $\mathrm{per}(A) = \mathrm{per}(A') = 1$, $A \leq \triangle_n$ *et* $A' \leq \triangle_n$. *S'il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T = A'$, alors $P = Q$.*

PREUVE

Notons B la (0,1)-matrice $A - \mathbb{I}_n$. La matrice B est bien définie car la matrice A a des 1 sur tous les éléments de sa diagonale principale (lemme 4.3.4). Si $PAQ^T = A'$,

alors $PBQ^T + P\mathbb{I}_nQ^T = A'$. Mais, $A' \leq \triangle_n$, et par suite $P\mathbb{I}_nQ^T \leq \triangle_n$. D'où, $P = Q$, par la remarque 4.3.9. $\boxed{4.3.10}$

Remarquons que, dans le cas général, étant données deux (0,1)-matrices A et A' d'ordre n, décider s'il existe une matrice de permutation R d'ordre n telle que $RAR^T = A'$ est un problème ouvert appelé ISOMORPHISME DE GRAPHES [GJ79].

4.3.3 Conclusion

Nous avons montré dans cette section que dans le cas particulier où per$(A) > 0$, le problème MATRICE INF-TRIANGULAIRE est résoluble en temps polynomial : l'algorithme *calcul-mat-*PQ^T calcule en temps $O(n^2)$ des matrices de permutation P et Q^T d'ordre n telles que $PAQ^T \leq \triangle_n$ si per$(A) = 1$; si per$(A) > 1$, alors $PAQ^T \not\leq \triangle_n$ pour toutes les matrices de permutation P et Q d'ordre n. Il nous reste donc à considérer dans la suite le cas per$(A) = 0$.

THÉORÈME 4.3.11 ([ES73]) *Le nombre de* (0,1)*-matrices d'ordre n dont le permanent est nul est donné asymptotiquement par* $n2^{n^2-2n+1}$.

Une conséquence importante de ce théorème est que presque toutes les (0,1)-matrices d'ordre n ont un permanent non nul. Par conséquent, le problème MATRICE INF-TRIANGULAIRE est résoluble en temps polynomial (en utilisant par exemple l'algorithme *calcul-mat-*PQ^T) pour «*presque toutes*» les (0,1)-matrices d'ordre n. Signalons pour terminer que presque toutes les (0,1)-matrices d'ordre n ont un déterminant non nul [BR91].

4.4 Arrangement linéaire triangulaire

4.4.1 Introduction

Face à un problème de décision sur les (0,1)-matrices, il est naturel de tenter de le coder comme un problème sur les graphes. Nous pouvons en effet toujours supposer qu'une (0,1)-matrice $A = [a_{i,j}]$ d'ordre n est la matrice d'adjacence d'un graphe orienté $G = (V,E)$ également d'ordre n défini par :

$$V = \{u_1, u_2, \cdots, u_n\}$$
$$E = \{(u_i, u_j) \mid a_{i,j} = 1\}.$$

Lorsque la matrice A est symétrique, le graphe associé peut être supposé non orienté :

$$E = \{\{u_i, u_j\} \mid a_{i,j} = a_{j,i} = 1\}.$$

Rappelons que deux graphes G et H d'ordre n sont *isomorphes*, noté $G \cong H$, s'il existe une matrice de permutation P d'ordre n telle que $PA_GP^T = A_H$ où A_G (resp. A_H) est la matrice d'adjacence du graphe G (resp. H). Remarquons que la matrice de permutation P «*agit*» sur le graphe G comme une «*renumérotation*» des sommets. Que faire dans le cas PAQ^T où deux matrices de permutation sont utilisées comme c'est par exemple le cas dans le problème MATRICE INF-TRIANGULAIRE? Nous pouvons difficilement considérer ces deux matrices de permutation comme une

«*double renumérotation*» des sommets du graphe (nous verrons cependant que, dans le cas particulier des *graphes bipartis* et des *graphes de séparation*, cette opération a un sens).

Nous allons dans cette section montrer qu'il existe un problème de renumérotation des sommets d'un graphe non orienté (ARRANGEMENT LINÉAIRE TRIANGULAIRE) tel que si ce dernier est résoluble en temps polynomial, alors la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes est également résoluble en temps polynomial. Malheureusement, ce problème est difficile et nous montrerons qu'il est NP-complet.

Nous considérons dans la suite le problème de décision suivant :

PROBLÈME 4.4.1 (ISO MATRICE INF-$\triangle$)
DONNÉES : Une (0,1)-matrice symétrique A d'ordre n.
QUESTION : Existe-t-il une matrice de permutation R d'ordre n telle que $RAR^T \leq \triangle_n$?

EXEMPLE 12 *Soit A la matrice de permutation d'ordre 5 définie par :*

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Il existe une matrice de permutation P d'ordre 5 telle que :

$$\begin{aligned} PAP^T &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \\ &\leq \triangle_5. \end{aligned}$$

REMARQUE 4.4.2 Puisque $A = [a_{i,j}]$ est une (0,1)-matrice symétrique d'ordre n, le problème obtenu en remplaçant la condition «$RAR^T \leq \triangle_n$» par «$RAR^T \leq \triangleright_n$» admet une solution si et seulement si $a_{i,j} = 0$ pour $1 \leq i,j \leq n$ et $i \neq j$. ◇

Nous allons maintenant montrer que le problème ISO MATRICE INF-$\triangle$ est relié à notre problème initial MATRICE INF-TRIANGULAIRE. Le lemme suivant caractérise complètement les matrices de permutation solutions du problème ISO MATRICE INF-$\triangle$ dans le cas où A est une (0,1)-matrice symétrique d'ordre $2n$ qui contient une sous-matrice principale égale à $\mathbf{1}_{n,n}$. Rappelons que la (0,1)-matrice $\mathbf{1}_{n,n}$ est définie par $\mathbf{1}_{i,j} = 1$ pour $1 \leq i,j \leq n$.

LEMME 4.4.3 *Soient A une (0,1)-matrice symétrique d'ordre $2n$ et P une matrice de permutation d'ordre $2n$ telle que $PAP^T \leq \triangle_{2n}$. Si A est de la forme :*

$$A = \begin{bmatrix} \mathbf{0}_{n,n} & B \\ B^T & \mathbf{1}_{n,n} \end{bmatrix}$$

où B est une (0,1)-matrice d'ordre n, alors P est de la forme :

$$P = \begin{bmatrix} P_1 & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & P_2 \end{bmatrix}$$

où P_1 et P_2 sont des matrices de permutation d'ordre n.

PREUVE

$$\begin{aligned} PAP^T \leq \triangle_{2n} \quad &\Leftrightarrow \quad \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} \mathbf{0}_{n,n} & B \\ B^T & \mathbf{1}_{n,n} \end{bmatrix} \begin{bmatrix} P_{11}^T & P_{21}^T \\ P_{12}^T & P_{22}^T \end{bmatrix} \leq \triangle_{2n} \\ &\Rightarrow \quad P_{12}B^TP_{11}^T + (P_{11}B + P_{12}\mathbf{1}_{n,n})P_{12}^T = 0 \\ &\Rightarrow \quad P_{12}\mathbf{1}_{n,n}P_{12}^T = \mathbf{0}_{n,n} \\ &\Rightarrow \quad P_{12} = \mathbf{0}_{n,n}. \end{aligned}$$

Mais, si $P_{12} = 0$, alors la matrice P_{11} est une matrice de permutation d'ordre n. Par conséquent, $P_{21} = \mathbf{0}_{n,n}$ et P_{22} est une matrice de permutation d'ordre n. Autrement dit, s'il existe une matrice de permutation P d'ordre $2n$ telle que $PAP^T \leq \triangle_{2n}$, alors P est de la forme

$$P = \begin{bmatrix} P_1 & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & P_2 \end{bmatrix}.$$

$\boxed{4.4.3}$

Nous allons montrer que, s'il existe un algorithme polynomial pour le problème ISO MATRICE INF-$\triangle$, alors il existe également un algorithme polynomial pour le problème MATRICE INF-TRIANGULAIRE. Plus précisément, nous associons à toute instance I du problème MATRICE INF-TRIANGULAIRE une instance $I' = f(I)$ calculable en temps polynomial du problème ISO MATRICE INF-$\triangle$ telle qu'il existe une solution pour I si et seulement si il existe une solution pour I'.

PROPOSITION 4.4.4 *Soit A une (0,1)-matrice symétrique d'ordre $2n$. Si A est de la forme :*

$$A = \begin{bmatrix} \mathbf{0}_{n,n} & B \\ B^T & \mathbf{1}_{n,n} \end{bmatrix}$$

où B est une (0,1)-matrice d'ordre n, alors il existe une matrice de permutation R d'ordre $2n$ telle que $RAR^T \leq \triangle_{2n}$ si et seulement si il existe des matrices de permutation P et Q d'ordre n telles que $PBQ^T \leq \triangle_n$.

PREUVE

Soit R une matrice de permutation d'ordre $2n$ et considérons sa décomposition par blocs donnée par :

$$R = \begin{bmatrix} P_{11} & P_{12} \\ Q_{21} & Q_{22} \end{bmatrix}$$

où P_{11}, P_{12}, Q_{21} et Q_{22} sont des (0,1)-matrice d'ordre n. En utilisant le lemme 4.4.3, si $RAR^T \leq \triangle_{2n}$, alors R est de la forme :

$$R = \begin{bmatrix} P & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & Q \end{bmatrix}$$

où P et Q sont des matrices de permutation d'ordre n. D'où,

$$\begin{bmatrix} P & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & Q \end{bmatrix} \begin{bmatrix} \mathbf{0}_{n,n} & B \\ B^T & \mathbf{1}_{n,n} \end{bmatrix} \begin{bmatrix} P^T & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & Q^T \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n,n} & PBQ^T \\ QB^TP^T & \mathbf{1}_{n,n} \end{bmatrix} \leq \triangle_{2n}$$

et par suite, $PBQ^T \leq \triangle_n$.

Inversement, s'il existe des matrices de permutation P et Q d'ordre n telles que $PBQ^T \leq \triangle_n$, alors

$$\begin{aligned} PBQ^T \leq \triangle_n &\Leftrightarrow B \leq P^T \triangle_n Q \\ &\Leftrightarrow B^T \leq Q^T \triangle_n^T P. \end{aligned}$$

Or, $\triangle_n^T = \triangle_n$. Par conséquent, $B^T \leq Q^T \triangle_n P$, et par suite, $QB^TP^T \leq \triangle_n$. Alors,

$$\begin{bmatrix} \mathbf{0}_{n,n} & PBQ^T \\ QB^TP^T & \mathbf{1}_{n,n} \end{bmatrix} \leq \triangle_{2n}.$$

En remarquant que :

$$\begin{bmatrix} P & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & Q \end{bmatrix} \begin{bmatrix} \mathbf{0}_{n,n} & B \\ B^T & \mathbf{1}_{n,n} \end{bmatrix} \begin{bmatrix} P^T & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & Q^T \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n,n} & PBQ^T \\ QB^TP^T & \mathbf{1}_{n,n} \end{bmatrix},$$

il vient $RAR^T \leq \triangle_{2n}$ pour :

$$R = \begin{bmatrix} P & \mathbf{0}_{n,n} \\ \mathbf{0}_{n,n} & Q \end{bmatrix}.$$

$\boxed{4.4.4a}$

4.4.2 Arrangement linéaire triangulaire

Nous avons vu précédemment que s'il existe un algorithme en temps polynomial pour le problème ISO MATRICE INF-$\triangle$, alors il existe également un algorithme en temps polynomial pour le problème MATRICE INF-TRIANGULAIRE. Cependant, nous allons montrer maintenant que ISO MATRICE INF-$\triangle$ est un problème NP-complet. C'est-à-dire que, sauf si P = NP, il n'existe aucun algorithme en temps polynomial pour le problème ISO MATRICE INF-$\triangle$. La définition suivante traduit le problème en termes de graphe.

DÉFINITION 4.4.5 (ARRANGEMENT LINÉAIRE TRIANGULAIRE) *Soit $G = (V,E)$ un graphe d'ordre n. Un* arrangement linéaire triangulaire *de G est une bijection σ de V dans $[\![n]\!]$ telle que $\sigma(u) + \sigma(v) > n$ pour tout $\{u,v\} \in E$.*

Nous appelons ARRANGEMENT LINÉAIRE TRIANGULAIRE le problème de décision associé :

PROBLÈME 4.4.6 (ARRANGEMENT LINÉAIRE TRIANGULAIRE)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n.
QUESTION : Existe-t-il un arrangement linéaire triangulaire de G?

Étant donné un graphe $G = (V,E)$, il existe une bijection σ de V dans $[\![n]\!]$ telle que $\sigma(u) + \sigma(v) > n$ pour tout $\{u,v\} \in E$ si et seulement si il existe une matrice de permutation P d'ordre n telle que $PAP^T \leq \triangle_n$, où A est la (0,1)-matrice d'adjacence du graphe G. Il suffit donc de montrer qu'il existe un algorithme en temps polynomial pour le problème ARRANGEMENT LINÉAIRE TRIANGULAIRE pour prouver que le problème ISO MATRICE INF-$\triangle$ est également résoluble en temps polynomial. Malheureusement, le problème ARRANGEMENT LINÉAIRE TRIANGULAIRE est NP-complet comme nous allons le montrer, et par conséquent nous ne pouvons rien conclure quant à la complexité du problème MATRICE INF-TRIANGULAIRE.

Nous considérons dans un premier temps une variante simple du problème EXACT 3SAT : soit ϕ une formule booléenne sous forme normale conjonctive d'ordre 3 contenant exactement m clauses, existe-t-il une interprétation f qui rend vraie au moins $m-1$ clauses de la formule booléenne ϕ? Nous appelons $\{m-1,m\}$ EXACT 3SAT cette variante du problème EXACT 3SAT.

PROPOSITION 4.4.7 $\{m-1,m\}$ EXACT 3SAT *est un problème* NP-*complet.*

PREUVE

Soit une instance arbitraire du problème EXACT 3SAT donnée par une formule booléenne ϕ sous forme normale conjonctive d'ordre 3, c'est à dire avec exactement 3 littéraux par clause. Nous notons $C = \{C_1,C_2,\ldots,C_m\}$ l'ensemble des clauses,

et $X = \{x_1,x_2,\ldots,x_n\}$ l'ensemble des variables booléennes. Soient a,b et c trois nouvelles variables booléennes et

$$\begin{array}{ll} C_{m+1} = a \vee b \vee c & C_{m+5} = \bar{a} \vee b \vee c \\ C_{m+2} = a \vee b \vee \bar{c} & C_{m+6} = \bar{a} \vee b \vee \bar{c} \\ C_{m+3} = a \vee \bar{b} \vee c & C_{m+7} = \bar{a} \vee \bar{b} \vee c \\ C_{m+4} = a \vee \bar{b} \vee \bar{c} & C_{m+8} = \bar{a} \vee \bar{b} \vee \bar{c}. \end{array}$$

huit nouvelles clauses. Remarquons que pour toute interprétation des variables $\{a,b,c\}$, exactement sept des clauses C_i, $m+1 \leq i \leq m+8$, sont satisfaites. L'instance correspondante du problème $\{m'-1,m'\}$ EXACT 3SAT est donnée par la formule booléenne ϕ' définie par :

$$\phi' = \bigwedge_{1 \leq i \leq m+8} C_i$$

et $m' = m+8$. Il existe une interprétation satisfaisante f de la formule booléenne ϕ si et seulement si il existe une interprétation f' qui rende vraie au moins $m'-1 = m+7$ clauses de la formule booléenne ϕ'. $\boxed{4.4.7}$

PROPOSITION 4.4.8 ARRANGEMENT LINÉAIRE TRIANGULAIRE *est un problème* NP-*complet.*

PREUVE

L'appartenance du problème ARRANGEMENT LINÉAIRE TRIANGULAIRE à NP est triviale. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème $\{m-1,m\}$ EXACT 3SAT donnée par une formule booléenne ϕ sous forme normale conjonctive d'ordre 3. Nous notons $C = \{c_1,c_2,\ldots,c_m\}$ l'ensemble des clauses et $X = \{x_1,x_2,\ldots,x_n\}$ l'ensemble des variables booléennes. Soient V_C, V_Λ et A trois ensembles de sommets disjoints définis par :

$$\begin{aligned} V_C &= \{c_i \mid 1 \leq i \leq m\} \\ V_\Lambda &= \{\lambda_{i,j} \mid 1 \leq i \leq m \text{ et } 1 \leq j \leq 3\} \\ A &= \{a_i \mid 1 \leq i \leq 2m\}. \end{aligned}$$

L'ensemble V_C contient un sommet pour chaque clause, tandis que l'ensemble V_Λ contient un sommet pour chaque littéral[4]. Considérons maintenant les ensembles d'arêtes E_1, E_2, E_3 et E_4 définis par :

$$\begin{aligned} E_1 &= \{\{c_i,\lambda_{i,j}\} \mid 1 \leq i \leq m \text{ et } 1 \leq j \leq 3\} \\ E_2 &= \{\{\lambda_{i,j},\lambda_{i,k}\} \mid 1 \leq i \leq m \text{ et } 1 \leq i < k \leq 3\} \\ E_3 &= \{\{\lambda_{i,j},\lambda_{k,\ell}\} \mid 1 \leq i < k \leq m \text{ et } \lambda_{i,j} = \neg\lambda_{k,\ell}\} \\ E_4 &= \{\{c_i,c_j\} \mid 1 \leq i < j \leq m\}. \end{aligned}$$

L'instance correspondante du problème ARRANGEMENT LINÉAIRE TRIANGULAIRE nous est donnée par le graphe $G = (V,E)$ défini par :

$$\begin{cases} V = V_C \cup V_\Lambda \cup A \\ E = E_1 \cup E_2 \cup E_3 \cup E_4. \end{cases}$$

4. Par abus de notations, nous notons $\lambda_{i,j}$ à la fois les littéraux des clauses et les sommets de V_Λ.

Intuitivement, les ensembles E_1, E_2 et E_4 permettent de choisir exactement un littéral par clause, tandis que l'ensemble E_3 garantit une interprétation satisfaisante de la formule booléenne ϕ. Clairement, $|V| = 6m$ car les ensembles V_C, V_Λ et A sont disjoints. La construction peut s'effectuer en temps polynomial. Un exemple de construction est donné en figure 4.3

Supposons qu'il existe une interprétation f qui satisfasse au moins $m-1$ clauses de la formule booléenne ϕ. Sans perte de généralité, nous pouvons supposer que la clause c_m n'est pas nécessairement satisfaite par l'interprétation f. De même, en réordonnant éventuellement l'ordre des variables de chaque clause, nous pouvons supposer que chaque clause est satisfaite par son premier littéral, *i.e.* les sommets $\lambda_{i,1}$, $1 \leq i \leq m-1$, correspondent aux littéraux qui satisfont les $m-1$ premières clauses. Considérons une bijection σ de V dans $[\![6m]\!]$ définie par :

$$\begin{cases} 1 \leq \sigma(a_i) \leq 2m \quad \text{pour } a_i \in A \\ 2m+1 \leq \sigma(\lambda_{1,1}) \leq \sigma(\lambda_{2,1}) \leq \ldots \leq \sigma(\lambda_{m-1,1}) \leq 3m-1 \\ \sigma(c_m) = 3m \\ 3m+1 \leq \sigma(\lambda_{m,j}) \leq 3m+3 \quad \text{pour } 1 \leq j \leq 3 \\ 3m+4 \leq \sigma(c_{m-1}) \leq \sigma(c_{m-2}) \leq \ldots \leq \sigma(c_1) \leq 4m+2 \\ 4m+3 \leq \sigma(\lambda_{i,j}) \leq 6m \quad \text{pour } 1 \leq i \leq m-1 \text{ et } 2 \leq j \leq 3 \end{cases}$$

Montrons que la bijection σ satisfait $\sigma(u) + \sigma(v) > 6m$ pour tout $\{u,v\} \in E$. Soit $\{u,v\} \in E$ une arête quelconque du graphe G.

- Si $\{u,v\} \in E_1$, alors $\{u,v\}$ est de la forme $\{c_i,\lambda_{i,j}\}$ pour $1 \leq i \leq m$ et $1 \leq j \leq 3$. Considérons les deux cas suivants:
  - si $i \neq m$, alors $\sigma(c_i) + \sigma(\lambda_{i,j}) \geq (2m+k) + (4m+3-k) > 6m$ pour $1 \leq k \leq m-1$;
  - si $i = m$, alors $\sigma(c_m) + \sigma(\lambda_{m,j}) \geq 3m + (3m+1) > 6m$.
- Si $\{u,v\} \in E_2$, alors $\sigma(u) + \sigma(v) \geq (2m+1) + (4m+3) > 6m$.
- Si $\{u,v\} \in E_3$, alors $\sigma(u) + \sigma(v) \geq (2m+1) + (4m+3) > 6m$ car f est une interprétation satisfaisante de la formule booléenne ϕ.
- Si $\{u,v\} \in E_4$, alors $\sigma(u) + \sigma(v) \geq 3m + (3m+4) > 6m$.

Inversement, supposons qu'il existe une bijection σ de V dans $[\![6m]\!]$ telle que $\sigma(u) + \sigma(v) > 6m$ pour tout $\{u,v\} \in E$. Remarquons dans un premier temps que pour tout $u \in V$ et $v \in V$, si $1 \leq \sigma(u) < \sigma(v) \leq 3m$ alors $\{u,v\} \notin E$. En effet, $\sigma(u) + \sigma(v) \leq 3m + (3m-1) \not> 6m$. Autrement dit, pour tout $u \in V$, si $\sigma(u) \leq 3m$, alors $\sigma(v) > 3m$ pour tout $v \in N(u)$, *i.e.* le sous-ensemble $V' = \{u \mid u \in V \text{ et } \sigma(u) \leq 3m\}$ est un ensemble stable dans G. En conséquence, pour tout i, $1 \leq i \leq m$, il existe au plus un sommet $\lambda_{i,j} \in V_\Lambda$, $1 \leq j \leq 3$, tel que $\sigma(\lambda_{i,j}) \leq 3m$. Il suffit pour cela de remarquer que les sommets $\lambda_{i,j}$, $1 \leq j \leq 3$, induisent un graphe complet d'ordre 3. De même, il existe au plus un sommet c_i, $1 \leq i \leq m$ tel que $\sigma(c_i) \leq 3m$. Il suffit cette fois-ci de remarquer que les sommets de V_C induisent un graphe complet d'ordre m. Alors, nous avons :

$$\text{card}\left(V_\Lambda \cap \{\lambda_{i,j} \mid 3m+1 \leq \sigma(\lambda_{i,j}) \leq 6m\}\right) \geq 2m$$

et

$$\text{card}\left(V_C \cap \{c_i \mid 3m+1 \leq \sigma(c_i) \leq 6m\}\right) \geq m-1.$$

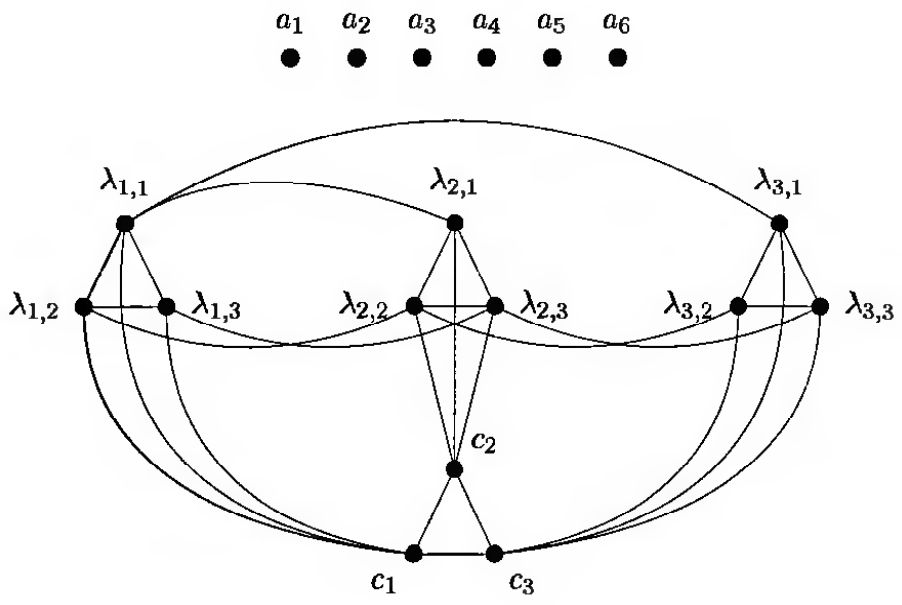


FIG. 4.3 – *Illustration de la construction utilisée dans la preuve de la proposition 4.4.8 pour la formule booléenne* $\phi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_2 \vee x_3)$. *Les sommets* $\lambda_{1,1}$, $\lambda_{1,2}$ *et* $\lambda_{1,3}$ *représentent la clause* $\bar{x}_1 \vee x_2 \vee x_3$, *les sommets* $\lambda_{2,1}$, $\lambda_{2,2}$ *et* $\lambda_{2,3}$ *représentent la clause* $x_1 \vee \bar{x}_2 \vee \bar{x}_3$, *et les sommets* $\lambda_{3,1}$, $\lambda_{3,2}$ *et* $\lambda_{3,3}$ *représentent la clause* $x_1 \vee x_2 \vee x_3$.

Par suite,

$$\text{card}\left(V_\Lambda \cap \{\lambda_{i,j} \mid 1 \leq \sigma(\lambda_{i,j}) \leq 3m\}\right) \geq m - 1.$$

Montrons maintenant que les sommets $\lambda_{i,j} \in V_\Lambda$ tels que $1 \leq \sigma(\lambda_{i,j}) \leq 3m$ définissent une interprétation de la formule booléenne ϕ qui satisfait au moins $m - 1$ clauses. En effet, pour tout $\lambda_{i,j} \in V_\Lambda$ et $\lambda_{k,\ell} \in V_\Lambda$, si $1 \leq \sigma(\lambda_{i,j}) < \sigma(\lambda_{k,\ell}) \leq 3m$, alors $\lambda_{i,j} \neq \neg\lambda_{k,\ell}$ (il suffit ici de remarquer que $\{\lambda_{i,j},\lambda_{k,\ell}\} \in E_3$ si $\lambda_{i,j} = \neg\lambda_{k,\ell}$). $\boxed{4.4.8}$

REMARQUE 4.4.9 Le problème ARRANGEMENT LINÉAIRE TRIANGULAIRE reste NP-complet même si nous remplaçons le graphe non orienté par un graphe orienté. Rappelons qu'un graphe orienté $D = (V,E)$ consiste en un ensemble de sommets V et un ensemble *ordonné* de paires de sommets appelées *arcs*. ARRANGEMENT LINÉAIRE TRIANGULAIRE peut être transformé en sa version orientée ARRANGEMENT LINÉAIRE TRIANGULAIRE ORIENTÉ en remplaçant simplement chaque arête $\{u,v\}$ du graphe non orienté par les deux arcs (u,v) et (v,u). Par essence, la version non orientée (ARRANGEMENT LINÉAIRE TRIANGULAIRE) n'est finalement qu'un cas particulier du cas orienté. ◇

Nous avons déjà vu qu'associé au *tri topologique* d'un graphe orienté $G = (V,E)$, correspond le problème de décision suivant: Soit A une (0,1)-matrice d'ordre n, existe-t-il une matrice de permutation P d'ordre n telle que $PAP^T \leq \triangle_n$? Dans les problèmes *tri topologique* et *arrangement linéaire triangulaire*, seule l'orientation

de la matrice triangulaire est modifiée. Cependant, la proposition 4.4.8 nous montre que cette différence d'orientation modifie radicalement la complexité (si P $\neq$ NP).

Remarquons cependant que la proposition 4.4.8 ne nous permet en aucun cas de conclure que MATRICE INF-TRIANGULAIRE soit également un problème NP-complet. En effet, dans la proposition 4.4.8, la matrice d'adjacence du graphe G n'est pas une (0,1)-matrice d'ordre $2n$ de la forme :

$$A = \begin{bmatrix} \mathbf{0}_{n,n} & B \\ B^T & \mathbf{1}_{n,n} \end{bmatrix}.$$

Or, la proposition 4.4.4 impose cette forme particulière. Sous cette hypothèse, si $RAR^T \leq \triangle_{2n}$ pour une matrice de permutation R d'ordre $2n$, alors la matrice de permutation R est nécessairement de la forme :

$$R = \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix}$$

où P et Q sont deux matrices de permutation d'ordre n.

Remarquons maintenant qu'aux boucles près, si A est de la forme

$$A = \begin{bmatrix} \mathbf{0}_{n,n} & B \\ B^T & \mathbf{1}_{n,n} \end{bmatrix}$$

alors A est la matrice d'adjacence d'un graphe de séparation. Un graphe $G = (V,E)$ est un *graphe de séparation* s'il existe une partition $V = S \cup K$ des sommets telle que S est un ensemble stable et K est une clique. Remarquons qu'il n'y a aucune restriction sur les arêtes entre les sous-ensembles S et K. En général, la partition $V = S \cup K$ d'un graphe de séparation n'est pas unique. En effet, S (resp. K) n'est pas nécessairement un ensemble stable de cardinalité maximale (resp. une clique de cardinalité maximale). Un exemple est donné en figure 4.4. Une caractérisation plus complète des graphes de séparation est donnée dans [Gol80] en termes de sous-graphes induits interdits. F.R. McMorris et D.R. Shier [MS83] présentent une caractérisation des graphes de séparation en tant que graphes d'intersection.

PROPOSITION 4.4.10 *Si* MATRICE INF-TRIANGULAIRE *est un problème* NP*-complet, alors* ARRANGEMENT LINÉAIRE TRIANGULAIRE *restreint à la classe des graphes de séparation est un problème* NP*-complet.*

En rappelant qu'un graphe de séparation est un graphe triangulé, nous obtenons un résultat plus fort.

PROPOSITION 4.4.11 *Si* MATRICE INF-TRIANGULAIRE *est un problème* NP*-complet, alors* ARRANGEMENT LINÉAIRE TRIANGULAIRE *restreint à la classe des graphes triangulés est un problème* NP*-complet.*

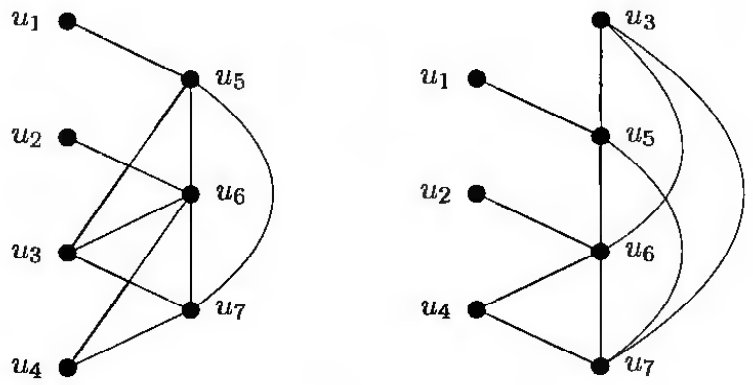


FIG. 4.4 – *Un graphe de séparation pour les partitions* $V = S_1 \cup K_1$ *avec* $S_1 = \{u_1,u_2,u_3,u_4\}$ *et* $K_1 = \{u_5,u_6,u_7\}$, *et* $V = S_2 \cup V_2$ *avec* $S_2 = \{u_1,u_2,u_4\}$ *et* $K_2 = \{u_3,u_5,u_6,u_7\}$.

4.4.3 Arrangement linéaire triangulaire et restrictions

Il est souvent intéressant en pratique de montrer qu'un problème est NP-complet sous certaines restrictions. Ces restrictions concernent par exemple la planarité, la connexité, le degré maximum des sommets, ..., ou concernent une classe de graphes. À ce titre, nous allons montrer que le problème ARRANGEMENT LINÉAIRE TRIANGULAIRE est NP-complet même si G est un graphe planaire avec boucles et $\Delta(G) \leq 3$ (rappelons que $\Delta(G)$ désigne le degré maximum d'un sommet du graphe G).

Un *ensemble stable* dans un graphe $G = (V,E)$ est un sous-ensemble $V' \subseteq V$ tel que pour tout $u \in V'$ et $v \in V'$, nous avons $\{u,v\} \notin E$. Autrement dit, un ensemble stable est un ensemble de sommets qui ne sont pas connectés deux à deux. ***ensemble stable***

THÉORÈME 4.4.12 ([KAR72]) ENSEMBLE STABLE *est un problème* NP-*complet.*

Le problème ENSEMBLE STABLE reste NP-complet pour les graphes planaires cubiques [GJS76]. Cependant, le problème est résoluble en temps polynomial si G est un graphe biparti[5] ou si $\Delta(G) \leq 2$ [GJ79]. Bien qu'employé fréquemment pour montrer qu'un problème est NP-complet, le problème ENSEMBLE STABLE n'est qu'une version différente des problèmes COUVERTURE DES SOMMETS et CLIQUE.

PROPOSITION 4.4.13 ARRANGEMENT LINÉAIRE TRIANGULAIRE *est un problème* NP-*complet même si G est un graphe planaire cubique avec boucles et* $\Delta(G) \leq 3$.

PREUVE

Soit une instance arbitraire du problème ENSEMBLE STABLE restreint à la classe des graphes planaires cubiques donnée par un graphe $G = (V,E)$ d'ordre n et par un

5. König a prouvé en 1931 que la cardinalité maximale d'un couplage dans un graphe biparti $G = (V,E)$ est égale à la cardinalité minimale d'une couverture des sommets. Or, si $V' \subseteq V$ est une couverture des sommets de cardinalité minimale, alors $V \setminus V'$ est un ensemble stable de cardinalité maximale.

entier positif $K \leq n$. Soient $G_a = (V_a,E_a)$, $G_b = (V_b,E_b)$ et $G_c = (V_c,E_c)$ les graphes définis par :

- $V_a = \{a_1,a_2,\ldots,a_n\}$;
- $E_a = \{\{a_i,a_j\} \mid \{u_i,u_j \in E\}$;
- $V_b = \{b_1,b_2,\ldots,b_K\}$;
- $E_b = \{\{b_i,b_i\} \mid 1 \leq i \leq K\}$;
- $V_c = \{c_1,c_2,\ldots,c_{n-K}\}$;
- $E_c = \emptyset$.

Autrement dit, G_a est une copie du graphe G, G_b est un graphe où les seules arêtes sont des boucles et G_c est un graphe complètement déconnecté. L'instance correspondante du problème ARRANGEMENT LINÉAIRE TRIANGULAIRE est donnée par le graphe $G^* = (V^*,E^*)$ d'ordre $m = 2n$ défini par $V^* = V_a \cup V_b \cup V_c$ et $E^* = E_a \cup E_b \cup E_c$. Remarquons que G^* est un graphe planaire avec boucles et $\Delta(G^*) \leq 3$. La construction du graphe G^* peut s'effectuer en temps polynomial.

Supposons qu'il existe un ensemble stable $V' \subseteq V(= V_a)$ de taille K dans G. Considérons la bijection σ de V^* dans $\{1,2,\ldots,2n\}$ définie par

$$\begin{cases} \forall\, u \in V_c, & 1 \leq \sigma(u) \leq n-K; \\ \forall\, u \in V', & n-K+1 \leq \sigma(u) \leq n; \\ \forall\, u \in V_b, & n+1 \leq \sigma(u) \leq n+K; \\ \forall\, u \in V_a \setminus V', & n+K+1 \leq \sigma(u) \leq 2n. \end{cases}$$

Si $\{u,v\} \in e^*$, alors $\{u,v\} \in E_a \cup E_b$ puisque G_c est un graphe complètement déconnecté. Plus précisément,

$$\{u,v\} \in E^* \quad \Rightarrow \quad \begin{cases} u,v \in V_b & \text{ou} \\ u,v \in V_a \setminus V' & \text{ou} \\ u \in V' \text{ et } v \in V_a \setminus V' & \end{cases}$$

D'où,

$$\begin{array}{lcl} u,v \in V_b & \Rightarrow & \sigma(u)+\sigma(v) \geq 2(n+1) > 2n \\ u,v \in V_a \setminus V' & \Rightarrow & \sigma(u)+\sigma(v) \geq 2(n+K+1) > 2n \\ u \in V' \text{ et } v \in V_a \setminus V' & \Rightarrow & \sigma(u)+\sigma(v) \geq n+K+1+n-K+1 > 2n \end{array}$$

et donc,

$$\forall\, \{u,v\} \in E^*, \quad \sigma(u)+\sigma(v) > 2n.$$

Inversement, supposons qu'il existe une bijection σ de V^* dans $\{1,2,\ldots,2n\}$ vérifiant $\sigma(u)+\sigma(v) > 2n$ pour tout $\{u,v\} \in E^*$. Remarquons dans un premier temps que, puisque G_b est un graphe où chaque sommet possède une boucle, alors $\sigma(b_i) > n$ pour $1 \leq i \leq K$. Autrement dit,

$$\sigma(V_b) \subset \{n+1,n+2,\ldots,2n\}.$$

Donc,

$$|\sigma(V^* \setminus V_b) \cap \{n+1,n+2,\ldots,2n\}| \leq n-K.$$

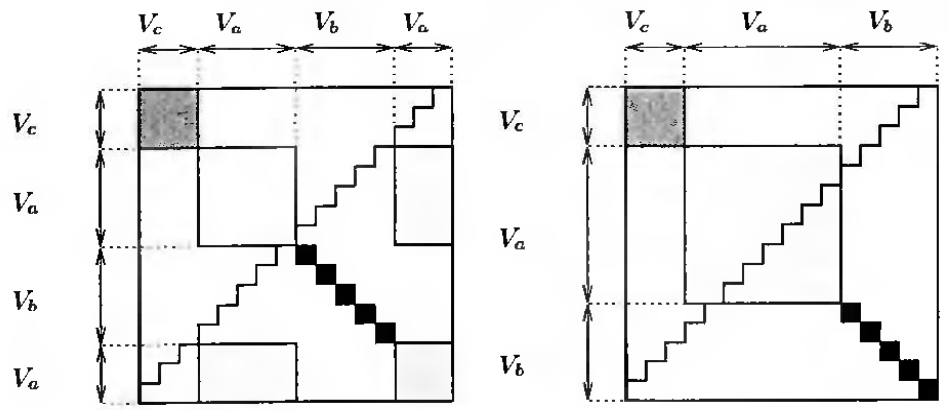


FIG. 4.5 – *Illustration de la preuve de la proposition 4.4.13. La configuration de droite n'est bien sûr pas unique.*

Or, $V_a \subset V^* \setminus V_b$, et par suite

$$|\sigma(V_a) \cap \{n+1, n+2, \ldots, 2n\}| \leq n - K.$$

En conséquence,

$$|\sigma(V_a) \cap \{1, 2, \ldots, n\}| \geq K$$

et il existe au moins K sommets de V_a qui ne sont pas connectés deux à deux. $\boxed{4.4.13}$

Une illustration de la preuve de la proposition 4.4.13 est donnée en figure 4.5. Les boucles sont représentées par des carrés de couleur sombre, tandis que l'ensemble stable est représenté par un carré grisé.

Remarquons que dans les propositions 4.4.13 et 4.4.8, le graphe construit n'est jamais connexe. Cependant, nous pouvons facilement montrer que le problème ARRANGEMENT LINÉAIRE TRIANGULAIRE reste NP-complet pour les graphes connexes. Il suffit en effet pour toute instance du problème ARRANGEMENT LINÉAIRE TRIANGULAIRE donnée par un graphe $G = (V,E)$ de considérer le nouveau graphe $G^* = (V^*,E^*)$ défini par :

- $V^* = V \cup \{x,y\}$
- $E^* = E \cup \{x,y\} \cup \{\{y,u\} \mid u \in V\}$.

où x et y sont deux nouveaux sommets (voir figure 4.6).

4.4.4 Le cas des graphes acycliques

Nous montrons dans cette sous-section que tout graphe acyclique admet un arrangement linéaire triangulaire et que cet arrangement peut être construit en temps polynomial.

Nous commençons par deux définitions. Soit $G = (V,E)$ un graphe. Une *feuille* est un sommet $u \in V$ tel que $d(u) = 1$. Deux feuilles $u,v \in V$, $u \neq v$, sont *indépendantes* si $(N(u) \cup \{u\}) \cap (N(v) \cup \{v\}) = \emptyset$. Autrement dit, deux feuilles u et v sont indépendantes si les arêtes incidentes à u et v sont disjointes. Cette définition est illustrée sur la figure 4.7.

feuille

indépendant

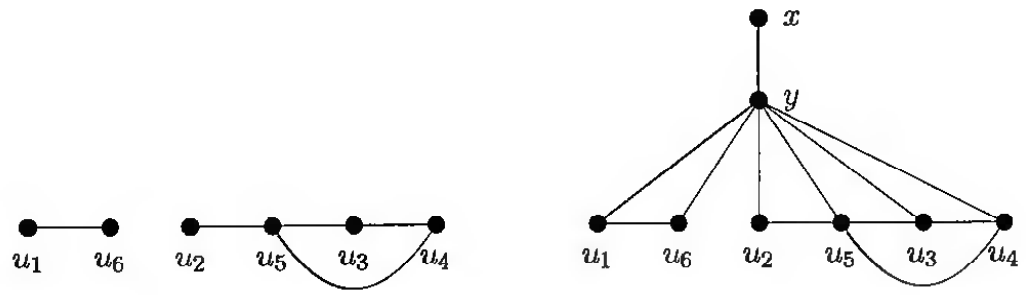


FIG. 4.6 – *Il existe un arrangement linéaire triangulaire du graphe de gauche si et seulement si il existe un arrangement linéaire triangulaire du graphe connexe de droite.*

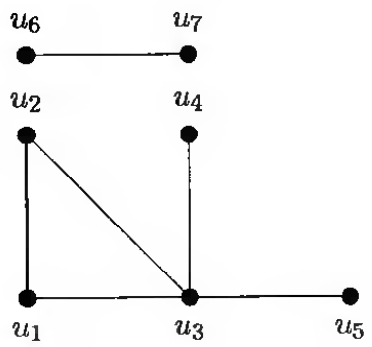


FIG. 4.7 – *Les feuilles du graphe sont u_4, u_5, u_6 et u_7. De plus, $\{u_4,u_6\}$, $\{u_4,u_7\}$, $\{u_5,u_6\}$ et $\{u_5,u_7\}$, sont des ensembles de feuilles indépendantes, tandis que $\{u_6,u_7\}$ et $\{u_4,u_5\}$ ne sont pas des ensembles de feuilles indépendantes.*

LEMME 4.4.14 *Soient $G = (V,E)$ un graphe d'ordre n et $a,b \in V$, $a \neq b$, deux sommets tels que $N(a) = \{b\}$. S'il existe un arrangement linéaire triangulaire σ de G, alors il existe un arrangement linéaire triangulaire σ' de G qui vérifie $\sigma'(a) = 1$ et $\sigma'(b) = n$.*

PREUVE

Soient π et π' les permutations de $[\![n]\!]$ définies par :

$$\pi(i) = \begin{cases} i+1 & \text{si } 1 \leq i < \sigma(a) \\ 1 & \text{si } i = \sigma(a) \\ i & \text{si } \sigma(a) < i \leq n \end{cases}$$

$$\pi'(i) = \begin{cases} i & \text{si } 1 \leq i < (\pi \circ \sigma)(b) \\ n & \text{si } i = (\pi \circ \sigma)(b) \\ i-1 & \text{si } (\pi \circ \sigma)(b) < i \leq n. \end{cases}$$

Considérons la bijection σ' de V dans $[\![n]\!]$ définie par $\sigma'(u) = (\pi' \circ \pi \circ \sigma)(u)$ pour tout $u \in V$. Montrons que σ' est également un arrangement linéaire triangulaire de G. Remarquons tout d'abord que $\sigma'(a) = 1$ et $\sigma'(b) = n$. Par conséquent, $\sigma'(a) + \sigma'(b) = n + 1 > n$. Soit $\{u,v\} \in E$ une arête quelconque avec $u,v \neq a,b$. Sans perte de généralité, supposons $\sigma(u) < \sigma(v)$. Remarquons que si $\sigma(u) < \sigma(v)$, alors $(\pi \circ \sigma)(u) < (\pi \circ \sigma)(v)$. Dans un premier temps, nous avons :

1. si $\sigma(u) < \sigma(a)$ et $\sigma(v) < \sigma(a)$, alors $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) = \sigma(u) + \sigma(v) + 2$;
2. si $\sigma(u) < \sigma(a)$ et $\sigma(v) > \sigma(a)$, alors $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) = \sigma(u) + \sigma(v) + 1$;
3. si $\sigma(u) > \sigma(a)$ et $\sigma(v) > \sigma(a)$, alors $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) = \sigma(u) + \sigma(v)$.

Aussi, nous avons $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) \geq \sigma(u) + \sigma(v)$. Envisageons maintenant les différents cas suivants.

1. Si $(\pi \circ \sigma)(u) < (\pi \circ \sigma)(b)$ et $(\pi \circ \sigma)(v) < (\pi \circ \sigma)(b)$, alors

$$\begin{aligned} \sigma'(u) + \sigma'(v) &= (\pi' \circ \pi \circ \sigma)(u) + (\pi' \circ \pi \circ \sigma)(v) \\ &= (\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) \\ &\geq \sigma(u) + \sigma(v) \\ &> n. \end{aligned}$$

2. Si $(\pi \circ \sigma)(u) < (\pi \circ \sigma)(b)$ et $(\pi \circ \sigma)(v) > (\pi \circ \sigma)(b)$, alors

$$\begin{aligned} \sigma'(u) + \sigma'(v) &= (\pi' \circ \pi \circ \sigma)(u) + (\pi' \circ \pi \circ \sigma)(v) \\ &= (\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) - 1. \end{aligned}$$

Si $\sigma(u) < \sigma(a)$, alors $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) \geq \sigma(u) + \sigma(v) + 1$. Par suite,

$$\begin{aligned} \sigma'(u) + \sigma'(v) &\geq \sigma(u) + \sigma(v) \\ &> n. \end{aligned}$$

Inversement, si $\sigma(u) > \sigma(a)$, alors $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) = \sigma(u) + \sigma(v)$. Considérons l'ordre relatif de a et b par la bijection σ. Si $\sigma(a) < \sigma(b)$, alors $\sigma(a) < \sigma(u) < \sigma(b) < \sigma(v)$ car $(\pi \circ \sigma)(u) = \sigma(u)$, $(\pi \circ \sigma)(b) = \sigma(b)$ et $(\pi \circ \sigma)(v) = \sigma(v)$. Aussi, $\sigma(u) + \sigma(v) > n + 2$ car $\sigma(a) + \sigma(b) > n$. D'où,

$$\begin{aligned}\sigma'(u) + \sigma'(v) &= \sigma(u) + \sigma(v) - 1\\ &> n + 1\\ &> n.\end{aligned}$$

Si $\sigma(a) > \sigma(b)$, alors $\sigma(b) < \sigma(a) < \sigma(u) < \sigma(v)$ car $\sigma(u) > \sigma(a)$. Aussi, $\sigma(u) + \sigma(v) > n + 4$ car $\sigma(a) + \sigma(b) > n$. D'où,

$$\begin{aligned}\sigma'(u) + \sigma'(v) &= \sigma(u) + \sigma(v) - 1\\ &> n + 3\\ &> n.\end{aligned}$$

3. Si $(\pi \circ \sigma)(u) > (\pi \circ \sigma)(b)$ et $(\pi \circ \sigma)(v) > (\pi \circ \sigma)(b)$, alors

$$\begin{aligned}\sigma'(u) + \sigma'(v) &= (\pi' \circ \pi \circ \sigma)(u) + (\pi' \circ \pi \circ \sigma)(v)\\ &= (\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) - 2.\end{aligned}$$

Si $\sigma(v) < \sigma(a)$, alors $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) = \sigma(u) + \sigma(v) + 2$. Par suite,

$$\begin{aligned}\sigma'(u) + \sigma'(v) &\geq \sigma(u) + \sigma(v)\\ &> n.\end{aligned}$$

Inversement, supposons $\sigma(v) > \sigma(a)$. Nous avons toujours $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) \geq \sigma(u) + \sigma(v)$. Considérons une nouvelle fois l'ordre relatif de a et b par σ. Si $\sigma(a) < \sigma(b)$, alors $\sigma(a) < \sigma(b) < \sigma(u) < \sigma(v)$ car $(\pi \circ \sigma)(u) = \sigma(u)$, $(\pi \circ \sigma)(b) = \sigma(b)$ et $(\pi \circ \sigma)(v) = \sigma(v)$. Aussi, $\sigma(u) + \sigma(v) > n + 4$ car $\sigma(a) + \sigma(b) > n$. D'où,

$$\begin{aligned}\sigma'(u) + \sigma'(v) &\geq \sigma(u) + \sigma(v) - 2\\ &> n + 2\\ &> n.\end{aligned}$$

Si $\sigma(a) > \sigma(b)$, alors $(\pi \circ \sigma)(b) = \sigma(b) + 1$. Aussi, $(\pi \circ \sigma)(u) > \sigma(b) + 1$ car $(\pi \circ \sigma)(u) > (\pi \circ \sigma)(b)$. De plus, $\sigma(v) + \sigma(b) > n + 1$ car $\sigma(a) + \sigma(b) > n$ et $\sigma(v) > \sigma(a)$. Aussi, $(\pi \circ \sigma)(u) + (\pi \circ \sigma)(v) > \sigma(b) + 1 + \sigma(v) > n + 2$. D'où,

$$\sigma'(u) + \sigma'(v) > n.$$

$\boxed{4.4.14}$

En d'autres termes, si G est un graphe muni d'un arrangement linéaire triangulaire σ et u est une feuille de G, alors nous pouvons toujours supposer que $\sigma(u) = 1$.

Cette propriété des arrangements linéaires triangulaires est à la base du lemme suivant.

LEMME 4.4.15 *Soient $G = (V,E)$ un graphe d'ordre n, et $a,b \in V$, $a \neq b$, deux sommets tels que $N(a) = \{b\}$. Il existe un arrangement linéaire triangulaire de G si et seulement si il existe un arrangement linéaire triangulaire du graphe $G|V'$ pour $V' = V \setminus \{a,b\}$.*

PREUVE

Notons $G' = (V',E')$ le sous-graphe de G induit par l'ensemble $V \setminus \{a,b\}$.

Supposons qu'il existe une bijection σ de V dans $[\![n]\!]$ telle que $\{u,v\} \in E \Rightarrow \sigma(u) + \sigma(v) > n$. En utilisant le lemme 4.4.14, nous pouvons supposer que $\sigma(a) = 1$ et $\sigma(b) = n$. La bijection σ' de V' dans $[\![n-2]\!]$ définie par $\sigma'(u) = \sigma(u) - 1$ pour tout $u \in V'$ est un arrangement linéaire triangulaire de G'. En effet, pour tout $\{u,v\} \in E'$, il vient

$$\begin{aligned}\sigma'(u) + \sigma'(v) &= \sigma(u) - 1 + \sigma(v) - 1\\ &> n - 2\end{aligned}$$

Inversement, supposons qu'il existe un arrangement linéaire triangulaire de G'. Alors, il existe une bijection σ' de V' dans $[\![n-2]\!]$ telle que si $\{u,v\} \in E'$, alors $\sigma'(u) + \sigma'(v) > n - 2$. Soit σ la bijection de V dans $[\![n]\!]$ définie par :

$$\forall u \in V, \quad \sigma(u) = \begin{cases} 1 & \text{si } u = a \\ \sigma'(u) + 1 & \text{si } u \neq a \text{ et } u \neq b \\ n & \text{si } u = b \end{cases}$$

Montrons que σ est un arrangement linéaire triangulaire de G. Soit $\{u,v\} \in E$. Alors,

- si $u \neq a$ et $v \neq b$, alors

$$\begin{aligned}\{u,v\} \in E &\Leftrightarrow \{u,v\} \in E'\\ &\Rightarrow \sigma'(u) + \sigma'(v) > n - 2\\ &\Leftrightarrow \sigma(u) + \sigma(v) > n\end{aligned}$$

- si $u = a$, alors $v = b$ et $\sigma(a) + \sigma(b) = 1 + n > n$
- si $u \neq a$ et $v = b$, alors $\sigma(u) + \sigma(b) \geq 1 + n > n$.

$\boxed{4.4.15}$

COROLLAIRE 4.4.16 *Soit G un graphe. Si G est un graphe acyclique, alors il existe un arrangement linéaire triangulaire de G.*

PREUVE

Si G est un graphe acyclique, alors il contient au moins une feuille ou il ne contient que des sommets isolés. Il suffit ensuite de remarquer que le sous-graphe induit par la suppression de deux sommets d'un graphe acyclique est toujours un graphe acyclique. $\boxed{4.4.16}$

4.5 Sous-matrice et couplage

4.5.1 Introduction

Nous étudions dans cette section la complexité de différents problèmes concernant la triangularisation d'une sous-matrice quelconque[6]. Le problème central est le calcul d'une plus grande sous-matrice triangularisable par permutation des lignes et des colonnes. Nous appelons SOUS-MATRICE INF-TRIANGULAIRE le problème de décision associée.

PROBLÈME 4.5.1 (SOUS-MATRICE INF-TRIANGULAIRE)
DONNÉES : Une (0,1)-matrice A d'ordre n et un entier positif K.
QUESTION : Existe-t-il une sous-matrice B de A d'ordre K et des matrices de permutation P et Q d'ordre K telles que $PBQ^T \leq \triangle_K$?

Nous appelons dans la suite SOUS-MATRICE EXACT-TRIANGULAIRE et SOUS-MATRICE SUP-TRIANGULAIRE les deux problèmes obtenus pour $PBQ^T = \triangle_K$ et $PBQ^T \geq \triangle_K$.

L'étude de la complexité de ce problème est motivée par la relation qui existe entre les problèmes MATRICE INF-TRIANGULAIRE et SOUS-MATRICE INF-TRIANGULAIRE. En effet, nous montrons dans un premier temps que le problème SOUS-MATRICE INF-TRIANGULAIRE est polynomialement réductible à notre problème initial MATRICE INF-TRIANGULAIRE. Autrement dit, si SOUS-MATRICE INF-TRIANGULAIRE est un problème NP-complet, alors MATRICE INF-TRIANGULAIRE est également un problème NP-complet. Nous travaillons essentiellement ensuite sur les graphes bipartis. Nous rappelons tout d'abord quelques résultats de complexité sur le calcul de couplages particuliers dans les graphes bipartis. Nous présentons ensuite les notions de sous-graphe induit inf-triangulaire et exact-triangulaire. Nous montrons que la recherche du plus grand sous-graphe induit exact-triangulaire dans un graphe biparti est un problème NP-complet. Nous n'avons cependant pas été en mesure de donner la complexité du problème concernant la recherche du plus grand sous-graphe induit inf-triangulaire dans un graphe biparti.

4.5.2 Réduction

Nous montrons ici le lien entre les problèmes SOUS-MATRICE INF-TRIANGULAIRE et MATRICE INF-TRIANGULAIRE. Le lemme suivant utilise la proposition 4.2.6.

LEMME 4.5.2 *Soient A une (0,1)-matrice d'ordre n, $\alpha \subseteq [\![n]\!]$ l'ensemble des indices des lignes de A qui ne contiennent que des 0 et $\beta \subseteq [\![n]\!]$ l'ensemble des indices des colonnes de A qui ne contiennent que des 0. S'il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$, alors il existe des matrices de permutation R et S d'ordre n calculables en temps polynomial (à partir des matrices de permutation P et Q) telles que :*

1. $RAS^T \leq \triangle_n$;

6. Le cas particulier des sous-matrices principales est entièrement traité dans la section suivante.

2. *les* $|\alpha|$ *premières lignes de* RAS^T *sont indicées par* α;
3. *les* $|\beta|$ *dernières colonnes de* RAS^T *sont indicées par* β.

PROPOSITION 4.5.3 *Si* SOUS-MATRICE INF-TRIANGULAIRE *est un problème* NP-*complet, alors* MATRICE INF-TRIANGULAIRE *est également un problème* NP-*complet.*

PREUVE

Les problèmes SOUS-MATRICE INF-TRIANGULAIRE et MATRICE INF-TRIANGULAIRE appartiennent à NP. Nous exhibons une réduction polynomiale du problème SOUS-MATRICE INF-TRIANGULAIRE au problème MATRICE INF-TRIANGULAIRE. Soit une instance arbitraire du problème SOUS-MATRICE INF-TRIANGULAIRE donnée par une (0,1)-matrice A d'ordre n et par un entier positif K. L'instance correspondante du problème MATRICE INF-TRIANGULAIRE est donnée par une (0,1)-matrice B d'ordre $2n-K$ défini par :

$$B = \begin{bmatrix} \mathbf{0}_{n-K,n} & \mathbf{0}_{n-K,n-K} \\ A & \mathbf{0}_{n,n-K} \end{bmatrix},$$

où $\mathbf{0}_{i,j}$ désigne une matrice de taille i par j qui ne contient que des 0. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une sous-matrice A' de A d'ordre K et des matrices de permutation P_1 et Q_1 d'ordre K telles que $P_1 A' Q_1^T \leq \llcorner_K$. Soient R et S les matrices de permutation d'ordre n telles que :

$$RAS^T = \begin{bmatrix} A_{11} & A' \\ A_{21} & A_{22} \end{bmatrix}.$$

Notons V et W les matrices de permutation d'ordre n définies par :

$$V = \begin{bmatrix} P_1 & \mathbf{0}_{K,n-K} \\ \mathbf{0}_{n-K,K} & \mathbb{I}_{n-K} \end{bmatrix} R \quad \text{et} \quad W^T = S^T \begin{bmatrix} \mathbb{I}_{n-K} & \mathbf{0}_{n-K,K} \\ \mathbf{0}_{K,n-K} & Q_1^T \end{bmatrix}$$

et P_2 et Q_2 les matrices de permutation d'ordre $2n-K$ définies par :

$$P_2 = \begin{bmatrix} \mathbb{I}_{n-K} & \mathbf{0}_{n-K,n} \\ \mathbf{0}_{n,n-K} & V \end{bmatrix} \quad \text{et} \quad Q_2^T = \begin{bmatrix} W^T & \mathbf{0}_{n,n-K} \\ \mathbf{0}_{n-K,n} & \mathbb{I}_{n-K} \end{bmatrix}.$$

Alors,

$$\begin{aligned} P_2 B Q_2^T &= \begin{bmatrix} \mathbb{I}_{n-K} & \mathbf{0}_{n-K,n} \\ \mathbf{0}_{n,n-K} & V \end{bmatrix} \begin{bmatrix} \mathbf{0}_{n-K,n} & \mathbf{0}_{n-K,n-K} \\ A & \mathbf{0}_{n,n-K} \end{bmatrix} \begin{bmatrix} W^T & \mathbf{0}_{n,n-K} \\ \mathbf{0}_{n-K,n} & \mathbb{I}_{n-K} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}_{n-K,n} & \mathbf{0}_{n-K,n-K} \\ VAW^T & \mathbf{0}_{n,n-K} \end{bmatrix}. \end{aligned}$$

Or,

$$\begin{aligned} VAW^T &= \begin{bmatrix} P_1 & 0_{K,n-K} \\ 0_{n-K,K} & \mathbb{I}_{n-K} \end{bmatrix} \begin{bmatrix} A_{11} & A' \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \mathbb{I}_{n-K} & 0_{n-K,K} \\ 0_{K,n-K} & Q_1^T \end{bmatrix} \\ &= \begin{bmatrix} P_1 A_{11} & P_1 A' Q_1^T \\ A_{21} & A_{22} Q_1^T \end{bmatrix} \\ &\leq \begin{bmatrix} P_1 A_{11} & \triangle_K \\ A_{21} & A_{22} Q_1^T \end{bmatrix} \end{aligned}$$

et par suite $P_2 B Q_2^T \leq \triangle_{2n-K}$.

Inversement, supposons qu'il existe des matrices de permutation P_2 et Q_2 d'ordre $2n-K$ telles que $P_2 B Q_2^T \leq \triangle_{2n-K}$. En utilisant le lemme 4.5.2, nous pouvons supposer que la matrice $P_2 B Q_2^T$ est de la forme :

$$P_2 B Q_2^T = \begin{bmatrix} 0_{n-K,n} & 0_{n-K,n-K} \\ VAW^T & 0_{n,n-K} \end{bmatrix}$$

où V et W sont deux matrices de permutation d'ordre n. Par suite,

$$VAW^T = \begin{bmatrix} A_{11} & A' \\ A_{21} & A_{22} \end{bmatrix}$$

avec $A' \leq \triangle_K$. $\boxed{4.5.3}$

4.5.3 Couplage dans les graphes

couplage
couplage maximale

Soit $G = (V,E)$ un graphe d'ordre n. Deux arêtes sont *incidentes* si elles ont un sommet extrémité en commun. Un *couplage* est un sous-ensemble d'arêtes $\mathcal{M} \subseteq E$ non incidentes deux à deux. Un *couplage de cardinalité maximale*[7] est un couplage $\mathcal{M}$ tel que $|\mathcal{M}| \geq |\mathcal{M}'|$ pour tout couplage $\mathcal{M}'$.

En 1973, J.E. Hopcroft et R.M. Karp[HK73] ont proposé un algorithme en temps $O(|V|^{\frac{1}{2}} \cdot |E|)$ pour calculer un couplage de cardinalité maximale dans un graphe biparti. J. Edmonds [Edm65] a présenté un algorithme en temps $O(|V|^3)$ dans le cas général[8]. Un algorithme en temps $O(\sqrt{|V|} \cdot |E|)$ est présenté dans [MV80].

couplage induit

Un *couplage induit* ou *couplage fort* dans un graphe est un sous-graphe induit composé d'arêtes disjointes : *i.e.* les composantes connexes sont des arêtes disjointes $e_1, e_2, \ldots, e_k$. Autrement dit, il n'existe aucune arête de G qui connecte un sommet de e_i avec un sommet de e_j pour $i \neq j$. La taille maximale d'un couplage induit dans un graphe G est notée $\mu_s(G)$. Nous associons au couplage induit le problème de décision suivant : Étant donné un graphe G et un entier positif K, est-ce que $\mu_s(G) \geq K$?

7. Il est malheureusement d'usage d'appeler un couplage de cardinalité maximale un couplage maximal. Or, un couplage maximal - un couplage qui ne peut pas être étendu par l'ajout d'une nouvelle arête - n'est pas nécessairement un couplage de cardinalité maximale.

8. La mise en œuvre de cet algorithme est une tâche assez difficile. Néanmoins, une analyse fine de la complexité et des structures de données est présentée dans [Jun99].

La recherche d'un couplage induit de cardinalité maximale dans un graphe est un problème NP-complet [Cam89, SV82, Lew96]. M.C. Golumbic et R.C. Lasket [GL93] ont proposé un schéma général pour trouver un couplage induit dans un graphe. L'idée consiste à transformer le graphe G en un graphe G^* et à rechercher un ensemble stable maximum dans ce dernier. Le graphe $G^* = (E,E^*)$ a pour sommets les arêtes du graphe G et les arêtes $e = \{u_1,u_2\}$ et $e' = \{u_3,u_4\}$ sont adjacentes dans G^* si et seulement si elles ne forment pas un couplage induit. Autrement dit, les arêtes $e = \{u_1,u_2\}$ et $e' = \{u_3,u_4\}$ sont adjacentes dans G^* si et seulement si le sous-graphe de G induit par l'ensemble $\{u_1,u_2,u_3,u_4\}$ est différent de $2K_2$. Il suffit ensuite de considérer des classes de graphes C pour lesquelles:

1. $G \in C \Rightarrow G^* \in C$;
2. la recherche d'un ensemble stable maximum s'effectue en temps polynomial pour la classe C.

Ainsi, il existe un algorithme en temps polynomial pour la recherche d'un couplage induit de cardinalité maximale si

- G est un graphe d'intervalle ou triangulé;
- G est un graphe trapézoïdal, de permutation ou à seuil;
- G est un arbre.

Remarquons que ces graphes sont parfaits [Gol80]. Ce n'est pas toujours le cas puisque la recherche d'un couplage induit de cardinalité maximale est un problème NP-complet si G est un graphe biparti de degré maximum 4 [SV82, Cam89].

Un couplage $\mathcal{M}$ dans un graphe $G = (V,E)$ est *unique restreint* s'il n'existe aucun autre couplage de même cardinalité sur les sommets couverts par le couplage $\mathcal{M}$. La taille maximale d'un couplage unique restreint dans un graphe G est notée $\mu_r(G)$. Clairement, $\mu_s(G) \leq \mu_r(G)$ puisqu'un couplage induit est un couplage unique restreint. Néanmoins, l'écart entre μ_s et μ_r peut être arbitrairement grand ou petit. En effet, nous avons par exemple: ***unique restreint***

- $\mu_s(K_n) = \mu_r(K_n) = 1$;
- $\mu_s(C_n) = \lfloor \frac{n}{3} \rfloor < \lceil \frac{n}{2} - 1 \rceil = \mu_r(G)$.

Caractériser les graphes pour lesquels $\mu_s(G) = \mu_r(G)$ est un problème ouvert [GHL]. La recherche d'un couplage unique restreint de cardinalité maximale est un problème NP-complet même si G est un graphe biparti avec $\Delta(G) \leq 7$ ou un graphe de séparation. Par suite, le problème est également NP-complet même si G est un graphe triangulé ou un graphe de comparaison[9] [GHL]. Néanmoins, le problème est résoluble en temps polynomial si G est un graphe à seuil ou un graphe d'intervalle propre.

4.5.4 Sous-graphe induit exact-triangulaire

Nous avons vu que la recherche d'un couplage induit de taille K et d'un couplage unique restreint de taille K dans un graphe biparti sont des problèmes NP-complets. Nous pouvons donner une interprétation matricielle de ces deux problèmes. En effet, soit $G = (V_1 \cup V_2,E)$, $|V_1| = m$ et $|V_2| = n$, un graphe biparti. Notons A sa matrice

9. Il suffit de remarquer qu'un graphe de séparation est un graphe triangulé et qu'un graphe biparti est un graphe de comparaison.

d'adjacence réduite (A est une (0,1)-matrice rectangulaire en général). Alors, il existe des matrices de permutation P d'ordre m et Q d'ordre n telles que :

$$PAQ^T = \begin{bmatrix} A' & * \\ * & * \end{bmatrix}$$

où A' est un matrice d'ordre K qui vérifie $A' = \mathbb{I}_K$ s'il existe un couplage induit de taille K et où $\mathbb{I}_K \leq A' \leq \llcorner_K$ s'il existe un couplage unique restreint de taille K.

DÉFINITION 4.5.4 (GRAPHE BIPARTI EXACT-TRIANGULAIRE) *Un graphe biparti* $G = (V_1 \cup V_2, E)$, $|V_1| = |V_2| = n$, *est un* graphe biparti exact-triangulaire *s'il existe des bijections* σ *de* V_1 *dans* $[\![n]\!]$ *et* τ *de* V_2 *dans* $[\![n]\!]$ *telles que* $\{u,v\} \in E$ *si et seulement si* $\sigma(u) \geq \tau(v)$ *pour tout* $u \in V_1$ *et* $v \in V_2$.

Nous appelons SOUS-GRAPHE INDUIT EXACT-TRIANGULAIRE le problème de décision associé à la recherche du plus grand sous-graphe induit exact-triangulaire dans un graphe biparti. Ce problème se formalise de la façon suivante :

> PROBLÈME 4.5.5 (SOUS-GRAPHE INDUIT EXACT-TRIANGULAIRE)
> DONNÉES : Un graphe biparti $G = (V_1 \cup V_2, E)$ et un entier positif K.
> QUESTION : Le graphe G contient-il un sous-graphe induit $G' = (V_1' \cup V_2', E')$ exact-triangulaire avec $|V_1'| = |V_2'| \geq K$?

Étant donnés une (0,1)-matrice A de taille m par n et un entier positif K, le problème est équivalent à la recherche d'une sous-matrice B d'ordre K pour laquelle il existe des matrices de permutation P et Q d'ordre K telles que $PBQ^T = \llcorner_K$.

PROPOSITION 4.5.6 SOUS-GRAPHE INDUIT EXACT-TRIANGULAIRE *est un problème* NP-*complet.*

PREUVE

Le problème SOUS-GRAPHE BIPARTI EXACT-TRIANGULAIRE appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème EXACT 3SAT donnée par une formule booléenne ϕ sous forme normale conjonctive d'ordre 3. Nous notons $C = \{c_1, c_2, \ldots, c_m\}$ l'ensemble des clauses et $X = \{x_1, x_2, \ldots, x_n\}$ l'ensemble des variables booléennes.

Soient X, L, C et Λ les ensembles de sommets disjoints définis par :

$$\begin{cases} X &= \{x_i \mid 1 \leq i \leq n\} \\ L &= \{v_i \mid 1 \leq i \leq n\} \cup \{f_i \mid 1 \leq i \leq n\} \end{cases}$$

$$\begin{cases} C &= \{c_i \mid 1 \leq i \leq m\} \\ \Lambda &= \{\lambda_{i,j} \mid 1 \leq i \leq m \ \wedge \ 1 \leq j \leq 3\}. \end{cases}$$

Intuitivement, les ensembles X et L sont associés aux variables booléennes, tandis que les ensembles C et Λ sont associés aux clauses. Plus précisément, l'ensemble X

contient un sommet pour chaque variable, l'ensemble L contient un sommet pour chaque littéral de chaque variable, l'ensemble C contient un sommet pour chaque clause et l'ensemble Λ contient un sommet pour chaque littéral de chaque clause. Soient a et b deux nouveaux sommets distincts qui n'appartiennent pas aux ensembles X, L, Λ et C. Ces deux sommets vont jouer un rôle de «*séparateur*» dans la suite. Plus précisément, le sommet a va séparer X d'une partie de Λ, tandis que le sommet b va séparer une partie de L de C.

Définissons les ensembles d'arêtes E_{XL}, E_a, $E_{\Lambda C}$, $E_{\Lambda b}$ et $E_{\Lambda L}$:

$$\begin{cases} E_{XL} &= \{\{x_i,v_k\},\{x_i,f_k\} \mid 1 \leq k \leq i \leq n\} \\ E_a &= \{\{a,b\}\} \cup \{\{a,v_i\},\{a,f_i\} \mid 1 \leq i \leq n\} \\ E_{\Lambda C} &= \{\{\lambda_{i,j},c_k\} \mid 1 \leq k \leq i \leq m \quad \text{et} \quad 1 \leq j \leq 3\} \\ E_{\Lambda b} &= \{\{\lambda_{i,j},b\} \mid 1 \leq i \leq m \quad \text{et} \quad 1 \leq j \leq 3\}. \end{cases}$$

Pour tout $1 \leq i \leq m$ et $1 \leq j \leq 3$, notons $ind(i,j)$ l'indice de la variable qui apparaît en position j dans la clause i. Autrement dit, $ind(i,j) = k$ si x_k ou $\bar{x}_k$ apparaît en position j dans la clause i. Nous pouvons maintenant définir l'ensemble d'arêtes $E_{\Lambda L}$. Pour tout $1 \leq i \leq m$ et $1 \leq j \leq 3$, l'ensemble $E_{\Lambda L}$ contient les arêtes $\{\lambda_{i,j},v_k\}$ et $\{\lambda_{i,j},f_k\}$ pour $1 \leq k \leq n$ et $k \neq ind(i,j)$. Notons $k' = ind(i,j)$, si $x_{k'}$ apparaît en position j de la clause i, alors l'ensemble $E_{\Lambda L}$ contient de plus l'arête $\{\lambda_{i,j},v_{k'}\}$, sinon il contient l'arête $\{\lambda_{i,j},f_{k'}\}$ ($\bar{x}_{k'}$ apparaît en position j dans la clause i). Remarquons alors que tout sommet $\lambda_{i,j} \in \Lambda$ est connecté à exactement $2n-1$ sommets de l'ensemble L.

L'instance correspondante du problème SOUS-GRAPHE BIPARTI EXACT-TRIANGULAIRE est donnée par le graphe $G = (V,E)$ définis par :

$$\begin{cases} V &= \{a,b\} \cup X \cup L \cup C \cup \Lambda \\ E &= E_{XL} \cup E_a \cup E_{\Lambda C} \cup E_{\Lambda b} \cup E_{\Lambda L} \end{cases}$$

et par un entier $K = n+m+1$. Le graphe G est biparti pour la bipartition $V = V_1 \cup V_2$ définie par :

$$\begin{cases} V_1 &= \{a\} \cup X \cup \Lambda \\ V_2 &= \{b\} \cup L \cup C. \end{cases}$$

La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une interprétation satisfaisante f de la formule booléenne ϕ. Sans perte de généralité, nous pouvons supposer que chaque clause est satisfaite par son premier littéral. Dans la suite, v/f_i, $1 \leq i \leq n$, désigne soit v_i soit f_i. Soient $V_1' \subseteq V_1$ et $V_2' \subseteq V_2$ les sous-ensembles de sommets définis par :

$$\begin{cases} V_1' &= \{a\} \cup \{x_i \mid 1 \leq i \leq n\} \cup \{\lambda_{i,1} \mid 1 \leq i \leq m\} \\ V_2' &= \{b\} \cup \{v/f_i \mid 1 \leq i \leq n \wedge v/f_i = v_i \text{ ssi } f(x_i) = \text{vrai}\} \cup \{c_i \mid 1 \leq i \leq m\}. \end{cases}$$

Nous vérifions facilement que $|V_1'| = |V_2'| = K = n+m+1$. De plus, pour tout $1 \leq i \leq n$, soit $v_i \in V_2'$, soit $f_i \in V_2'$. Considérons les bijections σ de V_1' dans $[\![K]\!]$ et

τ de V_2' dans $[\![K]\!]$ définies par :

$$\begin{cases} \sigma(x_i) = i & \text{pour } 1 \leq i \leq n \\ \sigma(a) = n+1 & \\ \sigma(\lambda_{i,1}) = n+i+1 & \text{pour } 1 \leq i \leq m \end{cases}$$

et

$$\begin{cases} \tau(v/f_i) = i & \text{pour } 1 \leq i \leq n \text{ et } v/f_i \in L \cap V_2' \\ \tau(b) = n+1 & \\ \tau(c_i) = n+i+1 & \text{pour } 1 \leq i \leq m. \end{cases}$$

Remarquons que les deux bijections utilisent l'ordre arbitraire donné sur les variables et les clauses. Montrons que le sous-graphe biparti induit par $V_1' \cup V_2'$ est exact-triangulaire.

Dans un premier temps, en remarquant que par construction :

$$\begin{cases} \{x_i, v/f_k\} \in E & \text{pour } 1 \leq k \leq i \leq n \text{ et } v/f_k = v_k \text{ ou } f_k \\ \{x_i, b\} \notin E & \text{pour } 1 \leq i \leq n \\ \{x_i, c_k\} \notin E & \text{pour } 1 \leq i \leq n \text{ et } 1 \leq k \leq m \end{cases}$$

il vient

$$\sigma(u) \geq \tau(v) \quad \text{si et seulement si} \quad \{u,v\} \in E \tag{4.1}$$

pour tout $u \in X \cap V_1' = X$ et $v \in V_2'$, car les bijections σ et τ respectent l'ordre arbitraire sur les variables et V_2' contient soit v_k, soit f_k pour $1 \leq k \leq n$. De plus, puisque le sommet a est connecté à tous les sommets de L et à b (mais pas aux sommets de C), alors

$$\sigma(u) \geq \tau(v) \quad \text{si et seulement si} \quad \{u,v\} \in E \tag{4.2}$$

pour tout $u \in X \cup \{a\}$ et $v \in V_2'$. Remarquons maintenant que :

$$\sigma(u) \geq \tau(v) \quad \text{si et seulement si} \quad \{u,v\} \in E \tag{4.3}$$

pour tout $u \in \Lambda \cap V_1'$ et $v \in C \cap V_2' = C$. En effet, Les bijections σ et τ utilisent l'ordre arbitraire sur les clauses et, pour tout $1 \leq i \leq m$, exactement un des sommets de $\{\lambda_{i,j} \mid 1 \leq j \leq 3\}$ appartient à V_1'. Enfin, puisque $L \cap V_2'$ correspond à une interprétation satisfaisante de la formule booléenne ϕ, il vient

$$\sigma(u) \geq \tau(v) \quad \text{si et seulement si} \quad \{u,v\} \in E \tag{4.4}$$

pour tout $u \in \Lambda \cap V_1'$ et $v \in V_2'$. Par suite (équations 4.2 et 4.4),

$$\sigma(u) \geq \tau(v) \quad \text{si et seulement si} \quad \{u,v\} \in E$$

pour tout $u \in V_1'$ et $v \in V_2'$. Un exemple de construction est donné en figure 4.8.

Inversement supposons qu'il existe dans le graphe biparti G un sous-graphe induit exact-triangulaire de taille K. Soit $G' = (V_1' \cup V_2', E')$ ce graphe. Alors, il existe des

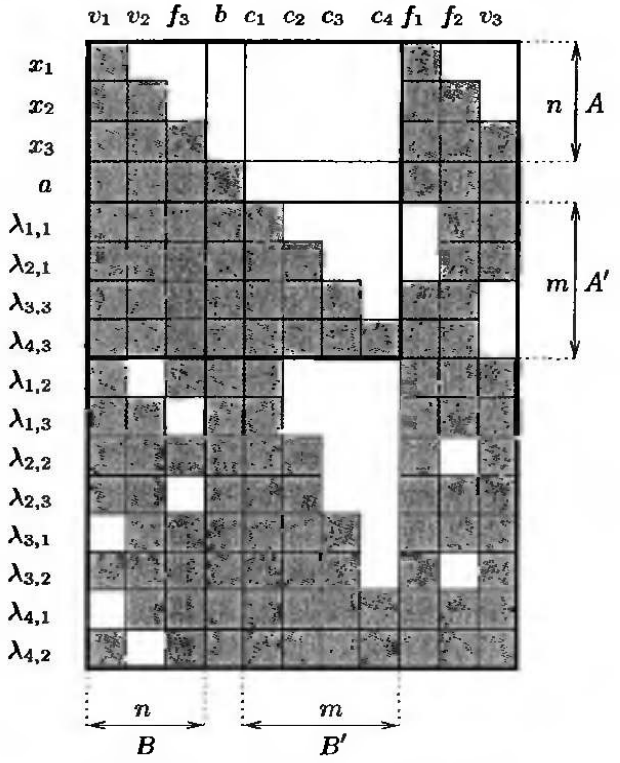


FIG. 4.8 – *Illustration de la construction utilisée dans la proposition 4.5.6 pour la formule booléenne* $\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$. *La figure est donnée pour l'interprétation satisfaisante* f *définie par* $f(x_1) =$ *vrai,* $f(x_2) =$ *vrai et* $f(x_3) =$ *faux. Remarquons que nous ne faisons pas ici l'hypothèse que chaque clause est satisfaite par son premier littéral : les deux premières clauses sont satisfaites par* x_1*, tandis que les deux dernières sont satisfaites par* $\bar{x}_3$*. Les cases grisées représentent des éléments non nuls.*

bijections σ de V_1' dans $[\![K]\!]$ et τ de V_2' dans $[\![K]\!]$ telles que $\{v_1,v_2\} \in E$ si et seulement si $\sigma(v_1) \geq \tau(v_2)$ pour tout $v_1 \in V_1'$ et $v_2 \in V_2'$. Notons A et A' les sous-ensembles de sommets définis par :

$$\begin{cases} A &= \{u \mid u \in V_1' \wedge 1 \leq \sigma(u) \leq n\} \\ A' &= \{u \mid u \in V_1' \wedge n+2 \leq \sigma(u) \leq n+m+1\} \end{cases}$$

et B et B' les sous-ensembles de sommets définis par :

$$\begin{cases} B &= \{u \mid u \in V_2' \wedge 1 \leq \tau(u) \leq n\} \\ B' &= \{u \mid u \in V_2' \wedge n+2 \leq \tau(u) \leq n+m+1\}. \end{cases}$$

Remarquons dans un premier temps que pour tout sommet $u \in A$, alors $|N(u)| \leq n+\sigma(u)$ car u possède exactement $\sigma(u)$ voisins dans E'. Or, $|N(\lambda_{i,j})| \geq 2n+2$ pour tout $\lambda_{i,j} \in V_1$ et $|N(a)| = 2n+1$. Par conséquent, $A \subseteq X$. Mais $|A| = |X|$. Aussi, nous avons en fait $A = X$. Par suite, $B \subseteq L$ car les sommets de X ne sont connectés qu'aux sommets de L. De plus, $L \cap B' = \emptyset$.

Supposons qu'il existe i, $1 \leq i \leq n$, tel que $v_i \in V_2'$ et $f_i \in V_2'$. Sans perte de généralité, supposons de plus que $\tau(v_i) < \tau(f_i)$. Alors, pour le sommet $x_k \in X$ tel que $\sigma(x_k) = \tau(v_i)$, nous avons $\{x_k,v_i\} \in E$ et $\{x_k,f_i\} \in E$. Ce qui constitue une contradiction puisque $\{x_k,f_i\} \in E'$ et $\sigma(x_k) < \tau(f_k)$. Aussi, par argument de cardinalité, nous avons :

$$\forall i, 1 \leq i \leq n, \quad \begin{cases} \text{soit } v_i \in V_2' \text{ et } f_i \notin V_2' \\ \text{soit } f_i \in V_2' \text{ et } v_i \notin V_2'. \end{cases} \tag{4.5}$$

Cette «*partition*» de L définit une interprétation f de l'ensemble des variables en posant :

$$f(x_i) = \text{vrai} \quad \text{si et seulement si} \quad v_i \in V_2'.$$

Montrons que f est une interprétation satisfaisante de la formule booléenne ϕ.

Tout d'abord, remarquons que $a \in V_1'$. En effet, si tel n'est pas le cas, soit $\lambda_{i,j}$ le plus petit littéral par $\sigma(\lambda_{i,j}) = n+1$. Alors, il est relié aux éléments de $L \cap V_2'$ (n éléments), à b et à c_i au moins, ce qui est une contradiction. Aussi, nous avons, $\sigma(a) = n+1$. De plus, $\tau(b) = n+1$ car le sommet a n'est connecté à aucune clause. Par conséquent, nous avons $A' \subset A$ et $B' = C$.

Supposons maintenant par l'absurde que deux sommets $\lambda_{i,j}$ et $\lambda_{i,k}$ pour $1 \leq i \leq m$ et $j \neq k$ soient dans V_1'. Alors, ils auraient même degré dans E' car dans le sous-graphe induit G', ils sont reliés aux mêmes clauses et au même nombre de sommets de L, ce qui constitue une contradiction. Ceci nous oblige à choisir un littéral et un seul par clause.

Considérons une clause $c_i = \lambda_{i,1} \vee \lambda_{i,2} \vee \lambda_{i,3}$. Dans V_1', nous avons choisit exactement un littéral, disons $\lambda_{i,1}$ pour simplifier. Soit $x_k = \lambda_{i,1}$. Alors, $f(\lambda_{i,1}) = \text{vrai}$ car dans E (et *a fortiori* dans E') $\lambda_{i,1}$ n'est pas relié à f_k. Par conséquent, puisque cette remarque s'applique à toutes le clauses, nous en concluons que f est une interprétation satisfaisante de la formule booléenne ϕ.

$\boxed{4.5.6}$

Nous avons déjà mentionné que le problème de décision associé à la recherche d'un couplage unique restreint de cardinalité maximale (problème COUPLAGE UNIQUE RESTREINT) est NP-complet [GHL]. Ce problème mesure la taille maximale d'un couplage unique $\mathcal{M}$, c'est-à-dire $|\mathcal{M}|$. À l'instar des problèmes BICLIQUE et BICLIQUE ARÊTES[10] nous pouvons nous interroger sur la complexité du problème COUPLAGE UNIQUE RESTREINT lorsque celui-ci ne mesure plus $|\mathcal{M}|$, mais le nombre d'arêtes du sous-graphe biparti induit par $\mathcal{M}$. Nous appelons COUPLAGE UNIQUE RESTREINT ARÊTES cette variante du problème.

PROBLÈME 4.5.7 (COUPLAGE UNIQUE RESTREINT ARÊTES)
DONNÉES : Un graphe biparti $G = (V_1 \cup V_2, E)$ et un entier positif K.
QUESTION : Existe-t-il un couplage unique restreint $\mathcal{M} \subseteq E$ tel que le sous-graphe biparti induit par le $\mathcal{M}$ comporte K arêtes, *i.e.* $G|\mathcal{M} = (V_1' \cup V_2', E')$ et $|E'| = K$?

Nous utilisons la prroposition précédente pour montrer que cette modification de l'énoncé ne simpleifie pas le problème.

COROLLAIRE 4.5.8 COUPLAGE UNIQUE RESTREINT ARÊTES *est un problème* NP-*complet.*

PREUVE

Le problème COUPLAGE UNIQUE RESTREINT ARÊTES appartient à NP. Montrons qu'il est NP-complet. Considérons de nouveau le graphe biparti $G = (V_1 \cup V_2, E)$ utilisé dans la preuve de la proposition 4.5.6. Nous allons montrer qu'il existe une interprétation satisfaisante f de la formule booléenne ϕ si et seulement si il existe un couplage unique restreint $\mathcal{M} \in E$ tel que le sous-graphe biparti induit par $\mathcal{M}$ contienne $\frac{1}{2}K(K+1)$ arêtes pour $K = n+m+1$; autrement dit, $G[\mathcal{M}] = (V_1' \cup V_2', E')$ et $|E'| = \frac{1}{2}K(K+1)$.

S'il existe une interprétation satisfaisante de la formule booléenne ϕ, nous avons montré qu'il existe un sous-graphe biparti exact-triangulaire de taille $K = m+n+1$. Ce sous-graphe induit un couplage unique restreint $\mathcal{M}$ qui comporte exactement $\frac{1}{2}K(K+1)$ arêtes.

Inversement, supposons qu'il existe un couplage unique $\mathcal{M}$ dans G tel que le sous-graphe induit par $\mathcal{M}$ comporte $\frac{1}{2}K(K+1)$ arêtes. Clairement, $|\mathcal{M}| \geq K$, avec égalité si et seulement si $\mathcal{M}$ induit un sous-graphe biparti exact-triangulaire. Il suffit alors de reprendre la preuve de la proposition 4.5.6 et de remarquer que tout couplage unique restreint vérifie $|\mathcal{M}| \leq K$. $\boxed{4.5.8}$

10. Soit $G = (V_1 \cup v_2, E)$ un graphe biparti. Les sous-ensembles $V_1' \subseteq V_1$ et $V_2' \subseteq V_2$ forment une *biclique* de taille $|V_1'| + |V_2'|$ si $\{u,v\} \in E$ pour tout $u \in V_1'$ et $v \in V_1'$. En utilisant des techniques de flots, il est possible de calculer la taille de la plus grande biclique dans un graphe biparti en temps polynomial. Néanmoins, le calcul de la biclique équilibrée ($|V_1'| = |V_2'|$) de taille maximale dans un graphe biparti est un problème NP-complet par une réduction depuis CLIQUE [GJ79] (référencé dans [GJ79] sous le nom SOUS-GRAPHE BIPARTI COMPLET ÉQUILIBRÉ). Une biclique est maximale pour les arêtes si le produit $|V_1| \times |V_2'|$ est maximum parmi toutes les bicliques. Le calcul de la biclique maximale pour les arêtes (problème BICLIQUE ARÊTES) est un problème NP-complet [Pee00].

4.5.5 Sous-graphe induit inf / sup-triangulaire

DÉFINITION 4.5.9 (GRAPHE BIPARTI INF / SUP-TRIANGULAIRE) *Un graphe biparti* $G = (V_1 \cup V_2, E)$, $|V_1| = |V_2| = n$, *est un* graphe biparti inf-triangulaire *(resp.* sup-triangulaire*) s'il existe des bijections* σ *de* V_1 *dans* $[\![n]\!]$ *et* τ *de* V_2 *dans* $[\![n]\!]$ *telles que, pour tout* $u \in V_1$ *et* $v \in V_2$, *si* $\sigma(u) \geq \tau(v)$ *alors* $\{u,v\} \in E$ *(resp. si* $\sigma(u) < \tau(v)$, *alors* $\{u,v\} \notin E$*).*

Nous appelons SOUS-GRAPHE INDUIT INF-TRIANGULAIRE (resp. SOUS-GRAPHE INDUIT SUP-TRIANGULAIRE) le problème de décision associé à la recherche du plus grand sous-graphe induit inf-triangulaire (resp. sup-triangulaire) dans un graphe biparti. Il est immédiat de vérifier que le problème SOUS-GRAPHE INDUIT INF TRIANGULAIRE (resp. SOUS-GRAPHE INDUIT SUP-TRIANGULAIRE) est équivalent au problème de la recherche de la plus grande sous-matrice inférieurement triangulaire (resp. supérieurement triangulaire) par permutation des lignes et des colonnes.

Rappelons, que pour toute (0,1)-matrice A, la matrice $\bar{A}$ est obtenue en remplaçant dans A les composantes 0 par 1 et 1 par 0.

PROPOSITION 4.5.10 *Soient* A *une* (0,1)*-matrice de taille* m *par* n *et* $K > 1$ *un entier. Il existe des matrices de permutation* P_1 *d'ordre* m *et* Q_1 *d'ordre* n *telles que :*

$$P_1 A Q_1^T \leq \begin{bmatrix} \llcorner_K & \mathbf{1}_{K,n-K} \\ \mathbf{1}_{n-K,K} & \mathbf{1}_{n-K,n-K} \end{bmatrix}$$

si et seulement si il existe des matrices de permutation P_2 *d'ordre* m *et* Q_2 *d'ordre* n *telles que :*

$$P_2 \bar{A} Q_2^T \geq \begin{bmatrix} \llcorner_{K-1} & \mathbf{0}_{K-1,n-K+1} \\ \mathbf{0}_{n-K+1,K-1} & \mathbf{0}_{n-K+1,n-K+1} \end{bmatrix}.$$

Le proposition précédente se traduit immédiatement sur le graphe biparti asscoié à la (0,1)-matrice A.

COROLLAIRE 4.5.11 *Soit* G *un graphe biparti et* $K > 1$ *un entier. Il existe un sous-graphe induit inf-triangulaire de taille* K *si et seulement si il existe un sous-graphe induit sup-triangulaire de taille* $K - 1$.

COROLLAIRE 4.5.12 SOUS-GRAPHE INDUIT INF-TRIANGULAIRE *est un problème* NP*-complet si et seulement si* SOUS-GRAPHE INDUIT SUP-TRIANGULAIRE *est un problème* NP*-complet.*

Nous n'avons cependant pas été en mesure de déterminer la complexité du problème SOUS-GRAPHE INDUIT INF-TRIANGULAIRE[11] (ou bien, de façon équivalente, de déterminer la complexité du problème SOUS-GRAPHE INDUIT SUP-TRIANGULAIRE). Néanmoins, nous prouvons maintenant que la «*version pondérée*» de notre problème SOUS-GRAPHE INDUIT INF-TRIANGULAIRE est NP-complète.

11. Rappelons (proposition 4.5.3 page 87) que si le problème SOUS-GRAPHE INDUIT INF-TRIANGULAIRE est NP-complet, alors le problème MATRICE INF-TRIANGULAIRE est également NP-complet.

PROBLÈME 4.5.13 (SOUS-GRAPHE INDUIT INF-TRIANGULAIRE PONDÉRÉ)
DONNÉES : Un graphe biparti $B = (V_1 \cup V_2, E)$, un entier positif W et une fonction ω de E dans $\mathbb{Z}^+$.
QUESTION : Existe-t-il une sous-graphe induit $B' = (V_1' \cup V_2', E')$ inf-triangulaire tel que $\sum_{e \in E'} \omega(e) \geq W$?

PROPOSITION 4.5.14 SOUS-GRAPHE INDUIT INF-TRIANGULAIRE PONDÉRÉ *est un problème* NP*-complet.*

PREUVE

Le problème SOUS-GRAPHE INDUIT INF-TRIANGULAIRE PONDÉRÉ appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème CLIQUE donnée par un graphe $G = (V,E)$ d'ordre n et par un entier positif K. Par souci de clarté, nous notons $f(i) = \frac{1}{2}i(i-1)$. De plus, pour tout $u \in V$ et $e \in E$, nous notons $u \in e$ si u est un sommet extrémité de l'arête e. Soient X et Y deux ensembles de sommets disjoints définis par :

$$X = \{x_1, x_2, \ldots, x_{n-K+1}\}$$
$$Y = \{y_1, y_2, \ldots, y_{f(K)+1}\}$$

et F_1 et F_2 les ensemble d'arêtes définis par :

$$F_1 = \{\{x_i, y_j\} \mid 1 \leq i \leq n-K+1 \text{ et } 1 \leq j \leq f(K)+1\}$$
$$F_2 = \{\{e,u\} \mid e \in E, u \in V \text{ et } u \in e\}.$$

L'instance correspondante du problème SOUS-GRAPHE INDUIT INF-TRIANGULAIRE PONDÉRÉ est donnée par un graphe biparti $B = (V_1 \cup V_2, F)$, un entier positif W et une fonction ω de F dans $\mathbb{Z}^+$ définis par :

$$\begin{cases} V_1 &= E \cup X \\ V_2 &= V \cup Y \\ F &= F_1 \cup F_2 \\ W &= (f(n)+1)(f(K)+1)(n-K+1) \end{cases}$$

et

$$\forall e \in F, \quad \omega(e) = \begin{cases} f(n)+1 & \text{si } e \in F_1 \\ 1 & \text{si } e \in F_2. \end{cases}$$

Clairement, nous avons $|V_1| = |E| + n - K + 1$ et $|V_2| = n + f(K) + 1$ car les ensembles V, E, X et Y sont disjoints deux à deux. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une clique $V' \subseteq V$ de taille K dans le graphe G. Soient $V_1' \subseteq V_1$ et $V_2' \subseteq V_2$ les sous-ensembles définis par :

$$V_1' = X \cup \{e \mid e = \{u_i, u_j\} \in E, u_i \in V' \text{ et } u_j \in V'\}$$
$$V_2' = Y \cup \{u_i \mid u_i \in V \setminus V'\}.$$

Remarquons dans un premier temps que, puisque V' induit une clique, il n'existe aucune transition entre les sommets de $\{e \mid e = \{u_i,u_j\} \in E, u_i \in V' \text{ et } u_j \in V\}$ et ceux de $\{u_i \mid u_i \in V \setminus V'\}$. De plus, nous avons $|V_1'| = |V_2'| = f(K) + n - K + 1$. Notons $B' = (V_1' \cup V_2', F')$ le sous-graphe induit par les sous-ensembles V_1' et V_2'. Ce sous-graphe biparti est inf-triangulaire. En effet, il suffit pour cela de considérer les bijections σ de V_1' dans $[\![f(K) + n - K + 1]\!]$ et τ de V_2' dans $[\![f(K) + n - K + 1]\!]$ définies par :

$$\begin{aligned} &\forall e \in V_1' \cap E, && 1 \leq \sigma(e) \leq f(K) \\ &\forall x_i \in X, && f(K) + 1 \leq \sigma(x_i) \leq f(K) + n - K + 1 \end{aligned}$$

et

$$\begin{aligned} &\forall y_i \in Y, && 1 \leq \tau(y_i) \leq f(K) + 1 \\ &\forall u \in V_2' \cap V, && f(K) + 2 \leq \tau(u) \leq f(K) + n - K + 1 \end{aligned}$$

et de vérifier que pour tout $u \in V_1'$ et $v \in V_2'$, si $\{u,v\} \in F'$, alors $\sigma(u) \geq \tau(v)$. Enfin, nous avons :

$$\sum_{e \in F'} \omega(e) \geq W$$

car $X \subseteq V_1'$ et $Y \subseteq V_2'$.

Inversement, supposons qu'il existe un sous-graphe induit $B' = (V_1' \cup V_2', F')$ inf-triangulaire avec $\sum_{e \in F'} \omega(e) \geq W$. Alors, en notant $|V_1'| = |V_2'| = m$ pour simplifier, il existe des bijections σ de V_1' dans $[\![m]\!]$ et τ de V_2' dans $[\![m]\!]$ telles que si $\{u,v\} \in F'$, alors $\sigma(u) \geq \tau(v)$. Notons m_1 (resp. m_2) le nombre d'arêtes de F_1 (resp. F_2) présentes dans F', *i.e.* $m_1 = |F_1 \cap F'|$ et $m_2 = |F_2 \cap F'|$. Puisque $\omega(e) = f(n) + 1$ pour tout $e \in F_1$ et $\omega(e) = 1$ pour tout $e \in F_2$, il vient :

$$\begin{aligned} m_1(f(n) + 1) + m_2 &\geq W \\ &= (f(n) + 1)(f(K) + 1)(n - K + 1). \end{aligned}$$

Alors,

$$m_2 \geq (f(n) + 1)((f(K) + 1)(n - K + 1) - m_1).$$

Or, $m_2 \leq f(n) < f(n)+1$ avec $m_2 = f(n)$ si et seulement si $G = K_n$. Par conséquent,

$$(f(n) + 1) > (f(n) + 1)((f(K) + 1)(n - K + 1) - m_1).$$

D'où,

$$m_1 \geq (f(K) + 1)(n - K + 1)$$

car m_1 est un entier positif ou nul. Mais, $m_1 \leq (f(K) + 1)(n - K + 1)$ car $|X| = n - K + 1$ et $|Y| = f(K) + 1$. En conséquence, nous avons $m_1 = (f(K) + 1)(n - K + 1)$. Autrement dit, $X \subseteq V_1'$ et $Y \subseteq V_2'$. Mais, $\sigma(x_i) \geq f(K) + 1$ pour tout $x_i \in X$ car $d(x_i) = f(K) + 1$ dans B'. De même, $\tau(y_i) \leq f(K) + 1$ pour tout $y_i \in Y$ car

$d(y_i) = n - K + 1$ dans B'. Par conséquent, $|V_1'| = |V_2'| \geq f(K) + n - K + 1$. Alors, il existe un sous-ensemble $E' \subseteq E$, $|E'| = f(K)$, tel que $\sigma(e_i) \leq f(K)$ pour tout $e_i \in E$. De même, il existe un sous-ensemble $V' \subseteq V$, $|V'| = n - K$, tel que $\tau(u_i) \geq f(K) + 2$ pour tout $u_i \in V'$. Mais, puisque le sous-graphe induit B' est inf-triangulaire, il vient :

$$\forall e_i \in E', \forall u_j \in V', \quad \{e_i, u_j\} \notin F$$

car $\sigma(e_i) \leq f(K) < f(K) + 2 \leq \tau(u_j)$. Aussi, il existe $f(K)$ sommets de E qui ne sont pas adjacents à $n - K$ sommets de V. Mais, par construction du graphe biparti B, un sommet $e_i \in E$ est adjacent à un sommet $u_j \in V$ si et seulement si $u_j \in e_i$, *i.e.* le sommet u_j est un sommet extrémité de l'arête e_i dans G. Par conséquent, il existe $f(K)$ arêtes de G incidentes à au plus K sommets de G. Puisque $f(K) = \frac{1}{2}K(K-1)$, nous en concluons qu'il existe une clique de taille K dans le graphe G. $\boxed{4.5.14}$

4.6 Sous-matrice principale

4.6.1 Introduction

Nous avons étudié dans la section précédente le problème de la triangularisation d'une sous-matrice par permutation des lignes et des colonnes. Nous allons nous intéresser dans cette sous-section au cas particulier où la sous-matrice recherchée est une sous-matrice principale. Rappelons qu'une sous-matrice d'ordre m d'une matrice A d'ordre n est *principale* si elle peut être obtenue en supprimant $n - m$ lignes et $n - m$ colonnes de A, et si les indices de lignes et de colonnes supprimées sont identiques. De façon équivalente, une sous-matrice est principale si elle peut être placée dans le coin supérieur gauche de A par une même permutation des lignes et des colonnes.

sous-matrice principale

Nous considérons un ensemble de problèmes dont les noms génériques sont ISO SOUS-MATRICE INF-$\mathbb{T}$ et ISO SOUS-MATRICE EXACT-$\mathbb{T}$. Le paramètre $\mathbb{T}$ représente une matrice triangulaire : *i.e.* $\mathbb{T} = \mathbb{L}_K, \mathbb{A}_K, \mathbb{V}_K$ ou $\mathbb{V}_K$ pour un entier positif K donné.

PROBLÈME 4.6.1 (ISO SOUS-MATRICE INF-$\mathbb{T}$)
DONNÉES : Une (0,1)-matrice A d'ordre n, un entier positif K et une matrice triangulaire $\mathbb{T}$ d'ordre K.
QUESTION : Existe-t-il une matrice de permutation P d'ordre n telle que :

$$PAP^T \leq \begin{bmatrix} \mathbb{T} & \mathbf{1}_{K,n-K} \\ \mathbf{1}_{n-K,K} & \mathbf{1}_{n-K,n-K} \end{bmatrix}$$

PROBLÈME 4.6.2 (ISO SOUS-MATRICE EXACT-$\mathbb{T}$)
DONNÉES : Une (0,1)-matrice A d'ordre n un entier positif K et une matrice triangulaire $\mathbb{T}$ d'ordre K.
QUESTION : Existe-t-il une matrice de permutation P d'ordre n telle que :

$$PAP^T = \begin{bmatrix} \mathbb{T} & * \\ * & * \end{bmatrix}$$

Nous vérifions facilement qu'il suffit de distinguer deux classes de problèmes, à savoir :

1. $\mathbb{T} = \triangle_K$ ou $\triangledown_K$;
2. $\mathbb{T} = \triangle_K$ ou $\triangledown_K$.

Aussi, dans la suite, nous examinerons uniquement les cas $\mathbb{T} = \triangle_K$ et $\mathbb{T} = \triangle_K$. Nous montrerons dans la suite que tous ces problèmes (certains sont bien connus) sont NP-complets. Nous étudions dans un premier temps le cas $\mathbb{T} = \triangle_K$ (ou $\triangledown_K$), puis $\mathbb{T} = \triangle_K$ (ou $\triangledown_K$).

4.6.2 $\mathbb{T} = \triangle_K$ (ou $\triangledown_K$)

Dans le cas des sous-matrices principales, le problème ISO SOUS-MATRICE INF-$\triangle$ pour $\mathbb{T} = \triangle_K$ est fortement lié à l'acyclicité d'un graphe orienté. Rappelons dans un premier temps qu'un graphe orienté G d'ordre n est acyclique si et seulement si il existe une matrice de permutation P d'ordre n telle que $PAP^T \leq \triangle_n$, où A est la matrice d'adjacence de G. Le lemme suivant est immédiat.

LEMME 4.6.3 *Soient $G = (V,E)$ un graphe orienté d'ordre n, A la matrice d'adjacence de G et K un entier positif. Il existe un sous-ensemble $V' \subseteq V$, $|V'| = K$, tel que le sous-graphe induit par V' soit acyclique si et seulement si il existe une matrice de permutation P d'ordre n telle que :*

$$PAP^T \leq \begin{bmatrix} \triangle_K & \mathbf{1}_{n-K} \\ \mathbf{1}_{n-K,K} & \mathbf{1}_{n-K,n-K} \end{bmatrix}.$$

Nous pouvons décider si un graphe orienté $G = (V,E)$ est acyclique en temps $\Theta(|V| + |E|)$ en utilisant un tri topologique [CLR92]. Par contre, étant donnés un graphe orienté d'ordre n et un entier K, décider s'il existe un sous-graphe induit acyclique d'ordre K est un problème NP-complet en général [Kar72] [12].

COROLLAIRE 4.6.4 ISO SOUS-MATRICE INF-$\triangle$ *et* ISO SOUS-MATRICE INF-$\triangledown$ *sont des problèmes* NP*-complets.*

12. Le problème est en général référencé sous le nom FEEDBACK VERTEX SET ou MAXIMUM ACYCLIC SUBGRAPH PROBLEM [Kar72, GJ79, FPR99]. Dans sa définition originale, étant donné un graphe orienté $G = (V,E)$, il s'agit de déterminer un sous-ensemble de sommet $V' \subseteq V$ de cardinalité minimale tel que V' contienne au moins un sommet de tout cycle orienté dans G. Récemment, P.D. Seymour [Sey95] a proposé un algorithme d'approximation de ratio $O(\log n \log\log n)$ pour ce problème

Le problème ISO SOUS-MATRICE INF-$\triangle$ est trivial si $K = n$ et A est une (0,1)-matrice symétrique. Nous allons montrer que le problème ISO SOUS-MATRICE INF-$\triangle$ est NP-complet même si la matrice A est symétrique et si chaque ligne ou colonne contient au plus trois éléments non nuls. Nous utilisons pour cela une réduction depuis le problème RECOUVREMENT EXACT PAR 3-ENSEMBLES. Rappelons dans un premier temps l'énoncé de ce problème.

PROBLÈME 4.6.5 (RECOUVREMENT EXACT PAR 3-ENSEMBLES)
DONNÉES : Une famille $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ de sous-ensembles, $|S_i| = 3$ pour tout $S_i \in \mathcal{F}$, d'un $3q$-ensemble X.
QUESTION : Existe-t-il une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ telle que chaque élément de X apparaît dans exactement un sous-ensemble de $\mathcal{F}'$?

THÉORÈME 4.6.6 ([KAR72]) RECOUVREMENT EXACT PAR 3-ENSEMBLES *est un problème* NP-*complet.*

Le problème RECOUVREMENT EXACT PAR 3-ENSEMSLES est plus communément appelé x3c. Remarquons que puisque chaque sous-ensemble contient exactement trois éléments, alors $|\mathcal{F}'| = q$. Le problème reste NP-complet même si chaque élément de X apparaît dans au plus trois sous-ensembles distincts de la famille $\mathcal{F}$. Cependant, le problème est résoluble en temps polynomial si chaque élément de X apparaît dans au plus deux sous-ensembles distincts de $\mathcal{F}$ [GJ79]. La variante RECOUVREMENT EXACT PAR 2-ENSEMBLES est également résoluble en temps polynomial [GJ79]. x3c

PROPOSITION 4.6.7 ISO SOUS-MATRICE INF-$\triangle$ *est un problème* NP-*complet même si la matrice A est symétrique et chaque ligne ou colonne contient au plus trois éléments non nuls.*

PREUVE

Considérons une instance arbitraire du problème x3c donnée par une famille $\mathcal{F} = \{S_1,S_2,\ldots,S_m\}$ de sous-ensembles, $|S_i| = 3$ pour tout $S_i \in \mathcal{F}$, d'un $3q$-ensemble X. Sans perte de généralité, nous pouvons supposer que chaque élément de X apparaît dans au plus trois sous-ensembles de la famille $\mathcal{F}$. L'instance correspondante de ISO SOUS-MATRICE INF-$\triangle$ est donnée par une (0,1)-matrice symétrique $A = [a_{i,j}]$ d'ordre m définie par $a_{i,j} = 1$ si et seulement si $S_i \cap S_j \neq \emptyset$ pour tout $1 \leq i \leq m$ et $1 \leq j \leq m$, et par un entier positif $K = q$. Remarquons que $a_{i,i} = 1$ pour tout $1 \leq i \leq m$. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de q sous-ensembles disjoints. Soit $\{S_{i_1},S_{i_2},\ldots,S_{i_q}\}$ cet ensemble. Considérons la sous-matrice principale B de A correspondant aux lignes et aux colonnes indicées par l'ensemble $\{i_1,i_2,\ldots,i_q\}$. Autrement dit, B est la matrice d'intersection des sous-ensembles $\{S_{i_1},S_{i_2},\ldots,S_{i_q}\}$. Puisque les sous-ensembles de $\mathcal{F}'$ sont disjoints deux à deux, alors B est la matrice identité d'ordre $K = q$. Par suite, nous avons $B \leq \triangle_K$.

Inversement, supposons qu'il existe une sous-matrice principale $B = [b_{i,i}]$ de A d'ordre K telle que $B \leq \triangle_K$. Par construction, B est une (0,1)-matrice symétrique

et $b_{i,i} = 1$ pour tout $1 \leq i \leq K$. Alors, $B = \mathbb{I}_K$ est l'unique solution de l'équation $B \leq \triangle_K$. Par conséquent, il existe K sous-ensembles de $\mathcal{F}$ disjoints deux à deux. $\boxed{4.6.7}$

COROLLAIRE 4.6.8 ISO SOUS-MATRICE INF-$\triangledown$ *est un problème* NP*-complet même si la matrice A est symétrique et chaque ligne ou colonne contient au plus trois éléments non nuls.*

EXEMPLE 13 *Soit $\mathcal{F} = \{S_1,S_2,S_3,S_4,S_5,S_6\}$ une famille de sous-ensembles d'un $3q$-ensemble X pour $q = 4$ définie par :*

$$S_1 = \{x_1,x_5,x_9\} \qquad S_2 = \{x_2,x_6,x_{10}\} \qquad S_3 = \{x_2,x_6,x_{11}\}$$
$$S_4 = \{x_1,x_7,x_9\} \qquad S_5 = \{x_3,x_7,x_{11}\} \qquad S_6 = \{x_4,x_8,x_{12}\}$$

Remarquons que la sous-famille $\mathcal{F}' = \{S_1,S_2,S_5,S_6\}$ est composée de sous-ensembles disjoints. La (0,1)-matrice d'intersection est notée A et la matrice de permutation P place les éléments indicés par $\{1,2,5,6\}$ aux quatre premières positions :

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{et} \quad PAP^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

◇

À mi-chemin entre les deux problèmes SOUS-MATRICE INF-TRIANGULAIRE et ISO SOUS-MATRICE INF-$\triangle$, nous considérons le problème suivant : Étant donnée une (0,1)-matrice A d'ordre n, existe-t-il une sous-matrice principale B de A d'ordre K et des matrices de permutation P et Q d'ordre K telles que $PBQ^T \leq \triangle_K$? Autrement dit, à la différence du problème précédent, nous autorisons des permutations *différentes* pour les lignes et les colonnes de la sous-matrice principale. Nous allons montrer que ce problème est également NP-complet.

LEMME 4.6.9 *Soit $B = [b_{i,j}]$ une (0,1)-matrice symétrique d'ordre n telle que $b_{i,i} = 1$ pour tout $1 \leq i \leq n$. S'il existe des matrices de permutation P et Q d'ordre n telles que $PBQ^T \leq \triangle_n$, alors $B = \mathbb{I}_n$ et $P = Q$.*

PREUVE

Puisque $b_{i,i} = 1$ pour $1 \leq i \leq n$, alors $\mathrm{per}(B) > 0$. En utilisant le théorème 4.3.4, nous avons :

$$\mathbb{I}_n \leq PBQ^T \leq \triangle_n.$$

En notant $C = [c_{i,j}]$ la (0,1)-matrice définie par $C = PBP^T$, il vient :

$$\mathbb{I}_n \leq C(PQ^T) \leq \triangle_n.$$

Remarquons que C est également une (0,1)-matrice symétrique et $c_{i,i} = 1$ pour $1 \leq i \leq n$. Alors, la matrice C est de la forme $\mathbb{I}_n + D$ où $D = [d_{i,j}]$ est une (0,1)-matrice symétrique avec $d_{i,i} = 0$ pour $1 \leq i \leq n$. Par suite,

$$\mathbb{I}_n \leq \mathbb{I}_n PQ^T + DPQ^T \leq \triangle_n$$

Si $P \neq Q$, alors $\mathbb{I}_n PQ^T \not\leq \triangle_n$, ce qui est une contradiction. Donc nous avons $P = Q$ et par suite $\mathbb{I}_n \leq PBP^T \leq \triangle_n$. Mais, puisque PBP^T est une (0,1)-matrice symétrique, $B = \mathbb{I}_n$ est l'unique solution. $\boxed{4.6.9}$

PROPOSITION 4.6.10 *Étant donnés une* (0,1)*-matrice A d'ordre n et un entier positif K, décider s'il existe une sous-matrice principale B d'ordre K de A et les matrices de permutation P et Q d'ordre K telles que $PBQ^T \leq \triangle_K$ est un problème* NP*-complet.*

PREUVE

Lemme 4.6.9 et proposition 4.6.7. $\boxed{4.6.10}$

Nous avons déjà montré que la recherche de la plus grande sous-matrice exactement triangulaire est un problème NP-complet dans le cas général[13]. Nous montrons que le problème est tout aussi difficile si nous recherchons la plus grande sous-matrice *principale* exactement triangulaire relativement à $\mathbb{T} = \triangle_K$.

PROPOSITION 4.6.11 ISO SOUS-MATRICE EXACT-$\triangle$ *est un problème* NP*-complet.*

PREUVE

Le problème ISO SOUS-MATRICE EXACT-$\triangle$ appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème CLIQUE donnée par un graphe $G = (V,E)$ d'ordre n et par un entier positif K. Par souci de clarté, nous notons $V = \{u_1, u_2, \ldots, u_n\}$. L'instance correspondante du problème ISO SOUS-MATRICE EXACT-$\triangle$ est donnée par une (0,1)-matrice $A = [a_{i,j}]$ d'ordre n définie par $a_{i,j} = 1$ si $\{u_i, u_j\} \in E$ et $i > j$. De plus, $a_{i,i} = 1$ pour $1 \leq i \leq n$. L'entier positif K est le même que celui de l'instance du problème CLIQUE. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une clique de taille K dans le graphe G. Soit $V' = \{u_{i_1}, u_{i_2}, \ldots, u_{i_K}\}$ cet ensemble. Sans perte de généralité, nous pouvons supposer $i_1 < i_2 < \ldots < i_K$. Considérons la sous-matrice principale $B = [b_{i,j}]$ de A correspondant aux lignes et aux colonnes indicées par l'ensemble $\{i_1, i_2, \ldots, i_K\}$. Alors, $b_{i,j} = 1$ pour $1 \leq j \leq i \leq K$. Par conséquent, il existe une matrice de permutation P d'ordre n telle que :

$$PAP^T = \begin{bmatrix} \triangle_K & * \\ * & * \end{bmatrix}.$$

Inversement, supposons qu'il existe une matrice de permutation P d'ordre n telle que :

$$PAP^T = \begin{bmatrix} B & * \\ * & * \end{bmatrix}$$

13. proposition 4.5.6 page 90.

où $B = \triangle_K$. Puisque B est une sous-matrice principale, ses lignes et ses colonnes sont indicées par un même ensemble $\{i_1, i_2, \ldots, i_K\}$, $i_1 < i_2 < \ldots < i_K$. Il est facile de vérifier par induction que le sous-ensemble V' induit un graphe complet d'ordre K. $\boxed{4.6.11}$

COROLLAIRE 4.6.12 ISO SOUS-MATRICE EXACT-$\triangle$ *est un problème* NP*-complet.*

4.6.3 $\mathbb{T} = \triangle_K$ (ou $\triangledown_K$)

Nous étudions maintenant les problèmes proposés dans le cas $\mathbb{T} = \triangle_K$ (ou $\triangledown_K$). Nous avons déjà montré que décider s'il existe une matrice de permutation d'ordre n telle que $PAP^T \leq \triangle_n$ est un problème NP-complet même si la matrice A est symétrique (problème ARRANGEMENT LINÉAIRE TRIANGULAIRE proposition 4.4.8 page 75). Remarquons alors que le problème ISO SOUS-MATRICE INF-$\triangle$ est équivalent au problème ARRANGEMENT LINÉAIRE TRIANGULAIRE pour $K = n$. Le corollaire suivant est alors immédiat.

COROLLAIRE 4.6.13 ISO SOUS-MATRICE INF-$\triangle$ *et* ISO SOUS-MATRICE INF-$\triangledown$ *sont des problèmes* NP*-complets même si la matrice A est symétrique et chaque ligne et colonne contient au plus trois éléments non nuls.*

Nous avons en fait déjà rencontré deux extensions de ce problème. Plus précisément, soit A la matrice d'adjacence d'un graphe G d'ordre n, décider s'il existe une matrice de permutation P telle que

$$PAP^T = \begin{bmatrix} B & * \\ * & * \end{bmatrix}$$

où B est une matrice d'ordre K qui satisfait l'une des deux conditions suivantes :

1. $B = \mathbb{C}_K$
2. $\mathbb{C}_K \leq B \leq \triangle_K$

est un problème NP-complet. En effet, la première condition correspond à la recherche d'un couplage fort dans G , tandis que la seconde correspond à la recherche d'un couplage unique restreint dans G. Autrement dit,

1. $B = \mathbb{C}_K$ si et seulement si il existe un couplage fort de taille $\frac{K}{2}$ dans G.
2. $\mathbb{C}_K \leq B \leq \triangle_K$ si et seulement si il existe un couplage unique restreint de taille $\frac{K}{2}$ dans G.

Nous pouvons associer au problème ISO SOUS-MATRICE INF-$\triangle$ le problème MAX ARRANGEMENT LINÉAIRE TRIANGULAIRE : Étant donnés un graphe G et un entier positif K, existe-t-il un sous-graphe induit de G de taille K pour lequel il existe un arrangement linéaire triangulaire? Nous montrons dans la proposition suivante qu'il existe un algorithme d'approximation pour le problème MAX ARRANGEMENT LINÉAIRE TRIANGULAIRE dont le ratio dépend du meilleur ratio d'approximation disponible pour le problème STABLE.

PROPOSITION 4.6.14 *Il existe un algorithme d'approximation de ratio $2r$ pour le problème* MAX ARRANGEMENT LINÉAIRE TRIANGULAIRE, *où r est le ratio d'approximation pour le problème* STABLE.

PREUVE

Considérons une instance arbitraire de notre problème MAX ARRANGEMENT LINÉAIRE TRIANGULAIRE donnée par un graphe G. Soient $OPT_{\text{ES}}(G)$ la taille maximale d'un ensemble stable dans G et $OPT_{\text{ALT}}(G)$ la taille maximale d'un sous-graphe induit de G pour lequel il existe un arrangement linéaire triangulaire. Nous vérifions facilement que $OPT_{\text{ES}}(G)$ et $OPT_{\text{ALT}}(G)$ satisfont la relation suivante:

$$OPT_{\text{ES}}(G) \geq \frac{1}{2}\, OPT_{\text{ALT}}(G). \tag{4.6}$$

Supposons maintenant que nous utilisions un algorithme d'approximation $\mathcal{A}_{\text{ES}}$ de ratio r pour obtenir un ensemble stable dans G. Alors, $\mathcal{A}_{\text{ES}}(G)$ et $OPT_{\text{ES}}(G)$ satisfont la relation suivante:

$$\mathcal{A}_{\text{ES}}(G) \geq \frac{1}{r}\, OPT_{\text{ES}}(G). \tag{4.7}$$

Soit $V' \subseteq V$ l'ensemble stable retourné par l'algorithme $\mathcal{A}_{\text{ES}}(G)$. Considérons maintenant l'algorithme $\mathcal{A}_{\text{ALT}}$ défini de la façon suivante: si $V' = V$ alors retourner V, sinon retourner $V \cup \{u\}$ où u est un sommet quelconque de $V \setminus V'$. Nous vérifions aisément qu'il existe toujours un arrangement linéaire triangulaire pour le graphe induit par le sous-ensemble de sommets retourné par l'algorithme $\mathcal{A}_{\text{ALT}}(G)$. Par conséquent, nous avons $\mathcal{A}_{\text{ALT}}(G) \geq \mathcal{A}_{\text{ES}}(G)$ pour toute instance G. En combinant ce résultat avec (4.6) et (4.7), nous obtenons:

$$\mathcal{A}_{\text{ALT}}(G) \geq \frac{1}{r}\, OPT_{\text{ES}}(G) \geq \frac{1}{2r}\, OPT_{\text{ALT}}(G).$$

Par conséquent,

$$\frac{OPT_{\text{ALT}}(G)}{\mathcal{A}_{\text{ALT}}(G)} \leq 2r.$$

$\boxed{4.6.14}$

Le plus petit ratio d'approximation connu pour le problème STABLE est - *à notre connaissance* - actuellement $O(n/(\log n)^2)$ [BH92], où n est le nombre de sommets du graphe[14]. Pour certaines classes de graphes, le problème STABLE est résoluble en temps polynomial; par exemple, il existe des algorithmes polynomiaux exacts pour les graphes biparti (par couplage), les graphes triangulés et les graphes circulaires.

REMARQUE 4.6.15 Soit G un graphe pour lequel le problème STABLE est résoluble en temps polynomial et le problème MAX ARRANGEMENT LINÉAIRE TRIANGULAIRE est NP-complet. Alors, il existe un algorithme d'approximation de ratio 2 pour le problème

14. Il est toujours surprenant de comparer ce ratio avec les ratios d'approximation disponibles pour le problème COUVERTURE DES SOMMETS; après tout, un sous-ensemble $V' \subseteq V$ est une couverture des sommets si et seulement si $V \setminus V'$ est un ensemble stable. Cependant, il existe des algorithmes (très simples) d'approximation de ratio 2 pour le problème STABLE. En d'autres termes, si les problèmes de décision STABLE et COUVERTURE DES SOMMETS sont équivalents, il semble que les problème d'approximation associés soient radicalement différents; l'approximation d'un ensemble stable ne nous aide que très peu pour approximer une couverture des sommets. Considérons par exemple un graphe G dont la plus petite *couverture des sommets* contient $\frac{n}{2} - 1$ sommets. Alors, un algorithme d'approximation de ratio 2 pour le problème COUVERTURE DES SOMMETS calcule une couverture des sommets contenant au plus $2(\frac{n}{2} - 1) = n - 2$ sommets. Cependant, cette couverture des sommets ne nous donne qu'un ensemble stable de taille 2; à comparer avec le plus grand *ensemble stable* qui contient $\frac{n}{2} + 1$ sommets.

MAX ARRANGEMENT LINÉAIRE TRIANGULAIRE. Nous ignorons cependant si un tel graphe existe. En effet, nous n'avons pas pu mettre en évidence un graphe pour lequel un ensemble stable de cardinalité maximale est calculable en temps polynomial et le problème MAX ARRANGEMENT LINÉAIRE TRIANGULAIRE est NP-complet. ◇

Nous considérons maintenant le problème de la recherche de la plus grande sous-matrice principale exactement triangulaire.

PROPOSITION 4.6.16 ISO SOUS-MATRICE EXACT-$\triangle$ *est un problème* NP-*complet.*

PREUVE

Le problème ISO SOUS-MATRICE EXACT-$\triangle$ appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème EXACT 3SAT donnée par une formule booléenne ϕ sous forme normale conjonctive d'ordre 3, c'est-à-dire avec exactement trois littéraux par clauses. Nous notons $C = \{c_1,c_2,\ldots,c_m\}$ l'ensemble des clauses, et $X = \{x_1,x_2,\ldots,x_n\}$ l'ensemble des variables booléennes. Comme toujours, nous notons $\lambda_{i,j}$ le j^{eme} littéral de la clause i. L'instance correspondante du problème ISO SOUS-MATRICE EXACT-$\triangle$ est donnée par une (0,1)- matrice $A = [a_{i,j}]$ d'ordre $4m$ et par un entier positif $K = 2m$. Par souci de clarté, nous indiçons les lignes (et les colonnes) de A par les littéraux et les clauses de la formule booléenne ϕ. Plus formellement, nous indiçons la matrice A par $\Lambda = \{\lambda_{i,j} \mid 1 \leq i \leq m \text{ et } 1 \leq j \leq 3\}$ et par c_i $(1 \leq i \leq m)$. La matrice A est définie par :

$$\begin{cases} a_{c_i,c_j} = 1 & \text{pour } 1 \leq i \leq m \text{ et } 1 \leq j \leq m \\ a_{\lambda_{i,j},c_k} = a_{c_k,\lambda_{i,j}} = 1 & \text{pour } 1 \leq k \leq i \leq m \text{ et } 1 \leq j \leq 3 \\ a_{\lambda_{i,j},\lambda_{k,\ell}} = 1 & \text{si et seulement si } 1 \leq i,k \leq m,\ 1 \leq j,\ell \leq 3 \text{ et } \lambda_{i,j} = \neg\lambda_{j,\ell} \end{cases}$$

Remarquons que la matrice A ainsi définie est symétrique. De plus, la diagonale principale de A contient exactement m éléments non nuls : a_{c_i,c_i} pour $1 \leq i \leq m$. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une interprétation satisfaisante f de la formule booléenne ϕ. Sans perte de généralité, nous pouvons supposer que chaque clause est satisfaite par son premier littéral. Soit P une matrice de permutation d'ordre $4m$ telle que :

$$PAP^T = \begin{bmatrix} B & * \\ * & * \end{bmatrix}$$

où B est une matrice d'ordre $2m$ dont la ligne (colonne) i est indicée par :

- $\lambda_{i,1}$ pour $1 \leq i \leq m$
- c_{2m-i+1} pour $m+1 \leq i \leq 2m$.

La construction est détaillée en figure 4.9. Montrons que $B = \triangle_{2m}$. Considérons la décomposition en matrices carrées de taille m :

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

La sous-matrice (principale) B_{22} est indicée par c_i, $1 \leq i \leq m$. Puisque $a_{c_i,c_j} = 1$ pour $1 \leq i,j \leq m$, alors nous avons $B_{22} = 1_{m,m}$. De même, puisque $a_{\lambda_{i,j},c_k} =$

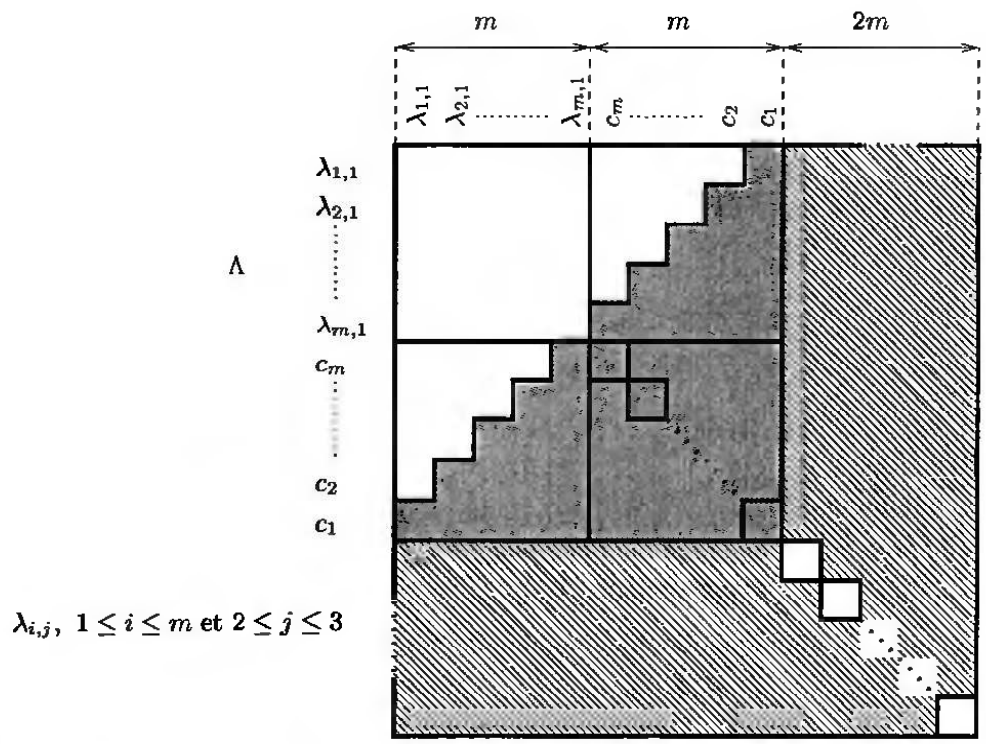


FIG. 4.9 – *Illustration de la construction utilisée dans la preuve de la proposition 4.6.16.*

$a_{c_k,\lambda_{i,j}} = 1$ pour $1 \leq k \leq i \leq m$ et $1 \leq j \leq 3$, il vient $B_{12} = B_{21} = \triangle_m$. Enfin, puisque f est une interprétation satisfaisante, alors $B_{11} = \mathbf{0}_{m,m}$. Par conséquent, nous avons $B = \triangle_{2m}$.

Inversement, supposons qu'il existe une matrice de permutation P d'ordre $4m$ telle que :

$$PAP^T = \begin{bmatrix} \triangle_{2m} & * \\ * & * \end{bmatrix}.$$

Considérons la décomposition en matrices carrées de taille m de $\triangle_{2m}$:

$$\triangle_{2m} = \begin{bmatrix} \mathbf{0}_{m,m} & \triangle_m \\ \triangle_m & \mathbf{1}_{m,m} \end{bmatrix}.$$

Puisque la diagonale principale de la matrice $\mathbf{1}_{m,m}$ (et donc de $\triangle_{2m}$) contient exactement m éléments non nuls, alors les m dernières lignes et colonnes de $\triangle_{2m}$ sont indicées par c_i pour $1 \leq i \leq m$. En effet, il existe exactement m éléments non nuls sur la diagonale principale de A et ses éléments correspondent aux a_{c_i,c_i} pour $1 \leq i \leq m$. Par suite, les m premières lignes et colonnes de $\triangle_{2m}$ sont indicées par m éléments de l'ensemble $\{\lambda_{i,j} \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\}$. Remarquons maintenant que $a_{\lambda_{i,j},c_k} = a_{c_k,\lambda_{i,j}} = 1$ pour $1 \leq k \leq i \leq m$ et $1 \leq j \leq 3$. Par suite,

1. la i^{eme} ligne (colonne), $1 \leq i \leq m$, de $\triangle_{2m}$ est indicée par $\lambda_{i,1}$, $\lambda_{i,2}$ ou $\lambda_{i,3}$,

2. la i^{eme} ligne (colonne), $m+1 \leq i \leq 2m$, de $\triangle_{2m}$ est indicée par c_{2m-i+1}.

Autrement dit, les i premières lignes (colonnes) de $\triangle_{2m}$ sont indicées par exactement un littéral de chaque clause. Notons Λ les m littéraux qui indicent les m premières lignes (colonnes) de $\triangle_{2m}$. Montrons que les littéraux de Λ déterminent une interprétation satisfaisante de la formule booléenne ϕ. Dans un premier temps, il n'existe pas dans Λ deux littéraux $\lambda_{i,j}$ et $\lambda_{i',j'}$ tels que $\lambda_{i,j} = x_k$ et $\lambda_{i',j'} = \bar{x}_k$. Il suffit pour cela de remarquer que la sous-matrice (principale) indicée par les littéraux de Λ ne contient que des 0; or, $a_{\lambda_{i,j},\lambda_{k,\ell}} = 1$ pour $1 \leq i,k \leq m$, $1 \leq j,\ell \leq 3$ et $\lambda_{i,j} = \neg\lambda_{j,\ell}$. Alors, l'interprétation f définie par :

$$f(x_k) = \begin{cases} \text{vrai} & \text{s'il existe } \lambda_{i,j} \in \Lambda \text{ tel que } \lambda_{i,j} = x_k \\ \text{faux} & \text{s'il existe } \lambda_{i,j} \in \Lambda \text{ tel que } \lambda_{i,j} = \bar{x}_k \\ \text{vrai ou faux} & \text{s'il existe pas } \lambda_{i,j} \in \Lambda \text{ tel que } \lambda_{i,j} = x_k \text{ ou } \lambda_{i,j} = \bar{x}_k \end{cases}$$

est une interprétation satisfaisante de la formule booléenne ϕ. $\boxed{4.6.16}$

Nous pouvons reformuler la proposition 4.6.16 sur les graphes avec boucles.

COROLLAIRE 4.6.17 *Soient $G = (V,E)$ un graphe d'ordre n avec éventuellement des boucles sur les sommets, et K un entier positif. Décider s'il existe un sous-ensemble $V' \subseteq V$, $|V'| = K$, et une bijection σ de V' dans $[\![K]\!]$ tels que :*

$$\forall u \in V' \; \forall v \in V', \qquad \{u,v\} \in E \quad \textit{si et seulement si} \quad \sigma(u) + \sigma(v) > K$$

est un problème NP*-complet.*

4.7 Arrangement linéaire à cumul croissant

4.7.1 Introduction

Nous allons dans cette section reformuler le problème MATRICE TRIANGULAIRE en termes de *voisinages* dans un graphe non orienté. Rappelons que, étant donné un graphe $G = (V,E)$, le voisinage d'un sommet $u \in V$, noté $N(u)$, est l'ensemble des sommets adjacents à u, *i.e.*

$$\forall u \in V, \qquad N(u) = \{v \mid v \in V \;\wedge\; \{u,v\} \in E\}.$$

Par extention, pour tout $V' \subseteq V$, nous notons $N(V') = \bigcup_{v \in V'} N(v)$. Nous considérons dans la suite le problème ARRANGEMENT LINÉAIRE À CUMUL CROISSANT défini ci-après.

PROBLÈME 4.7.1 (ARRANGEMENT LINÉAIRE À CUMUL CROISSANT)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n.
QUESTION : Existe-t-il un arrangement linéaire à cumul croissant de G, *i.e.* une bijection σ de V dans $[\![n]\!]$ telle que card $\left(\bigcup_{\sigma(u) \leq i} N(u)\right) \leq i$ pour $1 \leq i \leq n$?

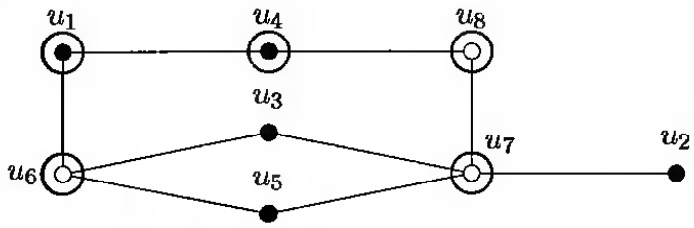
Nous vérifions facilement que notre nouveau problème ARRANGEMENT LINÉAIRE À CUMUL CROISSANT est équivalent au problème MATRICE INF-TRIANGULAIRE dans le cas symétrique. En effet, soit G un graphe d'ordre n et notons A sa matrice d'adjacence. Alors, il existe un arrangement linéaire à cumul croissant de G si et seulement si il existe les matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$.

REMARQUE 4.7.2 Soient $G = (V,E)$ un graphe d'ordre n et σ une bijection de V dans $[\![n]\!]$. Si σ est un arrangement linéaire triangulaire, alors σ est un arrangement linéaire à cumul croissant. L'inverse n'est pas toujours vrai. ◇

Nous allons examiner une restriction du problème ARRANGEMENT LINÉAIRE À CUMUL CROISSANT. Cette restriction se distingue par la présence d'un paramètre J (entier positif) supplémentaire dans l'instance qui indique le nombre maximum de sommets pour lequel σ est un arrangement linéaire à cumul croissant. Nous appelons ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX ce problème.

PROBLÈME 4.7.3 (ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n et un entier positif J.
QUESTION : Existe-t-il une bijection σ de V dans $[\![n]\!]$ telle que $\text{card}\left(\bigcup_{\sigma(u)\leq i} N(u)\right) \leq i$ pour $1 \leq i \leq J$?

EXEMPLE 14 *Soit $G = (V,E)$ le graphe défini ci-après :*



La bijection σ de V dans $[\![8]\!]$ définie par :

$$\sigma(u_1) = 4 \quad \sigma(u_2) = 1 \quad \sigma(u_3) = 3 \quad \sigma(u_4) = 5$$
$$\sigma(u_5) = 2 \quad \sigma(u_6) = 6 \quad \sigma(u_7) = 7 \quad \sigma(u_8) = 8$$

est un arrangement linéaire à cumul croissant maximum pour $J = 5$ car

$$\text{card}\Big(\bigcup_{\sigma'(u_i)\leq 1} N(u_i)\Big) = \text{card}(\{u_7\}) \leq 1;$$

$$\text{card}\Big(\bigcup_{\sigma'(u_i)\leq 2} N(u_i)\Big) = \text{card}(\{u_6,u_7\}) \leq 2;$$

$$\text{card}\Big(\bigcup_{\sigma'(u_i)\leq 3} N(u_i)\Big) = \text{card}(\{u_6,u_7\}) \leq 3;$$

$$\text{card}\Big(\bigcup_{\sigma'(u_i)\leq 4} N(u_i)\Big) = \text{card}(\{u_4,u_6,u_7\}) \leq 4;$$

$$\text{card}\Big(\bigcup_{\sigma'(u_i)\leq 5} N(u_i)\Big) = \text{card}(\{u_1,u_4,u_6,u_7,u_8\}) \leq 5.$$

◇

4.7.2 Complexité

Nous montrons dans cette sous-section que le problème ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX est un problème NP-complet même si G est un graphe de séparation[15]. Rappelons qu'un graphe $G = (V,E)$ est un graphe de séparation s'il existe une partition $V = S \cup K$ telle que S induit un ensemble stable et K induit une clique.

PROPOSITION 4.7.4 ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX *est un problème* NP-*complet même si G est un graphe de séparation.*

PREUVE

Le problème ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX appartient à NP. Montrons qu'il est NP-complet. Soit une instance du problème CLIQUE donnée par un graphe $G = (V,E)$, $|V| = n$ et $|E| = m$, et par un entier positif K. Sans perte de généralité, nous pouvons supposer $n > K + 1$. Par souci de clarté, nous notons $E = \{e_1,e_2,\ldots,e_m\}$. Soient V_1^*, V_2^*, V_3^* et V_4^* les ensembles de sommets disjoints définis par :

$$\begin{cases} V_1^* = E \\ V_2^* = V \\ V_3^* = \{e'_i \mid 1 \leq i \leq m\} \\ V_4^* = \{x_i \mid 1 \leq i \leq K\} \end{cases}$$

et E_1^*, E_2^*, E_3^*, E_4^* et E_5^* les ensembles d'arêtes définis par :

$$\begin{cases} E_1^* = \{\{e_i,u\} \mid e_i \in V_1^* \ \wedge \ u \in V_2^* \ \wedge \ u \in e_i\} \\ E_2^* = \{\{e_i,e'_i\} \mid 1 \leq i \leq m\} \\ E_3^* = \{\{e'_i,e'_j\} \mid e'_i \in V_3^* \ \wedge \ e'_j \in V_3^* \ \wedge \ e'_i \neq e'_j\} \\ E_4^* = \{\{e'_i,u_j\} \mid e'_i \in V_3^* \ \wedge \ u_j \in V_2^*\} \\ E_5^* = \{\{u_i,u_j\} \mid u_i \in V_2^* \ \wedge \ u_j \in V_2^* \ \wedge \ u_i \neq u_j\} \end{cases}$$

Remarquons que les sommets de V_4^* ne sont pas connectés. L'instance correspondante du problème ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX est donnée par un graphe $G^* = (V^*,E^*)$ et par un entier positif J définis par :

- $V^* = V_1^* \cup V_2^* \cup V_3^* \cup V_4^*$
- $E^* = E_1^* \cup E_2^* \cup E_3^* \cup E_4^* \cup E_5^*$
- $J = \frac{1}{2}K(K+1)$

15. Rappelons que nous avons établi que si le problème MATRICE INF-TRIANGULAIRE est un problème NP-complet, alors le problème ARRANGEMENT LINÉAIRE TRIANGULAIRE est un problème NP-complet même pour la classe des graphes de séparation.

Puisque V^* est une union disjointe, alors $|V^*| = 2m + n + K$. De plus,

$$|E^*| = 3m + \frac{1}{2}(n+m)(n+m-1).$$

Remarquons que G^* est un graphe de séparation. En effet, $V_1^* \cup V_4^*$ est un ensemble stable dans G^*, et $V_2^* \cup V_3^*$ induit une clique. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une clique $V' \subseteq V$ de taille K dans G. Soit $u^* \in V'$ un sommet distingué. Réordonnons les sommets de V^* par la bijection σ de V^* dans $[\![2m+n+K]\!]$ définie par :

$$\begin{cases} 1 \leq \sigma(x_i) \leq K & \text{si } x_i \in V_4^* \\ K+1 \leq \sigma(e_i) \leq 2K-1 & \text{si } e_i = \{u,v\} \in V_1^*,\ u = u^* \text{ et } v \in V' \setminus \{u^*\} \\ 2K \leq \sigma(e_i) \leq J & \text{si } e_i = \{u,v\} \in V_1^*,\ u \in V' \setminus \{u^*\} \text{ et } v \in V' \setminus \{u^*\} \\ J+1 \leq \sigma(u) \leq 2m+n+K & \text{sinon} \end{cases}$$

Montrons que $\operatorname{card}\left(\bigcup_{1 \leq \sigma(u) \leq i} N(u)\right) \leq i$ pour $1 \leq i \leq J$. Dans un premier temps,

$$\operatorname{card}\Big(\bigcup_{1 \leq \sigma(u) \leq i} N(u)\Big) = 0 \qquad \text{pour } 1 \leq i \leq K$$

car les sommets de V_4^* ne sont pas connectés. Montrons par induction que

$$\operatorname{card}\Big(\bigcup_{1 \leq \sigma(u) \leq i} N(u)\Big) = 2(i-K)+1 \qquad \text{pour } 1 \leq i \leq 2K-1.$$

C'est vrai pour $i = K+1$ car

$$\operatorname{card}\Big(\bigcup_{1 \leq \sigma(u) \leq K+1} N(u)\Big) = 3 = 2(K+1-K)+1.$$

Supposons que l'assertion soit vraie pour tout $i < 2K-1$. Alors,

$$\begin{aligned} \operatorname{card}\Big(\bigcup_{1 \leq \sigma(u) \leq i+1} N(u)\Big) &= \operatorname{card}\Big(\bigcup_{1 \leq \sigma(u) \leq i} N(u)\Big) + 2 \\ &= 2(i-K)+1+2 \\ &= 2(i+1-K)+1. \end{aligned}$$

Or, $i \leq 2K-1 \Leftrightarrow 2(i-K)+1 \leq i$. Par suite,

$$\operatorname{card}\Big(\bigcup_{1 \leq \sigma(u) \leq i} N(u)\Big) \leq i \qquad \text{pour } 1 \leq i \leq 2K-1$$

avec égalité pour $i = 2K-1$. Enfin, en remarquant maintenant que u^* est connecté à tous les sommets de la clique V', il vient :

$$\operatorname{card}\Big(\bigcup_{1 \leq \sigma(u) \leq i} N(u)\Big) = i \qquad \text{pour } 1 \leq i \leq J.$$

Inversement, supposons qu'il existe une bijection σ de V^* dans $[\![2m+n+K]\!]$ telle que $\text{card}\left(\bigcup_{\sigma(u)\leq i} N(u)\right) \leq i$ pour $1 \leq i \leq J$. Remarquons dans un premier temps que :

$$\begin{aligned} \forall e' \in V_3^*, &\quad \text{card}(N(e')) > n+m-1 \\ \forall u \in V_2^*, &\quad \text{card}(N(u)) > n+m-1 \end{aligned}$$

car $V_2^* \cup V_3^*$ induit une clique de taille $n+m$. Mais,

$$\begin{aligned} n+m-1 &> K+1+m-1 && \text{car } n > K+1 \\ &\geq K+\frac{1}{2}K(K-1) && \text{car } m \geq \frac{1}{2}K(K-1) \\ &= J. \end{aligned}$$

D'où,

$$\begin{aligned} \forall e' \in V_3^*, &\quad \text{card}(N(e')) > J; \\ \forall u \in V_2^*, &\quad \text{card}(N(u)) > J. \end{aligned}$$

Par suite, si $\sigma(u) \leq J$, alors $u \in V_1^* \cup V_4^*$. Notons $V' \subset V^*$ le sous-ensemble défini par :

$$V' = \{u \mid u \in V^* \ \wedge \ 1 \leq \sigma(u) \leq J\}$$

Alors,

$$J \geq |V_1^* \cap V'| \geq \frac{1}{2}K(K-1)$$

car $|V_4^*| = K$. La borne minimale (resp. maximale) est atteinte lorsque $V_4^* \cap V' = V_4^*$ (resp. $V_4^* \cap V' = \emptyset$). Puisque $\{e_i, e_i'\} \in E^*$ pour tout $1 \leq i \leq m$, il vient

$$\text{card}\left(V_3^* \cap \bigcup_{u \in V_1^* \cap V'} N(u)\right) = |V_1^* \cap V'| \geq \frac{1}{2}K(K-1)$$

Par conséquent,

$$\text{card}\left(V_2^* \cap \bigcup_{u \in V_1^* \cap V'} N(u)\right) \leq K.$$

Autrement dit, il existe dans G au moins $\frac{1}{2}K(K-1)$ arêtes incidentes à au plus K sommets. Nous en concluons qu'il existe une clique de taille K dans G. $\boxed{4.7.4}$

En rappelant qu'un graphe de séparation est un graphe triangulé[16], nous obtenons immédiatement le résultat suivant.

COROLLAIRE 4.7.5 ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX *est un problème* NP-*complet même si G est un graphe triangulé.*

16. Plus précisément, S. Földes et P.J. Hammer ont montré qu'un graphe G est un graphe de séparation si et seulement si G et sont complément $\overline{G}$ sont triangulés.

La proposition 4.7.4 se traduit immédiatement sur la (0,1)-matrice d'adjacence associée au graphe G.

COROLLAIRE 4.7.6 *Soient A une (0,1)-matrice symétrique d'ordre n et K un entier positif. Décider s'il existe des matrices de permutation P et Q d'ordre n telles que :*

$$PAQ^T \leq \begin{bmatrix} \llcorner_K & \mathbf{0}_{K,n-K} \\ \mathbf{1}_{n-K,K} & \mathbf{1}_{n-K,n-K} \end{bmatrix}$$

est un problème NP*-complet.*

En fait, nous pouvons relâcher la contrainte et montrer facilement que, étant donnés une (0,1)-matrice A symétrique d'ordre n et un entier positif K, décider s'il existe des matrices de permutation P et Q d'ordre n telles que :

$$PAQ^T \leq \begin{bmatrix} \mathbf{1}_{K,K} & \mathbf{0}_{K,n-K} \\ \mathbf{1}_{n-K,K} & \mathbf{1}_{n-K,n-K} \end{bmatrix}$$

est un problème NP-complet.

Nous avons déjà signalé que MATRICE INF-TRIANGULAIRE est un problème NP-complet si et seulement si ISO MATRICE INF-$\triangle$[17] restreint aux (0,1)-matrices symétriques d'ordre $2n$ de la forme :

$$\begin{bmatrix} \mathbf{0}_{n,n} & B \\ B^T & \mathbf{1}_{n,n} \end{bmatrix}$$

où B est une (0,1)-matrice d'ordre n est un problème NP-complet. La proposition 4.7.4 nous donne en fait un résultat «*proche*» de celui recherché.

COROLLAIRE 4.7.7 *Soient A une (0,1)-matrice symétrique d'ordre n_1+n_2, $n_1 \leq n_2$, de la forme :*

$$A = \begin{bmatrix} \mathbf{0}_{n_1,n_1} & B \\ B^T & \mathbf{1}_{n_2,n_2} \end{bmatrix}$$

où B est une (0,1)-matrice de taille n_1 par n_2 et $K \leq n_1$ un entier positif. Décider s'il existe une matrice de permutation P d'ordre n_1+n_2 telles que :

$$PAP^T \leq \begin{bmatrix} \mathbf{0}_{K,K} & \mathbf{0}_{K,n_1+n_2-2K} & \triangle_K \\ \mathbf{0}_{n_1+n_2-2K,K} & \mathbf{1}_{n_1+n_2-2K,n_1+n_2-2K} & \mathbf{1}_{n_1+n_2-2K,K} \\ \triangle_K & \mathbf{1}_{K,n_1+n_2-2K} & \mathbf{1}_{K,K} \end{bmatrix}$$

est un problème NP*-complet.*

4.8 Conclusion et problèmes ouverts

Le fil conducteur de ce chapitre a été l'étude de la complexité du problème MATRICE INF-TRIANGULAIRE, *i.e.* étant donnée une (0,1)-matrice A d'ordre n, existe-t-il des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \llcorner_n$? Nous

17. La version matricielle du problème ARRANGEMENT LINÉAIRE TRIANGULAIRE.

n'avons pas été en mesure ni de proposer un algorithme en temps polynomial, ni de montrer que le problème est NP-complet dans le cas général.

Nous avons tout d'abord montré que le problème MATRICE INF-TRIANGULAIRE est résoluble en temps quadratique lorsque le permanent est non nul[18] (rappelons que nous pouvons décider en temps polynomial si $\mathrm{per}(A) > 0$ en calculant un couplage dans le graphe biparti associé). Nous avions déjà vu au chapitre précédent[19] que si la matrice A d'ordre n contient au plus $n+1$ éléments non nuls, alors elle est triangularisable par permutation des lignes et des colonnes.

Un autre cas intéressant reste sans réponse. Rappelons tout d'abord qu'une (0,1)-matrice possède la propriété des 1 consécutifs pour les lignes s'il est possible de permuter ses colonnes afin que les éléments non nuls de chaque ligne apparaissent consécutivement [BL76, HPV99]. Un hypergraphe $\mathcal{H}$ est un *hypergraphe d'intervalle* si et seulement si sa matrice d'incidence possède la propriété des 1 consécutifs pour les lignes.

QUESTION 2 *Quelle est la complexité du problème* MATRICE INF-TRIANGULAIRE *si A est la matrice d'incidence d'un hypergraphe d'intervalle?*

La matrice d'incidence d'un hypergraphe d'intervalle est une (0,1)-matrice totalement unimodulaire [Ber76]. Aussi, plus généralement, nous posons la question concernant la complexité du problème MATRICE INF-TRIANGULAIRE lorsque A est une (0,1)-matrice d'ordre n totalement unimodulaire. Nous retrouverons cette question, sous une forme différente, dans le chapitre suivant.

Nous avons ensuite introduit le problème ARRANGEMENT LINÉAIRE TRIANGULAIRE. Nous avons remarqué que si le problème ARRANGEMENT LINÉAIRE TRIANGULAIRE est résoluble en temps polynomial, alors le problème MATRICE INF-TRIANGULAIRE est également résoluble en temps polynomial. Malheureusement, nous avons prouvé que ARRANGEMENT LINÉAIRE TRIANGULAIRE est un problème NP-complet[20]. Il est cependant résoluble en temps polynomial si G est un graphe acyclique[21]. De plus, nous avons vu que si MATRICE INF-TRIANGULAIRE est un problème NP-complet, alors ARRANGEMENT LINÉAIRE TRIANGULAIRE est un problème NP-complet même si G est un graphe de séparation[22].

QUESTION 3 *Quelle est la complexité du problème* ARRANGEMENT LINÉAIRE TRIANGULAIRE *si G est un graphe de séparation?*

Le cas des graphes bipartis reste également sans réponse.

QUESTION 4 *Quelle est la complexité du problème* ARRANGEMENT LINÉAIRE TRIANGULAIRE *si G est un graphe biparti?*

Plus généralement, nous ignorons en fait la complexité du problème ARRANGEMENT LINÉAIRE TRIANGULAIRE lorsque le graphe G est *parfait* [Ber75, Gol80].

18. proposition 4.3.7 page 67.
19. proposition 3.5.7 page 44.
20. proposition 4.4.8 page 75.
21. proposition 4.4.16 page 85.
22. corollaire 4.4.10 page 78.

Nous avons ensuite étudié des problèmes sur les formes triangulaires dans les sous-matrices (tous ces problèmes sont NP-complets dans le cas des sous-matrices principales). Nous avons en particulier montré que décider s'il existe une sous-matrice de taille K, K entier positif donné dans l'instance, exact-triangulaire est un problème NP-complet[23]. De plus, nous avons montré[24] que si SOUS-MATRICE INF-TRIANGULAIRE est un problème NP-complet, alors MATRICE INF-TRIANGULAIRE est également un problème NP-complet. Cependant, nous n'avons pas été en mesure de donner la complexité du problème SOUS-MATRICE INF-TRIANGULAIRE. Cependant, nous avons remarqué que la version pondérée de ce dernier problème est NP-complète[25].

QUESTION 5 *Quelle est la complexité du problème* SOUS-MATRICE INF-TRIANGULAIRE *?*

23. proposition 4.5.6 page 90.
24. proposition 4.5.3 page 87.
25. proposition 4.5.14 page 97.

Chapitre 5

Résultats complémentaires

Sommaire

5.1 Silhouette **117**
5.2 Surface **127**
5.3 Famille de couples de sous-ensembles **150**
5.4 Partition **154**
5.5 Conclusion et problèmes ouverts **161**

Ce chapitre est le prolongement de l'étude de la complexité du problème MATRICE INF-TRIANGULAIRE[1] *commencée au chapitre précédent. Nous présentons plusieurs généralisations et extensions de ce problème, ainsi que quelques problèmes complémentaires sur les* «formes triangulaires» *dans les* (0,1)*-matrices.*

Nous étudions dans la section 5.1 les silhouettes *des* (0,1)*-matrices et présentons un problème* NP*-complet qui contient le problème* MATRICE INF-TRIANGULAIRE *(proposition 5.1.7). La section 5.2 introduit la triangularisation d'une* (0,1)*-matrice par permutation des lignes et des colonnes comme un problème de compression. En particulier, nous montrons que la* surface *d'une* (0,1)*-matrice est la mesure* «naturelle» *associée à sa silhouette. Le problème de décision associé à la minimisation de surface s'avère* NP*-complet (proposition 5.2.18). Nous proposons ensuite en section 5.3 une nouvelle généralisation du problème* MATRICE INF-TRIANGULAIRE *basée sur la comparaison des cardinaux de couples de sous-ensembles que nous montrons* NP*-complète dans le cas général (proposition 5.3.2). Enfin, nous étudions dans la section 5.4 quelques problèmes de partition d'une* (0,1)*-matrice en sous-matrices principales inf / exact / sup-triangulaires.*

5.1 Silhouette

5.1.1 Introduction

Soit $A = [a_{i,j}]$ un (0,1)-matrice d'ordre n. Pour $1 \leq i \leq n$, nous notons $f_i(A)$ l'indice de la colonne du premier élément non nul dans la partie inférieure triangulaire

1. problème 4.1.1 page 58.

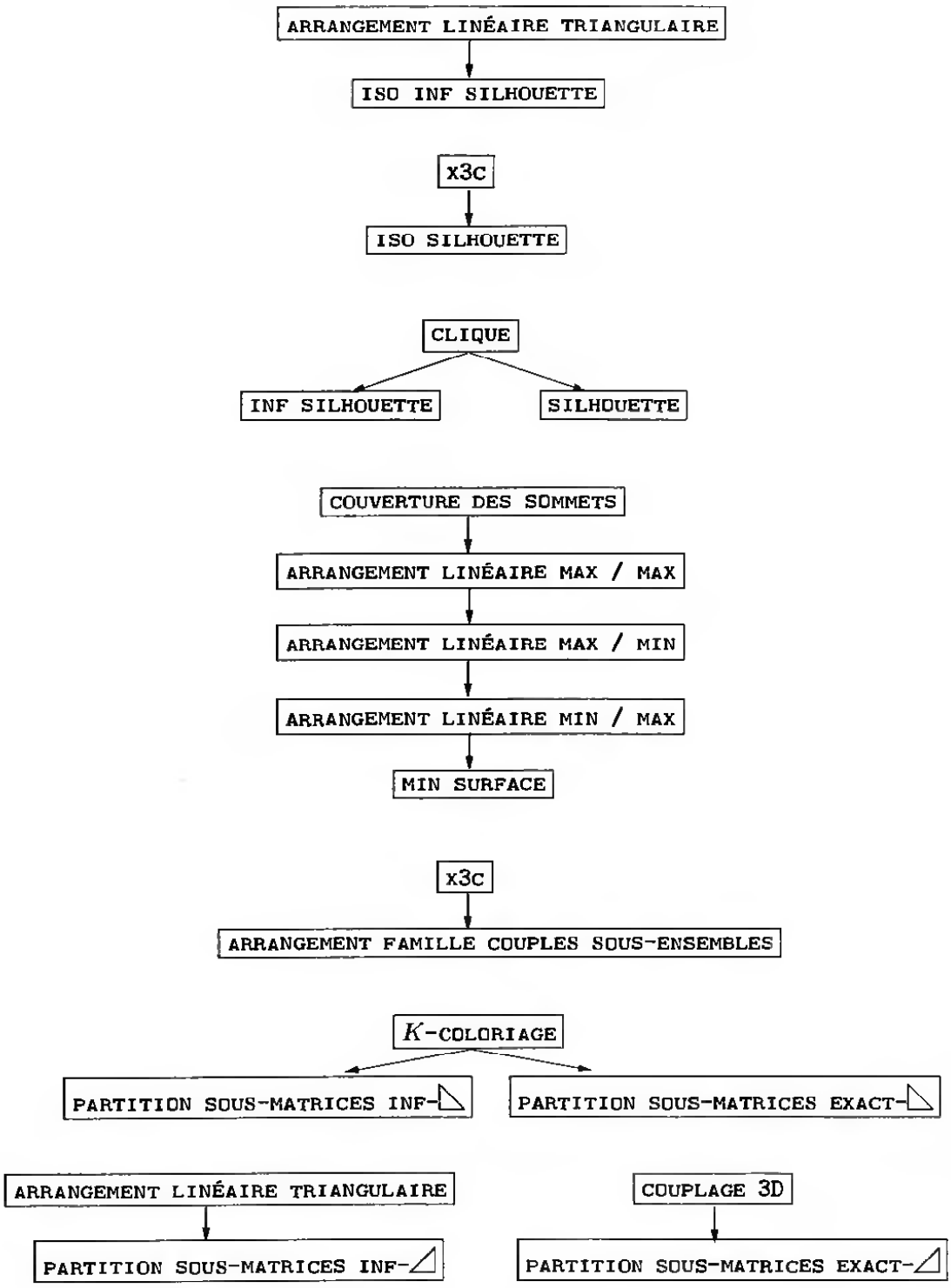


FIG. 5.1 – *Schéma des réductions polynomiales présentées dans ce chapitre.*

de la ligne i, ou i s'il n'en existe aucun, *i.e.* :

$$\forall i, 1 \leq i \leq n, \qquad f_i(A) = \min \begin{cases} j & a_{i,j} = 1 \text{ et } 1 \leq j < i \\ i & \end{cases}$$

EXEMPLE 15 *Soit A la (0,1)-matrice symétrique d'ordre 6 définie par :*

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Alors,

$$\begin{array}{lll} f_1(A) = 1 & f_2(A) = 1 & f_3(A) = 3 \\ f_4(A) = 1 & f_5(A) = 3 & f_6(A) = 4 \end{array}$$

◇

Soit A une (0,1)-matrice symétrique d'ordre n. La *largeur* de la ligne i, $1 \leq i \leq n$, est définie par $i - f_i(A)$. La *largeur de bande* de la matrice A, notée $bw(A)$[2], est la plus grande largeur d'une ligne de A : ***largeur de bande***

$$bw(A) = \max\{i - f_i(A) \mid 1 \leq i \leq n\}.$$

L'*enveloppe* d'une matrice symétrique A d'ordre n, notée $env(A)$, est l'ensemble des points de chaque ligne compris entre le premier élément non nul et la diagonale : ***enveloppe***

$$env(A) = \{(i,j) \mid 1 \leq i \leq n \text{ et } f_i(A) \leq j < i\}.$$

Le *profil* d'une matrice A, noté $prof(A)$[3], est le nombre d'éléments de son enveloppe : ***profil***

$$prof(A) = |\, env(A)|.$$

EXEMPLE 16 *En reprenant la matrice de l'exemple 15, il vient*

$$\begin{aligned} bw(A) &= \max\{0,1,0,3,2,2\} = 3; \\ env(A) &= \{(2,1),(4,1),(4,2),(4,3),(5,3),(5,4),(6,4),(6,5)\}; \\ prof(A) &= |\, env(A)| = 8. \end{aligned}$$

◇

2. pour *bandwidth*
3. parfois noté *Esize*(A)

La valeur de ces paramètres dépend fortement de l'ordre des lignes et des colonnes de la matrice A. Aussi, nous définissons les bornes suivantes :

$$bw^*(A) = \min\{bw(PAP^T) \mid P \text{ est une matrice de permutation d'ordre } n\}$$
$$prof^*(A) = \min\{prof(PAP^T) \mid P \text{ est une matrice de permutation d'ordre } n\}.$$

L'espace nécessaire pour factoriser une matrice en utilisant une méthode d'enveloppe est proportionnel à la taille de l'enveloppe. De même, le nombre d'opérations en virgule flottante est propotionnel à la somme des carrés des largeurs des lignes. Pour ces deux raisons, il est important de calculer un arrangement des lignes et des colonnes de la matrice A qui minimise la largeur de bande et la taille de l'enveloppe.

Ces problèmes sont difficiles dans le cas général. En effet, décider s'il existe une matrice de permutation d'ordre n telle que $bw^*(PAP^T) \leq K$ pour un entier positif K donné est un problème NP-complet [Pap76]. Le problème reste NP-complet même si A est la matrice d'adjacence (1) d'un arbre[4] dont les sommets ont pour degré au plus 3 [GGJK78], ou (2) d'un graphe sur une grille. De même, il est NP-complet si A est la matrice d'adjacence d'un *caterpillar* dont les brins ont pour longueur au plus 3 [Mon86]. Cette borne est optimale puisque le problème est résoluble en temps $O(n \log n)$ pour les caterpillars de longueur de brins au plus 2 [APSZ81]. Le problème est également NP-complet si A est une matrice non-symétrique; en particulier, il est NP-complet même si A est la matrice d'adjacence d'un arbre orienté enraciné où chaque sommet à pour degré sortant au plus 2 (le problème consiste dans ce cas, à minimiser la largeur de bande d'une (0,1)-matrice triangulaire) [GGJK78]. De même, décider s'il existe une matrice de permutation d'ordre n telle que $prof^*(PAP^T) \leq K$, pour un entier positif K donné, est un problème NP-complet [LY93].

5.1.2 Définitions

silhouette

DÉFINITION 5.1.1 (SILHOUETTE) *Soit $A = [a_{i,j}]$ une (0,1)-matrice de taille m par n. La* silhouette *de A, notée $\mathbb{S}_A$, est une application de $[\![n]\!]$ dans $[\![m+1]\!]$ définie par :*

$$\mathbb{S}_A(j) = \begin{cases} j & \text{si } a_{i,j} = 1 \text{ et } a_{i,j'} = 0 \text{ pour } 1 \leq j' \leq j \\ m+1 & \text{sinon} \end{cases}$$

Intuitivement, la silhouette d'une matrice consiste à considérer la matrice A comme une tableau de taille m par n et à fixer pour chaque colonne la «*hauteur*» maximale d'un élément non nul. Plus précisément, la définition 5.1.1 donne pour chaque colonne *exactement* la hauteur maximale d'un élément non nul (sauf si $\mathbb{S}_A = m+1$, auquel cas, la colonne j ne contient que des 0).

EXEMPLE 17 *En reprenant la matrice de l'exemple 15, il vient :*

$$\mathbb{S}_A(1) = 2 \qquad \mathbb{S}_A(2) = 1 \qquad \mathbb{S}_A(3) = 4$$
$$\mathbb{S}_A(4) = 1 \qquad \mathbb{S}_A(5) = 3 \qquad \mathbb{S}_A(6) = 4$$

◇

4. Mentionnons que très peu de problèmes sur les graphes restent NP-complets pour les arbres.

L'étude des silhouettes dans le cadre général du problème de la triangularisation des (0,1)-matrices par permutation des lignes et des colonnes est motivée par les trois remarques suivantes.

REMARQUE 5.1.2 Soit A une (0,1)-matrice d'ordre n. Si $\mathbb{S}_A(j) \leq j$ (resp. $\mathbb{S}_A(j) = j$) pour $1 \leq j \leq n$, alors $A \leq \llcorner_n$ (resp. $\mathbb{I}_n \leq A \leq \llcorner_n$). ◇

REMARQUE 5.1.3 Soit A une (0,1)-matrice d'ordre n. Alors, $\mathrm{per}(A) = 1$ si et seulement si il existe deux matrices de permutation P et Q d'ordre n telles que $\mathbb{S}_{PAQ^T}(j) = j$ pour $1 \leq j \leq n$. ◇

REMARQUE 5.1.4 Soient A une (0,1)-matrice d'ordre n, P une matrice de permutation d'ordre m et Q une matrice de permutation d'odre n. Alors, $PAQ^T \leq \llcorner_n$ si et seulement si $\mathbb{S}_{PAQ^T}(j) \geq j$ pour $1 \leq j \leq n$. ◇

Étants données deux applications f et g de $[\![n]\!]$ dans $[\![m+1]\!]$, nous notons $f \leq g$ si $f(i) \leq g(i)$ pour $1 \leq i \leq n$. Si A et B sont deux (0,1)-matrices de taille m par n, nous disons que leurs silhouettes sont *comparables* si $\mathbb{S}_A \leq \mathbb{S}_B$ ou $\mathbb{S}_B \leq \mathbb{S}_A$.

Nous nous proposons donc d'étudier dans cette sous-section différentes variantes du problème INF-SILHOUETTE dont l'énoncé est donné ci-après.

PROBLÈME 5.1.5 (INF-SILHOUETTE)
DONNÉES : Une (0,1)-matrice A de taille m par n et une application π de $[\![n]\!]$ dans $[\![m+1]\!]$.
QUESTION : Existe-t-il deux matrices de permutation P d'ordre m et Q d'ordre n telle que $\pi \leq \mathbb{S}_{PAQ^T}$?

DÉFINITION 5.1.6 (SILHOUETTE MONOTONE) *Soit A une (0,1)-matrice d'ordre n. Alors, $\mathbb{S}_A$ est dite* monotone croissante *(resp.* monotone décroissante*) si $\mathbb{S}_A(j) \leq \mathbb{S}_A(j')$ (resp. $\mathbb{S}_A(j) \geq \mathbb{S}_A(j')$) pour $1 \leq j < j' \leq n$. Si $\mathbb{S}_A(j) < \mathbb{S}_A(j')$ (resp. $\mathbb{S}_A(j) > \mathbb{S}_A(j')$) pour $1 \leq j < j' \leq n$, alors $\mathbb{S}_A$ est dite* monotone croissante strictement *(resp.* monotone décroissante strictement*).*

Nous appelons INF-SILHOUETTE MONOTONE CROISSANT STRICT la variante de notre problème pour laquelle π est une application croissante strictement. En utilisant les remarques précédentes, la proposition suivante est immédiate.

PROPOSITION 5.1.7 MATRICE INF-TRIANGULAIRE *et* INF-SILHOUETTE MONOTONE CROISSANT STRICT *restreint à la classe des (0,1)-matrices carrées sont deux problèmes équivalents.*

La monotonie stricte de l'application π est donc l'élément essentiel qui permet l'équivalence entre les deux problèmes. De façon plus générale, nous considérons dans la suite un nouveau paramètre, noté $|\pi|$, qui est le cardinal de l'image de l'application π, *i.e.* :

$$|\pi| = \mathrm{card}(\{\pi(j) \mid 1 \leq j \leq n\}).$$

Nous étudions dans un premier temps la complexité de deux variantes du problème INF-SILHOUETTE lorsque l'égalité $P = Q$ est imposée. Nous revenons ensuite au cas général et montrons qu'il est NP-complet même si A est une matrice avec au plus deux éléments non nuls par ligne.

5.1.3 Isomorphisme

Nous nous intéressons ici au problème INF-SILHOUETTE lorsque l'égalité $P = Q$ est imposée dans le problème. Nous appelons ISO INF-SILHOUETTE cette variante du probléme.

> PROBLÈME 5.1.8 (ISO INF-SILHOUETTE)
> DONNÉES : Une (0,1)-matrice A d'ordre n et une application π de $[\![n]\!]$ dans $[\![n+1]\!]$.
> QUESTION : Existe-t-il une matrice de permutation P d'ordre n telle que $\pi \leq \mathbb{S}_{PAP^T}$?

PROPOSITION 5.1.9 ISO INF-SILHOUETTE *est un problème* NP-*complet même si A est une* (0,1)-*matrice symétrique avec au plus trois éléments non nuls par ligne et par colonne.*

PREUVE

Il est facile de vérifier que le problème appartient à NP. Montrons qu'il est NP-complet. Soit une instance du problème LARGEUR DE BANDE donnée par une (0,1)-matrice symétrique d'ordre n avec au plus trois éléments non nuls par ligne et par colonne, et par un entier positif K. L'instance correspondante du problème ISO INF-SILHOUETTE est donnée (pour la même (0,1)-matrice A) par une application π de $[\![n]\!]$ dans $[\![n+1]\!]$ définie par :

$$\forall j, 1 \leq j \leq n, \qquad \pi(j) = \begin{cases} 1 & \text{si } 1 \leq j \leq K+1 \\ j-K & \text{si } K+2 \leq j \leq n \end{cases}$$

La construction peut s'effectuer en temps polynomial.

Puisque A est une (0,1)-matrice symétrique, nous vérifions facilement qu'il existe une matrice de permutation P d'ordre n telle que $bw(PAP^T) \leq K$ si et seulement si $\pi \leq \mathbb{S}_{PAP^T}$. $\boxed{5.1.9}$

REMARQUE 5.1.10 L'application π définie dans la preuve de la proposition 5.1.9 est monotone croissante. Par suite, ISO INF-SILHOUETTE MONOTONE CROISSANT est un problème NP-complet.

Il n'est bien sur pas étonnant de vérifier que le problème ISO INF-SILHOUETTE est NP-complet. Ce problème très général contient également trivialement le problème COUVERTURE DES SOMMETS pour l'application π de $[\![n]\!]$ dans $[\![m+1]\!]$ définie par :

$$\forall j, 1 \leq j \leq n, \qquad \pi(j) = \begin{cases} n-K+1 & \text{si } 1 < j \leq n-K \\ 1 & \text{si } n-K+1 \leq j \leq n \end{cases}$$

où K est la taille de la couverture des sommets.

Il est par contre plus intéressant de s'interroger sur l'impact du paramètre $|\pi|$ sur la complexité du problème ISO INF SILOUETTE. Pour quelles valeurs du paramètre π le problème est-il résoluble en temps polynomial? Le cas $|\pi| = 1$ est clairement trivial. Le cas $|\pi| = 2$ est NP-complet puisque nous avons remarqué que le problème ISO INF-SILHOUETTE contient le problème COUVERTURE DES SOMMETS. Nous montrons maintenant que le problème ISO INF SILOUETTE est NP-complet lême si $|\pi| = n$.

PROPOSITION 5.1.11 ISO INF-SILHOUETTE *est un problème* NP-*complet même si* $|\pi| = n$.

PREUVE

Soit une instance arbitraire du problème ARRANGEMENT LINÉAIRE TRIANGULAIRE[5] donnée par un graphe G d'ordre n. Soit $A = A(G)$ la matrice d'adjacence de G. L'instance correspondante du problème ISO INF-SILHOUETTE est donnée (pour la même (0,1)-matrice A) par une application π de $[\![n]\!]$ dans $[\![n+1]\!]$ définie par $\pi(j) = n-j+1$ pour $1 \leq j \leq n$. Nous vérifions $|\pi| = n$. De plus, il existe un arrangement triangulaire de G si et seulement si il existe une matrice de permutation P d'ordre n telle que $\pi \leq \mathbb{S}_{\text{PAPT}}$. $\boxed{5.1.11}$

Nous considérons maintenant le cas de la recherche d'une silhouette exacte. Autrement dit, il s'agit de rechercher une permutation simultannée des lignes et des colonnes de la matrice A de façon à ce que $\pi(j)$ pour $1 \leq j \leq n$ coïncide exactement avec le hauteur minimale d'un élément non nul de la colonne j. Nous appelons ISO SILHOUETTE ce problème.

PROBLÈME 5.1.12 (ISO SILHOUETTE)
DONNÉES : Une (0,1)-matrice A d'ordre n, et une application π de $[\![n]\!]$ dans $[\![n+1]\!]$.
QUESTION : Existe-t-il une matrice de permutation P d'ordre n telle que $\pi = \mathbb{S}_{\text{PAPT}}$?

Remarquons tout d'abord que les propositions 5.1.9 et 5.1.11 ne nous permettent pas de conclure quant à la complexité de ce nouveau problème. Cependant, le problème ISO SILHOUETTE ets NP-complet et nous donnons une réduction depuis le problème x3c.

Pour tout rationnel r, nous notons $\lceil r \rceil$ le plus petit entier supérieur ou égal à r.

PROPOSITION 5.1.13 ISO SILHOUETTE *est un problème* NP-*complet.*

PREUVE

Il est facile de vérifier que le problème ISO SILHOUETTE appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème x3c donnée par une famille $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$, $|S_i| = 3$ pour $1 \leq i \leq m$, de sous-ensembles d'un $3q$-ensemble X. Sans perte de généralité, nous pouvons supposer que chaque élément de

5. problème 4.4.6 page 74.

X apparaît dans au plus trois sous-ensembles de $\mathcal{F}$ [GJ79]. L'instance correspondante du problème ISO SILHOUETTE est donnée par une (0,1)-matrice $A = [a_{i,j}]$ d'ordre $m + 3q$ et par une application π de $[\![m + 3q]\!]$ dans $[\![m + 3q + 1]\!]$. La (0,1)-matrice A est définie par :

$$\forall i,\ 1 \leq i \leq m, \quad a_{i,j} = \begin{cases} 1 & \text{si } m+1 \leq j \leq m+3q \text{ et } x_{j-m} \in S_i \\ 0 & \text{sinon} \end{cases}$$

$$\forall i,\ m+1 \leq i \leq m+3q, \quad a_{i,j} = 1 \quad \text{pour} \quad 1 \leq j \leq m+3q.$$

Remarquons que la (0,1)-matrice A est de la forme :

$$A = \begin{bmatrix} \mathbf{0}_{m,m} & A' \\ \mathbf{1}_{3q,m} & \mathbf{1}_{3q,3q} \end{bmatrix}$$

où A' est une (0,1)-matrice de taille m par $3q$ avec exactement trois 1 par ligne et au plus trois 1 par colonne. L'application π de l'instance est définie par :

$$\pi(j) = \begin{cases} m+1 & \text{pour } 1 \leq j \leq m \\ \left\lceil \dfrac{j-m}{3} \right\rceil & \text{pour } m+1 \leq j \leq m+3q. \end{cases}$$

La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de q sous-ensembles disjoints. Soit $\{S_{i_1}, S_{i_2}, \ldots, S_{i_q}\}$ cet ensemble. Raisonnons dans un premier temps sur la sous-matrice A'. Soit P la matrice de permutation d'ordre m qui place les lignes indicées par $\{i_1, i_2, \ldots, i_q\}$ de la matrice A sur les q premières lignes; les autres lignes étant placées arbitrairement. Pour tout $1 \leq i \leq q$, notons j_i, j_i' et j_i'' les indices des colonnes des éléments non nuls de la ligne i dans la matrice PA'. Soit Q^T la matrice de permutation d'ordre $3q$ qui place les colonnes j_i, j_i' et j_i'' de la matrice PA' aux positions $3i-2$, $3i-1$ et $3i$. Alors, les q premières lignes de la matrice $PA'Q^T$ sont de la forme :

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \ldots & 0 & 0 & 0 \\ & \vdots & & & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 1 & 1 & 1 \end{bmatrix}.$$

Intuitivement, les lignes correspondantes aux sous-ensembles de $\mathcal{F}'$ sont placées sur q premières lignes et les éléments non nuls de ces q premières lignes apparaissent consécutivement et de gauche à droite. Remarquons maintenant que :

$$\mathbb{S}_{PA'Q^T}(j) = \left\lceil \frac{j}{3} \right\rceil$$

pour $1 \leq j \leq 3q$. Notons R la matrice de permutation d'ordre $m + 3q$ définie par :

$$R = \begin{bmatrix} P & \mathbf{0}_{m,3q} \\ \mathbf{0}_{3q,m} & Q \end{bmatrix}.$$

Alors, nous avons :

$$\begin{aligned} RAR^T &= \begin{bmatrix} P & \mathbf{0}_{m,3q} \\ \mathbf{0}_{3q,m} & Q \end{bmatrix} \begin{bmatrix} \mathbf{0}_{m,m} & A' \\ \mathbf{1}_{3q,m} & \mathbf{1}_{3q,3q} \end{bmatrix} \begin{bmatrix} P^T & \mathbf{0}_{m,3q} \\ \mathbf{0}_{3q,m} & Q^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}_{m,m} & PA'Q^T \\ \mathbf{1}_{3q,m} & \mathbf{1}_{3q,3q} \end{bmatrix}. \end{aligned}$$

Dans un premier temps, nous avons $\mathbb{S}_{\mathrm{RAR^T}}(j) = m+1$ pour $1 \leq j \leq m$. De plus, $\mathbb{S}_{\mathrm{RAR^T}} = \left\lceil \frac{j-m}{3} \right\rceil$ pour $m+1 \leq j \leq m+3q$ car $\mathbb{S}_{\mathrm{PA'Q^T}}(j) = \left\lceil \frac{j}{3} \right\rceil$ pour $1 \leq j \leq 3q$. Par conséquent, nous vérifions $\pi = \mathbb{S}_{\mathrm{RAR^T}}$.

Inversement supposons qu'il existe une matrice de permutation R d'ordre $m+3q$ telle que $\pi = \mathbb{S}_{\mathrm{RAR^T}}$. Notons α l'ensemble des lignes de la matrice A indicées de $m+1$ à $m+3q$. Remarquons alors que toutes les lignes de α contiennent exactement $m+3q$ éléments non nuls. Aussi, puisque $\pi(j) = m+1$ pour $1 \leq j \leq m$, les lignes $m+1$ à $m+3q$ de la matrice RAR^T sont également indicées par les lignes de l'ensemble α (à une permutation près de). Autrement dit, la matrice de permutation R est de la forme :

$$R = \begin{bmatrix} P & \mathbf{0}_{m,3q} \\ \mathbf{0}_{3q,m} & Q \end{bmatrix}.$$

Par suite,

$$\begin{aligned} RAR^T &= \begin{bmatrix} P & \mathbf{0}_{m,3q} \\ \mathbf{0}_{3q,m} & Q \end{bmatrix} \begin{bmatrix} \mathbf{0}_{m,m} & A' \\ \mathbf{1}_{3q,m} & \mathbf{1}_{3q,3q} \end{bmatrix} \begin{bmatrix} P^T & \mathbf{0}_{m,3q} \\ \mathbf{0}_{3q,m} & Q^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}_{m,m} & PA'Q^T \\ \mathbf{1}_{3q,m} & \mathbf{1}_{3q,3q} \end{bmatrix}. \end{aligned}$$

Nous en déduisons que :

$$\mathbb{S}_{\mathrm{PA'Q^T}}(j) = \left\lceil \frac{j}{3} \right\rceil$$

pour $1 \leq j \leq 3q$. Mais puisque toutes les lignes de la matrice A' contiennent exactement trois éléments non nuls, nous en concluons qu'il existe q sous-ensembles disjoints de $\mathcal{F}$. $\boxed{5.1.13}$

Nous ignorons si ISO SILHOUETTE reste un problème NP-complet si le nombre d'éléments non nuls sur chaque ligne et / ou colonne est borné par une constante. De plus, nous ignorons également la complexité du problème restreint à la classe des (0,1)-matrices symétriques.

5.1.4 Cas général

Nous revenons maintenant au cas général, *i.e.* Étant données une (0,1)-matrice A de taille m par n et une application π de $[\![n]\!]$ dans $[\![m+1]\!]$, existe-t-il des matrices de permutation P d'ordre m et Q d'ordre n telles que $\pi \leq \mathbb{S}_{\mathrm{PAQ^T}}$? Nous allons montrer que ce problème est NP-complet.

LEMME 5.1.14 *Soient A une (0,1)-matrice de taille m par n avec au plus deux 1 par ligne, et J et L deux entiers positifs. Décider s'il existe des matrices de permutation P d'ordre m et Q d'ordre n telles que:*

$$PAQ^T \leq \begin{bmatrix} \mathbf{1}_{J,L} & \mathbf{0}_{J,n-L} \\ \mathbf{1}_{m-J,L} & \mathbf{1}_{m-J,n-L} \end{bmatrix}$$

est un problème NP*-complet.*

PREUVE

Il est facile de vérifier que le problème appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème CLIQUE donnée par un graphe $G = (V,E)$ d'ordre n et par un entier positif K. L'instance correspondante du problème considéré est donnée par une (0,1)-matrice $A = [a_{i,j}]$ et par les entiers positifs J et L définis par:

- A est la matrice d'incidence arêtes-sommets de G: chaque ligne correspond à une arête et chaque colonne à un sommet; il existe exactement deux 1 par ligne qui correspondent aux sommets extrémités de l'arête associée à la ligne.
- $J = \frac{1}{2}K(K-1)$
- $L = K$

La construction peut s'effectuer en temps polynomial.

Il existe une clique de taille K dans G si et seulement si il existe deux matrices de permutation P d'ordre m et Q d'ordre n telles que:

$$PAQ^T \leq \begin{bmatrix} \mathbf{1}_{J,L} & \mathbf{0}_{J,n-L} \\ \mathbf{1}_{m-J,L} & \mathbf{1}_{m-J,n-L} \end{bmatrix}.$$

En effet, puisque les lignes de A sont toutes différentes, la sous-matrice $\begin{bmatrix} \mathbf{1}_{J,L} & \mathbf{0}_{J,n-L} \end{bmatrix}$ dénote un ensemble de $J = \frac{1}{2}K(K-1)$ arêtes incidentes à $L = K$ sommets. $\boxed{5.1.14}$

PROPOSITION 5.1.15 INF-SILHOUETTE *est un problème* NP*-complet même si A est une matrice avec au plus deux éléments non nuls par ligne.*

PREUVE

Soit une instance arbitraire du problème décrit en 5.1.14. L'instance correspondante du problème INF-SILHOUETTE est donnée (pour A) par une application π de $[\![n]\!]$ dans $[\![m+1]\!]$ définie par:

$$\forall j, 1 \leq j \leq n, \qquad \pi(j) = \begin{cases} 1 & \text{si } 1 \leq j \leq L \\ J+1 & \text{si } L+1 \leq j \leq n \end{cases}$$

Les deux problèmes sont équivalents et le lemme 5.1.14 permet de conclure. $\boxed{5.1.15}$

De même que dans le cas $P = Q$, nous considérons maintenant le problème de la recherche exacte d'une silhouette donnée par permutation des lignes et des colonnes. Nous appelons SILHOUETTE ce problème dont voici l'énoncé.

PROBLÈME 5.1.16 (SILHOUETTE)
DONNÉES : Une (0,1)-matrice A de taille m par n et une application π de $[\![n]\!]$ dans $[\![m+1]\!]$.
QUESTION : Existe-t-il deux matrices de permutation P d'ordre m et Q d'ordre n telles que $\pi = \mathbb{S}_{PAQ^T}$?

Nous allons montrer qu'à l'instar du problème ISO SILHOUETTE, ce problème est également difficile. Nous utilisons pour cela la preuve de la proposition 5.1.13.

PROPOSITION 5.1.17 SILHOUETTE *est un problème* NP*-complet même si A a au plus trois 1 par ligne et par colonne.*

PREUVE
Nous reprenons une partie de la preuve de la proposition 5.1.13. Plus précisément, nous ne considérons que la matrice A' de taille m par $3q$. Soit π l'application de $[\![3q]\!]$ dans $[\![m+1]\!]$ définie par $\pi(j) = \left\lceil \frac{j}{3} \right\rceil$ pour $1 \leq j \leq 3$. Alors, il existe q sous-ensembles disjoints de $\mathcal{F}$ si et seulement si il existe deux matrices de permutation P d'ordre m et Q d'ordre $3q$ telles que $\pi = \mathbb{S}_{PA'Q^T}$. $\boxed{5.1.17}$

REMARQUE 5.1.18 Si A est une matrice carrée d'ordre n et π est une *permutation* de $[\![n]\!]$, alors le problème SILHOUETTE est résoluble en temps $O(n^2)$. En effet, dans ce cas particulier, le problème est équivalent à $\mathbb{I}_n \leq PAQ^T \leq \mathbb{L}_n$. ◇

Nous avons discuté dans la sous-section précédente de l'influence du paramètre $|\pi|$ sur la complexité du problème ISO INF-SILHOUETTE: le problème est résoluble en temps polynomial si $|\pi| = 1$ et NP-complet pour $|\pi| = 2$ et $|\pi| = n$. Quelle est l'influence de ce paramètre sur la complexité du problème dans le cas présent? Cette question est en effet fondamentale puisque nous avons montré que le problème INF-SILHOUETTE est équivalent au problème $PAQ^T \leq \mathbb{L}_n$ si A est une matrice carrée d'ordre n et π est une permutation de $[\![n]\!]$, *i.e.* $|\pi| = n$.

Dans un premier temps, nous remarquons que le problème INF-SILHOUETTE est NP-complet même si $|\pi| = c$, où c est une constante indépendante de l'instance. En effet, dans la preuve de la proposition 5.1.15, nous avons $c = 2$. D'un autre coté, nous sommes une nouvelle fois dans l'incapacité de montrer que le problème est NP-complet ou résoluble en temps polynomial dans le cas $|\pi| = n$. D'une façon plus générale, nous ignorons la complexité du problème INF-SILHOUETTE lorsque $|\pi| = n - c$, où c est une constante indépendante de l'instance.

5.2 Surface

5.2.1 Introduction

De nombreux formats de compression pour les matrices creuses ont été proposés. Par exemple, la *représentation par lignes compressées* (CRS) utilise trois tableaux : `val`, `col_ind` et `row_ptr`. Le tableau `val` contient les éléments non nuls de la matrice

A dans l'ordre des lignes. Le tableau `col_ind` contient les indices des colonnes des éléments du tableau `val` : si `val`$[k] = a_{i,j}$, alors `col_ind`$[k] = j$. Enfin, le tableau `row_ptr` contient les positions de début de ligne dans le tableau `val` : si `val`$[K] = a_{i,j}$, alors `row_ptr` $\leq k \leq$ `row_ptr`$[i+1]$. En notant m le nombre d'éléments non nuls de la matrice A d'ordre n, par convention, `row_ptr`$[n+1] = m+1$. Ainsi, si A est une matrice non symétrique définie par :

$$A = \begin{bmatrix} a & 0 & 0 & b \\ 0 & c & d & 0 \\ 0 & e & f & g \\ h & 0 & 0 & 0 \end{bmatrix}$$

Alors,

val	a	b	c	d	e	f	g	h
col_ind	1	4	2	3	2	3	4	1

et

row_ptr	1	3	5	8	9

Le gain en espace est significatif, puisque $2m+n+1$ éléments suffisent à représenter la matrice. Remarquons que l'accès à un élément $a_{i,j}$ de la matrice A n'est plus en temps constant. La *représentation par colonnes compressées* (CCS ou Harwell-Boeing) utilise une description identique au format CRS mais en représentant les colonnes. Lorsque la matrice est à *largeur de bande limitée*, il est possible de tirer avantage de cette structure en représentant les sous-diagonales dans un même tableau[6].

La mise sous forme triangulaire d'une (0,1)-matrice est déjà une première forme de compression. En effet, au plus la moitié des éléments est non nul. Nous pouvons stocker les éléments $a_{i,j}$, $1 \leq j \leq i \leq n$, dans un tableau T en parcourant la matrice ligne par ligne. Par exemple, si A est la (0,1)-matrice définie par :

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

alors,

$$T = \boxed{1}\boxed{0\ 1}\boxed{1\ 1\ 0}\boxed{1\ 0\ 1\ 1}\boxed{0\ 1\ 1\ 1\ 1}$$

Il est facile de vérifier que $T[\frac{1}{2}i(i-1)+j] = a_{i,j}$. L'accès aux éléments de la matrice s'effectue donc en temps constant. Cependant, l'espace occupé par ce codage reste quadratique.

Nous allons tout au long de cette section envisager notre problème de référence MATRICE INF-TRIANGULAIRE[7] comme un problème de compression d'une (0,1)-matrice.

6. Rappelons que le probléeme de décision associé à la minimisation de la largeur de bande d'une (0,1)-matrice est NP-complet [Pap76, GGJK78].

7. problème 4.1.1 page 58.

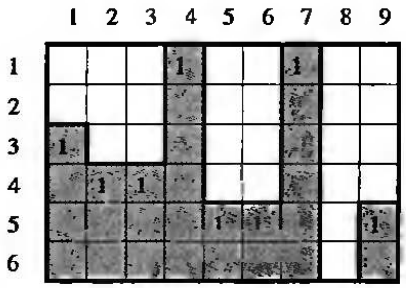


FIG. 5.2 – *Soit A la* (0,1)*-matrice ci-dessus (par souci de clarté, seuls les premiers 1 de chaque colonne sont représentés). Alors,* $|\mathbb{S}_A| = 9 \times (6+1) - (3+4+4+1+5+5+1+7+5) = 28$ *car* $m = 6$, $n = 9$ *et* $\mathbb{S}_A(1) = 3$, $\mathbb{S}_A(2) = 4$, $\mathbb{S}_A(3) = 4$, $\mathbb{S}_A(4) = 1$, $\mathbb{S}_A(5) = 5$, $\mathbb{S}_A(6) = 5$, $\mathbb{S}_A(7) = 1$, $\mathbb{S}_A(8) = 7$ *et* $\mathbb{S}_A(9) = 5$.

Nous introduisons dans un premier temps quelques définitions nécessaires à notre étude.

De même que le profil d'une matrice est une «*mesure*» de son enveloppe[8], nous associons à la silhouette d'une matrice une «*mesure*» que nous appelons *surface*. Rappelons que nous notons $\mathbb{S}_A$ la *silhouette* de la (0,1)-matrice A.

DÉFINITION 5.2.1 (SURFACE) *Soit $A = [a_{i,j}]$ une* (0,1)*-matrice de taille m par n. La* surface *de A, notée $|\mathbb{S}_A|$, est définie par:* ***surface***

$$|\mathbb{S}_A| = n(m+1) - \sum_{1 \leq j \leq m} \mathbb{S}_A(j)$$

Intuitivement, la *surface* d'une (0,1)-matrice A consiste donc à considérer la (0,1)-matrice A comme un tableau de taille m par n et à compter pour chaque colonne le nombre de lignes à partir du premier 1 (figure 5.2). Remarquons que si une colonne j ne contient que des 0, alors cette colonne contribue pour 0 à la surface de la matrice A. En fait, si nous considérons la silhouette d'une (0,1)-matrice A comme une ligne de crête, alors la surface associée à cette silhouette correspond à l'espace inférieur de la (0,1)-matrice.

Nous commençons par une remarque triviale - mais fondamentale - pour la suite de notre étude.

REMARQUE 5.2.2 La surface d'une (0,1)-matrice est invariante par permutation des colonnes. Autrement dit, soit A une (0,1)-matrice de taille m par n, alors

$$|\mathbb{S}_A| = |\mathbb{S}_{AQ^T}|$$

pour toute matrice de permutation Q d'ordre n. ◇

Puisque la surface d'une (0,1)-matrice est invariante par permutation des colonnes, nous pouvons définir deux *formes standards* pour la représentation des matrices. Si A est une (0,1)-matrice de taille m par n, nous disons que A est sous *forme* ***forme standa***

8. c.f. sous-section 5.1.1.

standard croissante (resp. *forme standard décroissante*) si $\mathbb{S}_A$ est monotone croissante (resp. monotone décroissante). Autrement dit, une matrice A est sous forme standard croissante (resp. décroissante) si ses colonnes sont triées par ordre croissant (resp. décroissant) de leur premier élément non nul. Remarquons que les colonnes qui ne contiennent que des 0 sont placées en dernières (resp. premières) positions lorsque la matrice est sous forme standard croissante (resp. décroissante).

Par exemple, sur les (0,1)-matrices données ci-dessous, A est sous forme standard croissante, tandis que B est sous forme standard décroissante :

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Nous pouvons maintenant établir un premier lien avec le problème de la triangularisation d'une (0,1)-matrice par permutation des lignes et des colonnes.

REMARQUE 5.2.3 Soit A une (0,1)-matrice d'ordre n. S'il existe des matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$, alors $|\mathbb{S}_{PAQ^T}| \leq \frac{1}{2}n(n+1)$. De plus, puisque la silhouette d'une (0,1)-matrice est invariante par permutation des colonnes, alors $PAQ^T \leq \triangle_n$ implique $|\mathbb{S}_{PA}| \leq \frac{1}{2}n(n+1)$. La réciproque n'est pas vraie. En effet, considérons la (0,1)-matrice A d'ordre 4 définie par :

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Alors, nous avons $|\mathbb{S}_A| = 10 \leq \frac{1}{2}4(4+1)$, mais il n'existe pas des matrices de permutation P et Q d'ordre 4 telles que $PAQ^T \leq \triangle_4$. ◇

Nous nous proposons dans cette section d'examiner le problème consistant à minimiser la surface d'une (0,1)-matrice par permutation des lignes. Nous montrerons dans la suite (sous-sections 5.2.3) que ce problème est NP-complet dans le cas général. Nous préciserons également que le problème consistant à maximiser cette surface est NP-complet.

5.2.2 Résultats expérimentaux

Nous présentons dans un premier temps quelques résultats expérimentaux sur les surfaces des (0,1)-matrices aléatoires.

Soient n et k, $0 \leq k \leq n^2$, deux entiers positifs. Nous notons $\mathcal{A}_k$ l'ensemble des (0,1)-matrices d'ordre n comportant exactement k éléments non nuls. Clairement, nous avons :

$$|\mathcal{A}_k| = \binom{n^2}{k}.$$

Nous utilisons l'algorithme avec rejet[9] (0,1)-*matrice-aléatoire* pour générer aléatoirement et équiprobablement des éléments de $\mathcal{A}_k$. Remarquons que si $k > \frac{n}{2}$, il est préférable d'utiliser l'algorithme pour placer les 0 plutôt que les 1.

Algorithm 7: Génération aléatoire et équiprobable d'une (0,1)-matrice contenant exactement k éléments non nuls.

(0,1)-*matrice-aléatoire*

Entrée: Les entiers positifs n et k avec $0 \leq k \leq n^2$.

Sortie: Une (0,1)-matrice A d'ordre n contenant exactement k éléments non nuls placés aléatoirement et équiprobablement.

```
début
  Soit A = 0_{n,n} une (0,1)-matrice
  ℓ ← 0
  tant que ℓ < k faire
    Tirer au sort deux entiers i et j avec 1 ≤ i ≤ m et 1 ≤ j ≤ n
    si a_{i,j} = 0 alors
      a_{i,j} ← 1
      ℓ ← ℓ + 1
fin
```

La figure 5.3 présente des mesures expérimentales de surface de (0,1)-matrices aléatoires pour différentes valeurs des paramètres n et k. Nous vérifions sur la figure que :

$$|\mathbb{S}_A| = \begin{cases} 0 & \text{si } k = 0 \\ n^2 & \text{si } k = n^2. \end{cases}$$

La figure 5.4 donne des résultats expérimentaux pour les surfaces moyennes obtenues par permutation des lignes sur des (0,1)-matrices d'ordre 8. La courbe inférieure (resp. supérieure) donne la surface minimale (resp. maximale) moyenne obtenue par permutation des lignes. La courbe centrale indique la surface moyenne sur toutes les permutations des lignes.

5.2.3 Minimisation de la surface d'une (0,1)-matrice

Nous montrons ici que le problème de décision associé à la minimisation d'une (0,1)-matrice par permutation des lignes est un problème NP-complet. Nous appelons MIN-SURFACE le problème de décision associé dont voici l'énoncé.

PROBLÈME 5.2.4 (MIN-SURFACE)
DONNÉES: Une (0,1)-matrice A de taille m par n et un entier positif K.
QUESTION: Existe-t-il une matrice de permutation P d'ordre m telle que $|\mathbb{S}_{PA}| \leq K$?

9. Les algorithmes avec rejet présentent en particulier l'avantage d'être relativement rapides et faciles à mettre en œuvre.

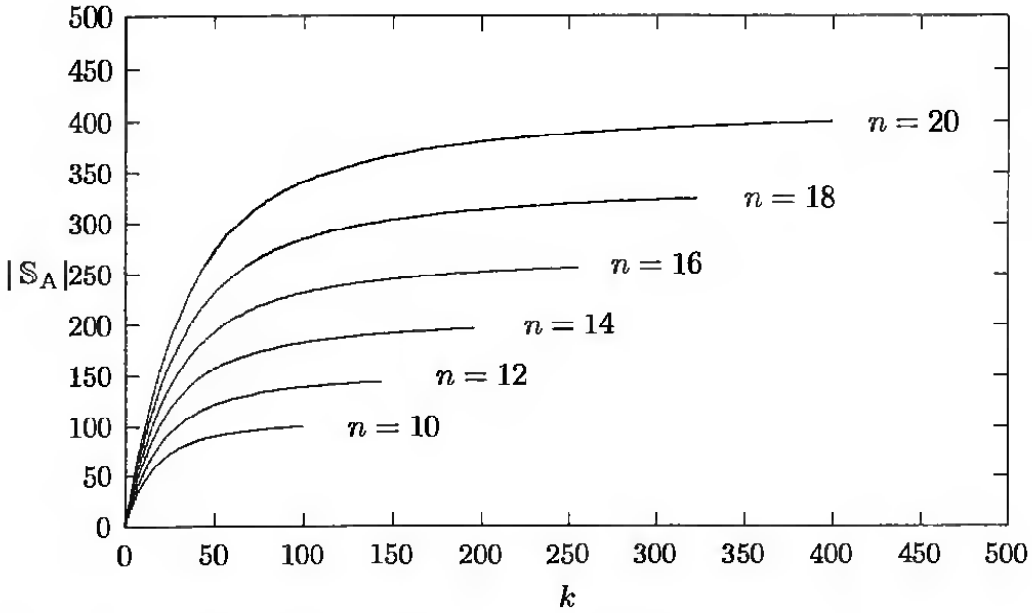


FIG. 5.3 – *Résultats expérimentaux pour la surface de* (0,1)*-matrices aléatoires de taille* n*. Le paramètre* k *est le nombre d'éléments non nuls de la matrice* ($0 \leq k \leq n^2$)*. Les courbes sont obtenues en générant 1000 matrices aléatoires pour chaque valeur du couple* (n,k)*,* $n \in \{10,12,14,16,18,20\}$ *et* $0 \leq k \leq n^2$*.*

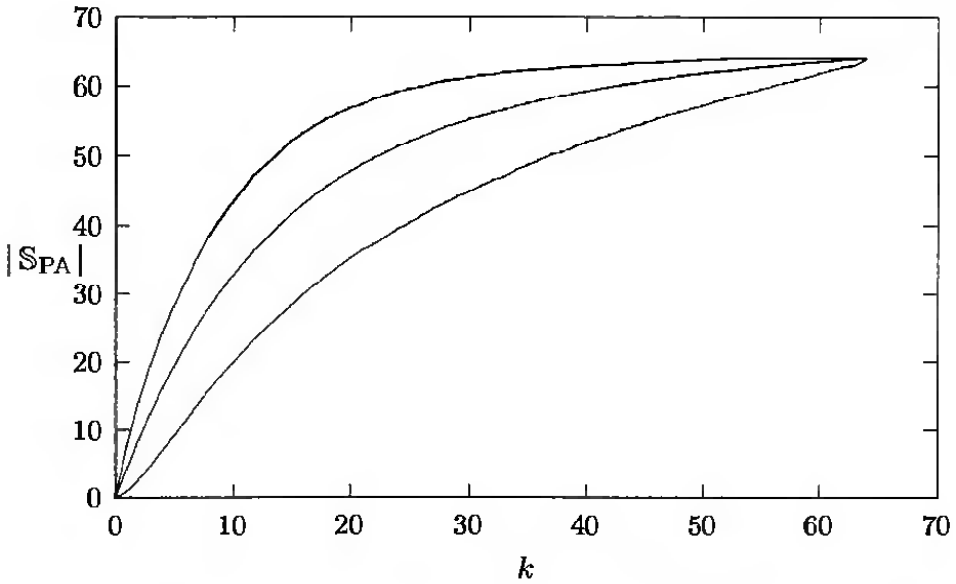


FIG. 5.4 – *Résultats expérimentaux pour les surfaces maximales, moyennes et minimales obtenues par permutation exhaustives des lignes sur des* (0,1)*-matrices d'ordre 8. Le paramètre k est le nombre d'éléments non nuls de la matrice. Les courbes sont obtenues en générant 1000 matrices aléatoires pour chaque valeur du paramètre k, $0 \leq 0 \leq 8^2$. Pour chaque* (0,1)*-matrice ainsi obtenue, nous générons les* 8! *permutations [Sed77] des lignes. La courbe inférieure (resp. supérieure) représente la surface minimale (resp. maximale) moyenne pour chaque valeur de k. La courbe centrale représente la surface moyenne pour chaque valeur de k.*

Avant de montrer que ce problème est NP-complet dans le cas général, nous prouvons qu'il est résoluble en temps constant dans le cas des (0,1)-matrices carrées dont le permanent est exactement 1.

PROPOSITION 5.2.5 *Soit A une (0,1)-matrice d'ordre n. Si A vérifie* $\mathrm{per}(A) = 1$, *alors le problème* MIN-SURFACE *est résoluble en temps constant pour A.*

PREUVE

En utilisant le théorème 4.3.4, nous avons $\mathrm{per}(A) = 1$ si et seulement si il existe deux matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \llcorner_n$ et la matrice PAQ^T a des 1 sur tous les éléments de sa diagonale principale, *i.e.* $\mathbb{I}_n \leq PAQ^T \leq \llcorner_n$. Alors, nous avons :

$$\min\{|\mathbb{S}_{PA}| \mid P \text{ permutation d'ordre } n\} = \frac{1}{2}n(n+1).$$

Par suite, la réponse au problème MIN-SURFACE est «*oui*» si $K \geq \frac{1}{2}n(n+1)$, et «*non*» si $K < \frac{1}{2}n(n+1)$. $\boxed{5.2.5}$

Nous allons maintenant montrer que MIN-SURFACE est un problème NP-complet dans le cas général. Nous considérons dans un premier temps trois nouveaux problèmes :

1. ARRANGEMENT LINÉAIRE MAX/MAX
2. ARRANGEMENT LINÉAIRE MAX/MIN
3. ARRANGEMENT LINÉAIRE MIN/MAX

et montrons qu'il sont tous NP-complets. La NP-complétude du problème MIN-SURFACE sera ensuite une simple conséquence de ces résultats.

Nous utilisons COUVERTURE DES SOMMETS comme problème de départ pour nos résultats. Nous rappelons tout d'abord l'énoncé de ce problème.

PROBLÈME 5.2.6 (COUVERTURE DES SOMMETS)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n et un entier positif K.
QUESTION : Existe-t-il un sous-ensemble $V' \subseteq V$, $|V'| \leq K$, tel que $u \in V'$ ou $v \in V'$ pour tout $\{u,v\} \in E$?

Remarquons que si V' est une couverture des sommets dans G, alors $V \setminus V'$ est un ensemble stable dans G. Il est possible de montrer que COUVERTURE DES SOMMETS est un problème NP-complet par une transformation polynomiale depuis 3SAT ([GJ79] en donne une preuve détaillée). Une réduction polynomiale depuis le problème CLIQUE est donnée dans [CLR92]. Le problème COUVERTURE DES SOMMETS est résoluble en temps polynomial pour les graphes parfaits (en particulier pour les graphes bipartis) [GJ79, Jun99].

Nous allons utiliser dans la suite une version restreinte du problème COUVERTURE DES SOMMETS. Rappelons que le *degré minimum* des sommets d'un graphe G est noté $\delta(G)$ et le *degré maximum* $\Delta(G)$. Un sommet de degré 0 est appelé un sommet *isolé*. Si $\delta(G) = \Delta(G)$, c'est-à-dire si tous les sommets de G ont pour dégré *exactement* k,

alors le graphe G est *régulier de degré k* ou plus simplement *k-régulier*. Dans le cas particulier où tous les sommets de G ont pour degré exactement 3, c'est-à-dire si G est un graphe 3-régulier, G est appelé un *graphe cubique*. ***cubique***

Un graphe est *planaire* s'il est possible de représenter le graphe dans le plan de sorte que les sommets correspondent à des points distincts, et les arêtes à des courbes de Jordan entre les points extrémités [Bol79]. ***planaire***

THÉORÈME 5.2.7 ([GJ79]) COUVERTURE DES SOMMETS *est un problème* NP*-complet même si G est un graphe planaire cubique.*

Montrer qu'un problème sur les graphes est NP-complet même si G est planaire est en général une restriction très intéressante. En effet, la *planarité* d'un graphe se retrouve souvent dans des problèmes pratiques, par exemple pour le routage des circuits intégrés. En particulier, le problème CIRCUIT HAMILTONIEN est NP-complet même si G est un graphe planaire cubique [GJT76][10].

Nous revenons maintenant à notre problème initial. Notre première réduction concerne un nouveau problème que nous appelons ARRANGEMENT LINÉAIRE MAX/MAX et qui s'énonce de la façon suivante.

PROBLÈME 5.2.8 (ARRANGEMENT LINÉAIRE MAX/MAX)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n et un entier positif J.
QUESTION : Existe-t-il une bijection ϕ de V dans $[\![n]\!]$ telle que

$$\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} \geq J \ ?$$

Il s'agit donc de maximiser la somme des sommets maximaux (par la bijection ϕ) de chaque arête. Dans ARRANGEMENT LINÉAIRE MAX/MAX, le premier MAX désigne donc ce qui est comptabilisé dans le graphe, à savoir le sommet maximum de chaque arête, tandis que le second MAX désigne l'optimisation désirée, à savoir ici une somme maximale.

PROPOSITION 5.2.9 ARRANGEMENT LINÉAIRE MAX/MAX *est un problème* NP*-complet même si G est un graphe planaire et $\Delta(G) \leq 3$.*

PREUVE

Le problème ARRANGEMENT LINÉAIRE MAX/MAX appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème COUVERTURE DES SOMMETS

10. En 1880, Tait avait conjecturé que tout graphe planaire cubique triples connectés (la suppression de deux sommets ne déconnecte pas le graphe) possédait un circuit hamiltonien. En 1946, Tutte a montré que cette conjecture était fausse en construisant le premier contre-exemple connu. Dans [GJT76], le problème CHEMIN HAMILTONIEN est montré NP-complet sous ces mêmes hypothèses. Autrement dit, non seulement la conjecture de Tait est fausse et il existe des graphes planaires cubiques triple connecté qui ne contiennent pas un chemin hamiltonien, mais il est probablement impossible de donner un algorithme efficace pour décider sous ces hypothèses si un graphe contient ou non un chemin hamiltonien.

donnée par un graphe $G_1 = (V_1,E_1)$ planaire cubique d'ordre n et par un entier positif K. Soit $p = 6n(K-1)$ un entier positif. Remarquons que p est un entier positif pair. Soient V_2 un ensemble de p nouveaux sommets :

$$V_2 = \{u_1,u_2,\ldots,u_p\}$$

et $E_2 \subset V_2 \times V_2$ un ensemble d'arêtes défini par :

$$E_2 = \{\{u_i,u_j\} \mid i+j = p+1\}.$$

Puisque p est pair, $|E_2| = \frac{1}{2}p$ et tous les sommets de V_2 ont pour degré exactement 1. L'instance correspondante du problème ARRANGEMENT LINÉAIRE MAX/MAX est donnée par un graphe $G^* = (V^*,E^*)$ et par un entier positif J définis par :

- $V^* = V_1 \cup V_2$;
- $E^* = E_1 \cup E_2$;
- $J = \frac{1}{4}p(\frac{3}{2}p+1) + \frac{1}{2}p(n-K) + \frac{3}{2}n(n-K+p+1)$.

Remarquons que $|V'| = n+p$ et $|E'| = \frac{3}{2}n + \frac{1}{2}p$. Nous vérifions facilement que le graphe G^* est planaire. De plus, tous les sommets de G^* ont pour degré au plus 3. La construction peut s'effectuer en temps polynomial.

Par souci de clarté, dans la suite, nous notons

$$\omega_i(\phi) = \sum_{\{u,v\}\in E_i} \max\{\phi(u),\phi(v)\}$$

pour $1 \leq i \leq 2$ et

$$\omega(\phi) = \omega_1(\phi) + \omega_2(\phi)$$

Supposons qu'il existe dans G_1 une couverture des sommets de taille K, c'est-à-dire un sous-ensemble $V' \subseteq V_1$ tel que $u \in V'$ ou $v \in V'$ pour tout $\{u,v\} \in E$. Considérons une bijection ϕ de V^* dans $[\![n+p]\!]$ définie par :

$$\begin{cases} 1 \leq \phi(u) \leq n-K & \text{pour } u \in V_1 \ \wedge \ u \notin V' \\ n-K+1 \leq \phi(u_i) \leq n-K+\frac{1}{2}p & \text{pour } u \in V_2 \ \wedge \ 1 \leq i \leq \frac{1}{2}p \\ n-K+\frac{1}{2}p+1 \leq \phi(u_i) \leq n-K+p & \text{pour } u \in V_2 \ \wedge \ \frac{1}{2}p+1 \leq i \leq p \\ n-K+p+1 \leq \phi(u) \leq n+p & \text{pour } u \in V_1 \ \wedge \ u \in V'. \end{cases}$$

Dans un premier temps, puisque V' est une couverture des sommets de taille K dans G_1, il vient :

$$\max\{\phi(u),\phi(v)\} \geq n-K+p+1 \quad \text{pour tout } \{u,v\} \in E_1$$

et par suite :

$$\omega_1(\phi) \geq \frac{3}{2}n(n-K+p+1)$$

car G_1 est un graphe cubique et contient donc exactement $\frac{3}{2}n$ arêtes. Ensuite, nous avons :

$$\omega_2(\phi) = \frac{1}{4}p(\frac{3}{2}p+1) + \frac{1}{2}p(n-K).$$

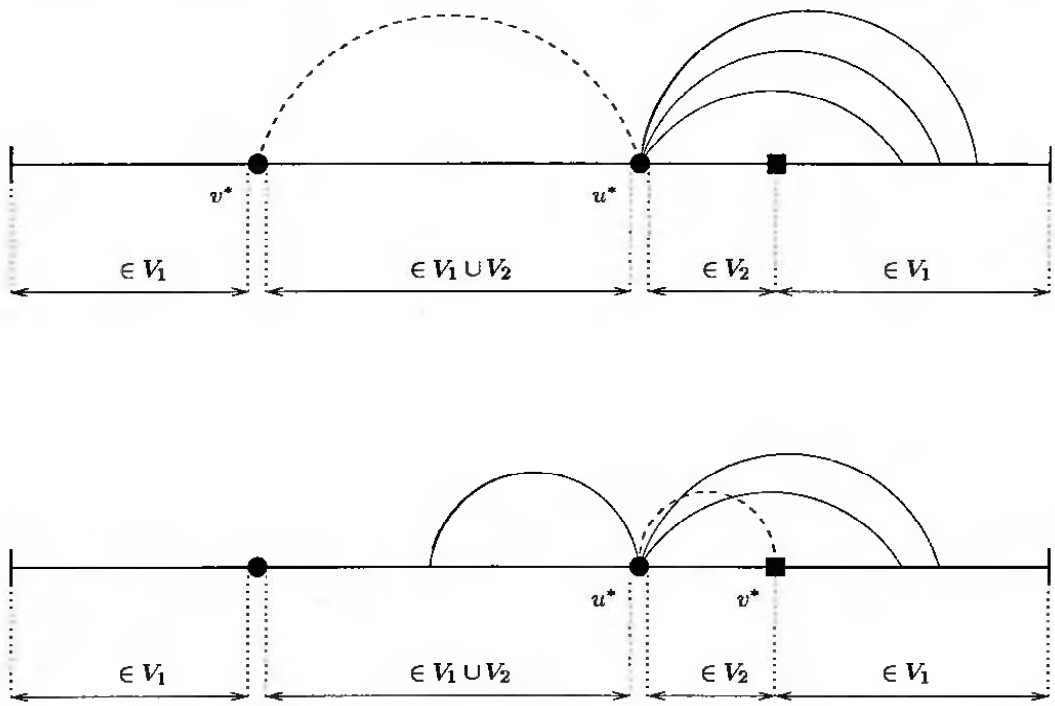


FIG. 5.5 – *Les cas $\bar{\phi}(u^*) < \bar{\phi}(u)$ pour tout $u \in N(u^*)$ et $\bar{\phi}(u^*) > \bar{\phi}(u)$ pour au moins un sommet $u \in N(u^*)$.*

Le premier terme compte les sommets maximaux de chaque arête de l'ensemble E_2, tandis que le second terme prend en compte le décalage induit par les $n-K$ sommets du sous-ensemble $V_1 \setminus V'$. En conséquence, nous vérifions :

$$\begin{aligned}\omega(\phi) &= \omega_1(\phi) + \omega_2(\phi)\\ &\geq \frac{1}{4}p(\frac{3}{2}p+1) + \frac{1}{2}p(n-K) + \frac{3}{2}n(n-K+p+1)\\ &= J.\end{aligned}$$

Inversement, supposons qu'il existe une bijection ϕ de V^* dans $[\![n+p]\!]$ telle que $\omega(\phi) \geq J$. Notons Φ l'ensemble des bijections de V^* dans $[\![n+p]\!]$. Définissons l'entier positif W^* et l'ensemble Φ^* par :

$$W^* = \max_{\phi \in \Phi}\{\omega_1(\phi) + \omega_2(\phi)\}$$

et

$$\Phi^* = \{\phi \in \Phi \mid \omega_1(\phi) + \omega_2(\phi) = W^*\}$$

Nous allons montrer qu'il existe au moins une bijection $\phi \in \Phi^*$ telle que les images de l'ensemble V_2 par ϕ forment un ensemble d'entiers consécutifs. Pour toute bijection $\phi \in \Phi^*$, notons $S(\phi) \subseteq V_1$ l'ensemble des sommets de V_1 encadrés par des éléments de V_2 dans ϕ, c'est-à-dire :

$$S(\phi) = \{u \mid u \in V_1 \wedge \exists u_i \in V_2, \exists u_j \in V_2, \quad \phi(u_i) < \phi(u) < \phi(u_j)\}.$$

Soit $\bar{\phi} \in \Phi^*$ une bijection minimale: $|S(\bar{\phi})| \leq |S(\phi)|$ pour toute bijection $\phi \in \Phi^*$. Montrons que $|S(\bar{\phi})| = 0$ et par suite que $\bar{\phi}$ est la bijection recherchée. En effet, supposons $|S(\bar{\phi})| \neq 0$ et soit $u^* \in S(\bar{\phi})$ le sommet vérifiant $\bar{\phi}(u^*) > \bar{\phi}(u)$ pour tout $u \in S(\bar{\phi})$, $u^* \neq u$. Envisageons les deux cas suivants (c.f. figure 5.5) :

1. Si $\bar{\phi}(u^*) < \bar{\phi}(u)$ pour tout $u \in N(u^*)$; autrement dit, tous les voisins de u^* ne sont pas encadrés par des éléments de V_2 dans $\bar{\phi}$. Soit $v^* \in V_2$ le sommet vérifiant $\bar{\phi}(v^*) < \bar{\phi}(v)$ pour tout $v \in V_2$, $v^* \neq v$. Considérons alors la bijection $\bar{\bar{\phi}} \in \Phi$ obtenue à partir de la bijection $\bar{\phi}$ en échangeant les valeurs des sommets u^* et v^*. Alors, $\omega(\bar{\bar{\phi}}) \geq \omega(\bar{\phi}) = W^*$ et $|S(\bar{\bar{\phi}})| < |S(\bar{\phi})|$, ce qui contredit le choix de la bijection $\bar{\phi}$.
2. Si $\bar{\phi}(u^*) > \bar{\phi}(u)$ pour au moins un sommet $u \in N(u^*)$. Soit $v^* \in V_2$ le sommet vérifiant $\bar{\phi}(v^*) > \bar{\phi}(v)$ pour tout $v \in V_2$, $v^* \neq v$. Considérons alors la bijection $\bar{\bar{\phi}} \in \Phi$ obtenue à partir de la bijection $\bar{\phi}$ en échangeant les valeurs des sommets u^* et v^*. Alors, $\omega(\bar{\bar{\phi}}) \geq \omega(\bar{\phi}) = W^*$ et $|S(\bar{\bar{\phi}})| < |S(\bar{\phi})|$, ce qui contredit une nouvelle fois le choix de la bijection $\bar{\phi}$.

Par conséquent, la bijection $\bar{\phi}$ vérifie $S(\bar{\phi}) = \emptyset$. Aussi, les images de l'ensemble V_2 par la bijection $\bar{\phi}$ forment un ensemble d'entiers consécutifs. Définissons la partition $V_1 = V' \cup V''$ par :

$$V' = \{u \mid u \in V_1 \wedge \bar{\phi}(u) > \bar{\phi}(u_i) \text{ pour tout } u_i \in V_2\}$$
$$V'' = \{u \mid u \in V_1 \wedge \bar{\phi}(u) < \bar{\phi}(u_i) \text{ pour tout } u_i \in V_2\}$$

et notons $|V'| = q$. Remarquons dans un premier temps que :

$$\omega_2(\bar{\phi}) \leq \frac{1}{4}p(\frac{3}{2}p + 1) + \frac{1}{2}p(n - q).$$

Encore une fois, le premier terme compte les sommets maximaux de chaque arête de l'ensemble E_2, tandis que le second terme prend en compte le décalage induit par les $n - q$ sommets du sous-ensemble V''. Il est cependant facile d'obtenir la borne maximale pour $\omega_2(\bar{\phi})$. Il suffit en effet de poser :

$$\begin{cases} n - q + 1 \leq \bar{\phi}(u_i) \leq n - K + \frac{1}{2}p & \text{pour } u \in V_2 \ \wedge \ 1 \leq i \leq \frac{1}{2}p; \\ n - q + \frac{1}{2}p + 1 \leq \bar{\phi}(u_i) \leq n - q + p & \text{pour } u \in V_2 \ \wedge \ \frac{1}{2}p + 1 \leq i \leq p. \end{cases}$$

Aussi, nous supposons dorénavant que :

$$\omega_2(\bar{\phi}) = \frac{1}{4}p(\frac{3}{2}p + 1) + \frac{1}{2}p(n - q).$$

En reprenant le calcul de $\omega(\bar{\Phi})$, il vient :

$$\begin{aligned} \omega(\bar{\phi}) &= \omega_1(\bar{\phi}) + \omega_2(\bar{\phi}) \\ &= \frac{1}{4}p(\frac{3}{2}p + 1) + \frac{1}{2}p(n - q) + \omega_1(\bar{\phi}) \\ &\geq J \quad \text{(par hypothèse)} \\ &= \frac{1}{4}p(\frac{3}{2}p + 1) + \frac{1}{2}p(n - K) + \frac{3}{2}n(n - K + p + 1) \end{aligned}$$

D'où,

$$\omega_1(\bar{\phi}) \geq \frac{1}{2}p(q-K) + \frac{3}{2}n(n-K+p+1).$$

Mais, nous avons toujours :

$$\omega_1(\bar{\phi}) \leq \frac{3}{2}n(n+p)$$

et par suite,

$$\frac{3}{2}n(K-1) \geq \frac{1}{2}p(q-K).$$

Or, $p = 6n(K-1)$. D'où,

$$\frac{3}{2}n(K-1) \geq 3n(K-1)(q-K).$$

Ce qui implique $q \leq K$.

Il reste à montrer que V' est une couverture des sommets pour le graphe G_1. Supposons qu'il existe $\{u,v\} \in E_1$ tel que $u \in V''$ et $v \in V''$. Alors,

$$\omega(\bar{\phi}) \leq (n-q) + \frac{1}{4}p(\frac{3}{2}p+1) + \frac{1}{2}p(n-q) + (\frac{3}{2}n-1)(n+p).$$

Mais, par hypothèse,

$$\omega(\bar{\phi}) \geq \frac{1}{4}p(\frac{3}{2}p+1) + \frac{1}{2}p(n-K) + \frac{3}{2}n(n-K+p+1).$$

D'où,

$$(n-q) + \frac{1}{2}p(n-q) - (n+p) \geq \frac{1}{2}p(n-K) - \frac{3}{2}n(K-1).$$

Or, $q \leq K$, et par suite nous avons :

$$p \leq \frac{3}{2}n(K-1) - K.$$

Ce qui est une contradiction puisque :

$$\begin{aligned} p &= 6n(K-1) \\ &> \frac{3}{2}n(K-1) - K. \end{aligned}$$

Par conséquent, nous avons $u \in V'$ ou $v \in V'$ pour tout $\{u,v\} \in E_1$. Aussi, puisque $|V'| \leq K$, le sous-ensemble V' est une couverture des sommets de taille au plus K du graphe G_1. $\boxed{5.2.9}$

PROPOSITION 5.2.10 ARRANGEMENT LINÉAIRE MAX / MAX *est un problème* NP-*complet même si G est un graphe sans triangles.*

PREUVE

Le problème COUVERTURE DES SOMMETS reste NP-complet pour les graphes sans triangles [Pol74]. Nous reprenons la preuve de la proposition 5.2.9 et nous utilisons comme problème de départ le problème COUVERTURE DES SOMMETS restreint à la classe

des graphes sans triangles. Nous vérifions facilement que l'instance correspondante du problème ARRANGEMENT LINÉAIRE MAX/MAX est un graphe sans triangles. 5.2.10

Nous allons maintenant montrer que le problème ARRANGEMENT LINÉAIRE MAX/MIN est NP-complet. Il s'agit cette fois ci de minimiser la somme des sommets maximaux de chaque arête. Le problème ARRANGEMENT LINÉAIRE MAX/MIN s'énonce simplement de la façon suivante.

PROBLÈME 5.2.11 (ARRANGEMENT LINÉAIRE MAX/MIN)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n et un entier positif K.
QUESTION : Existe-t-il une bijection ϕ de V dans $[\![n]\!]$ telle que :

$$\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} \leq K \ ?$$

En utilisant le problème ARRANGEMENT LINÉAIRE MAX/MAX et en considérant le graphe complémentaire, nous obtenons le résultat suivant.

PROPOSITION 5.2.12 ARRANGEMENT LINÉAIRE MAX/MIN *est un problème* NP-*complet.*

PREUVE

Le problème ARRANGEMENT LINÉAIRE MAX/MIN appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème ARRANGEMENT LINÉAIRE MAX/MAX donnée par un graphe $G = (V,E)$ et un entier positif J. Soit f la fonction définie par :

$$f(n) = \frac{1}{3}n(n^2 - 1).$$

Remarquons que pour tout graphe $G = (V,E)$ d'ordre n et toute bijection ϕ de V dans $[\![n]\!]$, nous avons toujours l'inégalité :

$$\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} \leq f(n).$$

En particulier,

$$\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} = f(n)$$

si et seulement si le graphe G est isomorphe à K_n, *i.e.* G est un graphe complet d'ordre n.

L'instance correspondante du problème ARRANGEMENT LINÉAIRE MAX/MIN est donnée par un graphe $G^* = (V^*,E^*)$ et par un entier positif K définis par :

- $V^* = V$;
- $E^* = \{\{u,v\} \mid \{u,v\} \notin E \wedge u \neq v\}$;
- $K = f(n) - J$.

Clairement, $|V^*| = n$ et $|E^*| = n^2 - n - |E|$. La construction peut s'effectuer en temps polynomial.

Remarquons dans un premier temps que pour toute bijection ϕ de V dans $[\![n]\!]$, nous avons :

$$\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} + \sum_{\{u,v\}\in E^*} \max\{\phi(u),\phi(v)\} = f(n).$$

Par conséquent, il existe une bijection ϕ de V dans $[\![n]\!]$ telle que :

$$\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} \geq J$$

si et seulement si il existe une bijection ϕ de V dans $[\![n]\!]$ telle que :

$$\sum_{\{u,v\}\in E^*} \max\{\phi(u),\phi(v)\} \leq f(n) - J = K.$$

$\boxed{5.2.12}$

À l'aide de la proposition 5.2.12, nous sommes désormais en mesure de montrer que ARRANGEMENT LINÉAIRE MIN / MAX est un problème NP-complet. Ce problème peut être vu comme le dual du problème ARRANGEMENT LINÉAIRE MAX / MIN : il s'agit ici de maximiser la somme des sommets minimaux de chaque arête. Le problème s'énonce de la façon suivante.

PROBLÈME 5.2.13 (ARRANGEMENT LINÉAIRE MIN / MAX)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n, et un entier K.
QUESTION : Existe-t-il une bijection ϕ de V dans $[\![n]\!]$ telle que :

$$\sum_{\{u,v\}\in E} \min\{\phi(u),\phi(v)\} \geq J \ ?$$

PROPOSITION 5.2.14 ARRANGEMENT LINÉAIRE MIN / MAX *est un problème* NP-*complet.*

PREUVE

Le problème ARRANGEMENT LINÉAIRE MIN / MAX appartient à NP. Montrons qu'il est NP-complet. Soient une instance arbitraire de ARRANGEMENT LINÉAIRE MAX / MIN donnée par un graphe $G = (V,E)$ d'ordre n et par un entier K. Nous allons montrer qu'il existe une bijection ϕ de V dans $[\![n]\!]$ telle que :

$$\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} \leq K$$

si et seulement si il existe une bijection $\bar{\phi}$ de V dans $[\![n]\!]$ telle que :

$$\sum_{\{u,v\}\in E} \min\{\bar{\phi}(u),\bar{\phi}(v)\} \geq |E|(n+1) - K.$$

Supposons qu'il existe une bijection ϕ de V dans $[\![n]\!]$ telle que:

$$\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} \leq K.$$

Soit $\bar{\phi}$ la bijection de V dans $[\![n]\!]$ définie par $\bar{\phi}(u) = n - \phi(u) + 1$ pour tout $u \in V$. Alors,

$$\begin{aligned}\sum_{\{u,v\}\in E} \min\{\bar{\phi}(u),\bar{\phi}(v)\} &= \sum_{\{u,v\}\in E} n+1-\max\{\phi(u),\phi(v)\}\\ &= |E|(n+1) - \sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\}\\ &\geq |E|(n+1) - K.\end{aligned}$$

Inversement, supposons qu'il existe une bijection $\bar{\phi}$ de V dans $[\![n]\!]$ telle que:

$$\sum_{\{u,v\}\in E} \min\{\bar{\phi}(u),\bar{\phi}(v)\} \geq |E|(n+1) - K.$$

Soit ϕ la bijection de V dans $[\![n]\!]$ définie par $\phi(u) = n - \bar{\phi}(u) + 1$ pour tout $u \in V$. Alors,

$$\begin{aligned}\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\} &= \sum_{\{u,v\}\in E} n+1-\min\{\bar{\phi}(u),\bar{\phi}(v)\}\\ &= |E|(n+1) - \sum_{\{u,v\}\in E} \min\{\bar{\phi}(u),\bar{\phi}(v)\}\\ &\leq K.\end{aligned}$$

$\boxed{5.2.14}$

Nous avons prouvé précédemment que les trois problèmes ARRANGEMENT LINÉAIRE MAX / MAX, ARRANGEMENT LINÉAIRE MAX / MIN et ARRANGEMENT LINÉAIRE MIN / MAX sont NP-complets (propositions 5.2.9, 5.2.12 et 5.2.14). Nous n'avons pas présenté le problème ARRANGEMENT LINÉAIRE MIN / MIN, mais, par extension, celui-ci consiste à minimiser la somme des sommets minimaux de chaque arête. Nous pouvons, à partir des propositions 5.2.9, 5.2.12 et 5.2.14, établir les résultats suivants.

PROPOSITION 5.2.15 *Soit $\mathcal{G}$ une classe de graphes. Alors, Les problèmes* ARRANGEMENT LINÉAIRE MAX / MAX *et* ARRANGEMENT LINÉAIRE MIN / MIN *restreints à la classe $\mathcal{G}$ sont polynomialement équivalents.*

PROPOSITION 5.2.16 *Soit $\mathcal{G}$ une classe de graphes. Alors, Les problèmes* ARRANGEMENT LINÉAIRE MAX / MIN *et* ARRANGEMENT LINÉAIRE MIN / MAX *restreints à la classe $\mathcal{G}$ sont polynomialement équivalents.*

La connexion entre les quatre problèmes est assurée par la proposition suivante. Pour toute classe de graphes $\mathcal{G}$, nous notons $\overline{\mathcal{G}}$ la classe des graphes complémentaires de $\mathcal{G}$, *i.e.* $\overline{\mathcal{G}} = \{\overline{G} \mid G \in \mathcal{G}\}$.

PROPOSITION 5.2.17 *Soit $\mathcal{G}$ une classe de graphes. Alors, les problèmes* ARRANGEMENT LINÉAIRE MIN / MIN *et* ARRANGEMENT LINÉAIRE MIN / MIN *restreints à la classe $\mathcal{G}$ sont polynomialements équivalents aux problèmes* ARRANGEMENT LINÉAIRE MAX / MIN *et* ARRANGEMENT LINÉAIRE MIN / MAX *restreints à la classe $\overline{\mathcal{G}}$.*

Il nous reste maintenant à montrer que ARRANGEMENT LINÉAIRE MIN / MAX se réduit polynomialement au problème MIN-SURFACE et prouver ainsi que MIN-SURFACE est un problème NP-complet.

PROPOSITION 5.2.18 MIN-SURFACE *est un problème* NP-*complet même si la matrice A a au plus deux éléments non nuls par colonne.*

PREUVE

Le problème MIN-SURFACE appartient à NP. Prouvons maintenant qu'il est NP-complet. Soit une instance arbitraire de ARRANGEMENT LINÉAIRE MIN / MAX donnée par un graphe $G = (V,E)$ d'ordre n et par un entier J. Posons $m = |E|$. Par souci de clarté, nous identifions l'ensemble des sommets V à l'ensemble $[\![n]\!]$ et l'ensemble des arêtes E à l'ensemble $\{e_1,e_2,\ldots,e_m\}$. L'instance correspondante de MIN-SURFACE est donnée par une (0,1)-matrice $A = [a_{i,j}]$ de taille n par m et par un entier K définis de la façon suivante :

- La matrice A est la *matrice d'incidence sommets-arêtes* de G. Autrement dit, chaque ligne i de la matrice A correspond à un sommet i et chaque colonne à une arête $e = \{i,j\} \in E$ de G; de plus, pour toute arête $e = \{i,j\} \in E$, la colonne e a tous ses éléments nuls sauf $a_{i,e}$ et $a_{j,e}$.
- $K = n(m+1) - J$.

Remarquons que, par construction, chaque colonne de la matrice A contient exactement deux éléments non nuls et la ligne i contient exactement $\deg(i)$ élément(s) non nul(s). La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une permutation ϕ de $[\![n]\!]$ telle que :

$$\sum_{\{i,j\}\in E} \min\{\phi(i),\phi(j)\} \geq J$$

Soit $P = [p_{i,j}]$ la matrice de permutation d'ordre n définie par $p_{i,j} = 1$ si et seulement si $\phi(i) = j$. Chaque colonne de la matrice A correspond à une arête et contient donc exactement deux 1. Alors,

$$\begin{aligned} |\mathbb{S}_{\mathrm{PA}}| &= n(m+1) - \sum_{1\leq j\leq m} \mathbb{S}_{\mathrm{Pa}}(j) \\ &= n(m+1) - \sum_{\{i,j\}\in E} \min\{\phi(i),\phi(j)\} \\ &\leq n(m+1) - J. \end{aligned}$$

Inversement, supposons qu'il existe une matrice de permutation $P = [p_{i,j}]$ d'ordre n telle $|\mathbb{S}_{\mathrm{PA}}| \leq K$. Soit ϕ la permutation de $[\![n]\!]$ définie par $\phi(i) = j$ si et seulement si $p_{i,j} = 1$. Alors,

$$\sum_{\{i,j\}\in E} \min\{\phi(i),\phi(j)\} = \sum_{1\leq j\leq m} \mathbb{S}_{\mathrm{PA}}(j).$$

D'où,

$$\begin{aligned} n(m+1) - \sum_{\{i,j\}\in E} \min\{\phi(i),\phi(j)\} &= n(m+1) - \sum_{1\leq j\leq m} \mathbb{S}_{\mathrm{PA}}(j) \\ &= |\mathbb{S}_{\mathrm{PA}}| \\ &\leq n(m+1) - J. \end{aligned}$$

Par suite,

$$\sum_{\{i,j\}\in E} \min\{\phi(i),\phi(j)\} \geq J.$$

$\boxed{5.2.18}$

5.2.4 Le cas des graphes de séparation

Nous montrons dans cette sous-section que les problèmes ARRANGEMENT LINÉAIRE MAX / MAX, ARRANGEMENT LINÉAIRE MIN / MIN, ARRANGEMENT LINÉAIRE MAX / MIN et ARRANGEMENT LINÉAIRE MIN / MAX sont résolubles en temps polynomial lorsque G est un graphe de séparation.

Rappelons qu'un graphe $G = (V,E)$ est un *graphe de séparation* s'il existe une bipartition $V = S \cup K$ de l'ensemble de ses sommets telle que S induit un ensemble stable et K induit une clique. Nous commençons par deux lemmes techniques.

LEMME 5.2.19 *Soient $G = (S \cup K,E)$ un graphe de séparation, $|S| + |K| = n$, et Φ_G l'ensemble des bijections ϕ de $S \cup K$ dans $[\![n]\!]$ pour lesquelles la somme $\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\}$ est maximale. Alors, il existe une bijection $\phi \in \Phi_G$ telle que $1 \leq \phi(u) \leq |S|$ pour tout $u \in S$ et $|S| + 1 \leq \phi(u) \leq n$ pour tout $u \in K$.*

PREUVE

Pour toute bijection $\phi \in \Phi_G$, nous notons $f(\phi) \subseteq S$ l'ensemble des sommets de S qui sont supérieurs à au moins un sommet de K pour la bijection ϕ:

$$f(\phi) = \{u \mid u \in S \text{ et } \exists v \in K, \phi(v) < \phi(u)\}.$$

Soit $\phi \in \Phi_G$ une bijection qui minimise la quantité $f(\phi)$. Nous allons montrer que $f(\phi) = \emptyset$. Supposons par l'absurde que $f(\phi) \neq \emptyset$. Soient u^* le plus grand sommet du sous-ensemble $f(\phi)$ pour la bijection ϕ et $v^* \in K$ le plus petit sommet du sous-ensemble K pour la bijection ϕ, *i.e.*:

$$\begin{aligned} u^* &= \max_{\phi} f(\phi) \\ v^* &= \min_{\phi} K \end{aligned}$$

Clairement, nous avons $\phi(v^*) < \phi(u^*)$. Considérons maintenant la bijection ϕ' de $S \cup K$ dans $[\![n]\!]$ obtenue à partir de ϕ par transposition des éléments u^* et v^*. Nous vérifions facilement que:

$$\sum_{\{u,v\}\in E} \max\{\phi'(u),\phi'(v)\} \geq \sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\}.$$

Mais, nous avons également $|f(\phi)| > |f(\phi')|$. C'est une contradiction puisque, par hypothèse, ϕ est une bijection de l'ensemble Φ_G qui minimise la quantité $|f(\phi)|$. Par conséquent, nous avons $f(\phi) = \emptyset$, ce qui satisfait la propriété demandée. $\boxed{5.2.19}$

Soit $G = (S \cup K,E)$, $|S| = n_S$ et $|K| = n_K$, un graphe de séparation et ϕ une bijection de $S \cup K$ dans $[\![n]\!]$ pour laquelle la somme $\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\}$ est maximale. Alors, en notant A la matrice d'adjacence du graphe G et P la matrice de permutation d'ordre $n_S + n_K$ associée à la bijection ϕ, le lemme 5.2.19 nous indique que nous pouvons toujours supposer que la matrice PAP^T est de la forme:

$$PAP^T = \begin{bmatrix} \mathbf{0}_{n_S,n_S} & B \\ B^T & \mathbf{1}_{n_K,n_K} - \mathbb{I}_{n_K} \end{bmatrix}$$

où B est une (0,1)-matrice de taille n_S par n_K.

LEMME 5.2.20 *Soient $G = (S \cup K,E)$ un graphe de séparation, $|S| + |K| = n$, et ϕ une bijection de $S \cup K$ dans $[\![n]\!]$ pour laquelle la somme $\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\}$ est maximale avec $1 \leq \phi(u) \leq |S|$ pour tout $u \in S$ et $|S| + 1 \leq \phi(u) \leq n$ pour tout $u \in K$. Alors,*

$$\forall v_1,v_2 \in K, \qquad \phi(v_1) < \phi(v_2) \quad \Rightarrow \quad d(v_1) \leq d(v_2).$$

Les lemmes précédents nous indiquent que nous pouvons calculer une bijection ϕ pour laquelle la somme $\sum_{\{u,v\}\in E} \max\{\phi(u),\phi(v)\}$ est maximale (1) en plaçant tout d'abord tous les sommets S avant ceux de K et (2) en ordonnant ensuite les sommets de K par degré croissant.

En conclusion, en rappelant que le complémentaire d'un graphe de séparation est encore un graphe de séparation, nous obtenons immédiatement les résultats suivants.

PROPOSITION 5.2.21 *Les problèmes* ARRANGEMENT LINÉAIRE MAX / MAX, ARRANGEMENT LINÉAIRE MAX / MAX, ARRANGEMENT LINÉAIRE MAX / MIN *et* ARRANGEMENT LINÉAIRE MIN / MAX *sont tous résolubles en temps polynomial lorsque G est un graphe de séparation.*

PREUVE

Lemmes 5.2.19 et 5.2.20 et propositions 5.2.15, 5.2.16 et 5.2.17. $\boxed{5.2.21}$

5.2.5 Surface minimale et forme triangulaire

Nous avons introduit dans cette section la minimisation de surface d'une (0,1)-matrice comme une forme de «*compression*». Le problème de décision associé est NP-complet dans le cas général (proposition 5.2.18). Nous avons également présenté la triangularisation d'une (0,1)-matrice A d'ordre n par permutation des lignes et des colonnes ($PAQ^T \leq \llcorner_n$) comme une forme de «*compression*». Dans quelle mesure ces deux problèmes sont-ils liés?

Dans un premier temps, s'il existe deux matrices de permutation P et Q d'ordre n telles que $\mathbb{I}_n \leq PAQ^T \leq \llcorner_n$, alors $|\mathbb{S}_{\mathrm{PA}}| = n(n+1) - \frac{1}{2}n(n+1) = \frac{1}{2}n(n+1)$ est minimale. Encore une fois, la présence de la diagonale principale non nulle simplifie

grandement le problème. Remarquons cependant que la réciproque n'est pas vraie. En effet, la matrice A d'ordre 3 définie par:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

vérifie $|\mathbb{S}_A| = 6 = (3 \times 4) - 6$, mais $\mathrm{per}(A) = 0$.

Abandonnons dorénavant la diagonale principale non nulle et considérons le problème suivant. Soit A une (0,1)-matrice d'ordre n pour laquelle:

1. il existe deux matrices de permutation P_1 et Q_1 d'ordre n telles que $P_1AQ_1^T \leq \triangle_n$ et
2. il existe une matrice de permutation P_2 telle que $|\mathbb{S}_{P_2A}|$ soit minimale.

Remarquons que les matrices P_1, Q_1 et P_2 ne sont pas nécessairement uniques. Nous nous intéressons à la question suivante: existe-t-il *toujours* une matrice de permutation Q_2 d'ordre n telle que $P_2AQ_2^T \leq \triangle_n$? Fondamentalement, ce problème se ramène à la question suivante: «*minimisation de surface*» et «*triangularisation par permutation des lignes et des colonnes*» sont-ils deux problèmes «*équivalents*» dans un sens que nous pouvons préciser?

La réponse à la question est tout d'abord négative. En effet, soit A la (0,1)-matrice d'ordre 5 définie par:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Nous vérifions $A \leq \triangle_5$ et $|\mathbb{S}_A| = (5 \times 6) - (1 + 2 + 3 + 6 + 6) = 12$. Remarquons de plus que $|\mathbb{S}_A|$ est minimale. Cependant, il existe une matrice de permutation P d'ordre 5 telle que:

$$PA = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Or, $|\mathbb{S}_{PA}| = (5 \times 6) - (4 + 1 + 1 + 6 + 6) = 12 = |\mathbb{S}_A|$, mais $PAQ^T \not\leq \triangle_5$ pour toute matrice de permutation Q d'ordre 5.

Cependant, nous ne sommes pas en mesure d'exhiber un exemple d'une (0,1)-matrice A d'ordre n triangularisable par permutation des lignes et des colonnes ($P_1AQ_1^T \leq \triangle_n$) pour laquelle il existe une matrice de permutation P_2 d'ordre n telle que les deux conditions suivantes soient vérifiées:

1. $|\mathbb{S}_{P_2A}| < |\mathbb{S}_{P_1AQ_1^T}|$;
2. $P_2AQ_2^T \not\leq \triangle_n$ pour toute matrice de permutation Q_2 d'ordre n.

Aussi, nous ne pouvons pas infirmer que la minimisation de surface et la triangularisation par permutation des lignes et des colonnes sont deux problèmes indépendants. Remarquons de plus que même s'il existait un algorithme en temps

polynomial pour le problème MATRICE INF-TRIANGULAIRE, celui-ci ne répondrait que partiellement à la question.

Nous pouvons néanmoins établir une relation entre la surface d'une (0,1)-matrice et la taille minimale d'une sous-matrice triangularisable par permutation des lignes et colonnes. Nous commençons par un lemme technique. Un vecteur $S = (s_1, s_2, \cdots, s_n)$ à valeurs entières positives est *monotone* [BR91] si $s_1 \geq s_2 \geq \cdots \geq s_n$.

LEMME 5.2.22 *Soient $S = (s_1, s_2, \cdots, s_n)$ un vecteur monotone à valeurs entières positives avec $n+1 \geq s_1 \geq s_2 \geq \cdots \geq s_n \geq 1$ et k un entier positif. Si*

$$\sum_{1 \leq i \leq n} s_i > n(k-1)$$

alors $s_i \geq k - i + 1$ pour $1 \leq i \leq k$.

PREUVE

Supposons par l'absurde que la propriété énoncée soit fausse. Alors, il existe un entier j, $1 \leq j \leq k$, tel que $s_j < k - j + 1$. Remarquons dans un premier temps que nous avons toujours $s_k \geq k - k + 1 = 1$ car $s_i \geq 1$ pour $1 \leq i \leq n$. Par conséquent, nous avons $j < k$. Décomposons la somme en deux parties :

$$\sum_{1 \leq i \leq n} s_i = \sum_{1 \leq i \leq j-1} s_i + \sum_{j \leq i \leq n} s_i.$$

Nous avons,

$$\begin{aligned} \sum_{1 \leq i \leq j-1} s_i &\leq \sum_{1 \leq i \leq j-1} n+1 \\ &= (j-1)(n+1) \end{aligned}$$

et

$$\begin{aligned} \sum_{j \leq i \leq n} s_i &\leq \sum_{j \leq i \leq n} k-j \\ &= (n-j+1)(k-j). \end{aligned}$$

car S est un vecteur monotone, *i.e.* si $s_j < k - j + 1$, alors $s_i < k - j + 1$ pour $j + 1 \leq i \leq n$. Par conséquent,

$$\begin{aligned} \sum_{1 \leq i \leq n} s_i &\leq (j-1)(n+1) + (n-j+1)(k-j) \\ &= (n+1)(k-1) - j(k-j). \end{aligned}$$

Or,

$$\begin{aligned} j - k < 0 &\Leftrightarrow (j-k+1) \leq 0 && \text{car } j < k \\ &\Leftrightarrow (j-k+1)(j-1) \leq 0 && \text{car } j \geq 1 \\ &\Leftrightarrow (k-1) - j(k-j) \leq 0 \\ &\Leftrightarrow (n+1)(k-1) - j(k-j) \leq n(k-1). \end{aligned}$$

Par conséquent, nous obtenons :

$$\sum_{1 \leq i \leq n} s_i \leq n(k-1).$$

Ce qui est une contradiction car nous avons :

$$\sum_{1 \leq i \leq n} s_i > n(k-1)$$

par hypothèse. $\boxed{5.2.22}$

En associant à la silhouette d'une (0,1)-matrice donnée sous forme standard décroissante un vecteur monotone, le corollaire suivant est une conséquence immédiate du lemme 5.2.22.

COROLLAIRE 5.2.23 *Soient A une (0,1)-matrice d'ordre n donnée sous forme standard décroissante et k un entier positif. Si $|\mathbb{S}_A| < n(n-k+2)$, alors*

$$A \leq \begin{bmatrix} \triangle_k & \mathbf{1}_{k,n-k} \\ \mathbf{1}_{n-k,k} & \mathbf{1}_{n-k,n-k} \end{bmatrix}.$$

PREUVE

Dans un premier temps, nous avons

$$\begin{aligned} |\mathbb{S}_A| < n(n-k+2) &\Leftrightarrow n(n+1) - \sum_{1 \leq i \leq n} \mathbb{S}_A(i) < n(n-k+2) \\ &\Leftrightarrow \sum_{1 \leq i \leq n} \mathbb{S}_A(i) > n(k-1). \end{aligned}$$

Considérons le vecteur $S = (\mathbb{S}_A(1), \mathbb{S}_A(2), \cdots, \mathbb{S}_A(n))$. Nous vérifions $n+1 \leq \mathbb{S}_A(i) \leq i$ pour $1 \leq i \leq n$ et, puisque A est sous forme standard décroissante, S est un vecteur monotone. En utilisant le lemme 5.2.22, il vient $\mathbb{S}_A(i) \geq k-i+1$ pour $1 \leq i \leq k$. Soit A' la sous-matrice princale de A formée des k premières lignes et colonnes. Nous vérifions facilement $A' \leq \triangle_k$. $\boxed{5.2.23}$

Le corollaire précédent nous donne en fait la taille minimale de la sous-matrice triangularisable par permutation des lignes et des colonnes.

COROLLAIRE 5.2.24 *Soient A une (0,1)-matrice d'ordre n et k un entier positif. Si $|\mathbb{S}_A| < n(n-k+2)$, alors il existe une sous-matrice de A d'ordre au moins k qui est triangularisable par permutation des lignes et des colonnes.*

5.2.6 Résultat négatif d'approximation absolue

Pour terminer cette section, nous présentons un résultat négatif d'approximation absolue pour le problème MIN-SURFACE. Étant donné un problème d'optimisation Π, un *algorithme d'approximation absolue* $\mathcal{A}$ est un algorithme d'approximation (en temps polynomial) pour Π qui vérifie $|\mathcal{A}(I) - OPT(I)| \leq k$ pour une constante

approximation absolue

$k > 0$ et toute instance I. Nous commençons par un lemme trivial qui met en évidence une propriété «*multiplicative*»[11] du problème MIN-SURFACE.

LEMME 5.2.25 *Soient B une (0,1)-matrice de taille m par n et B^* une (0,1)-matrice de taille m par pn, $p \geq 1$, définie par:*

$$B^* = \begin{bmatrix} B_1 & B_2 & \dots & B_p \end{bmatrix}$$

où $B_i = B$ pour $1 \leq i \leq p$. Alors, pour toute matrice de permutation P d'ordre m, nous avons $|\mathbb{S}_{PB}| = \frac{1}{p}|\mathbb{S}_{PB^*}|$.

PROPOSITION 5.2.26 *Si* P $\neq$ NP, *alors il n'existe aucun algorithme d'approximation absolue pour le problème* MIN-SURFACE.

PREUVE

Supposons par l'absurde qu'il existe un algorithme d'approximation absolue $\mathcal{A}$ pour le problème MIN-SURFACE. Notons k la performance garantie par l'algorithme $\mathcal{A}$. Nous allons montrer que cet algorithme peut être utilisé pour calculer une solution optimale pour toute instance I du problème MIN-SURFACE.

Soit une instance I du problème MIN-SURFACE donnée par une (0,1)-matrice B de taille m par n. Considérons une nouvelle instance I' obtenue en considérant la (0,1)-matrice B^* de taille m par $(k+1)n$ définie par:

$$B^* = \begin{bmatrix} B_1 & B_2 & \dots & B_{k+1} \end{bmatrix}$$

où $B_i = B$ pour $1 \leq i \leq k+1$. Nous utilisons notre algorithme d'approximation absolue $\mathcal{A}$ sur cette nouvelle instance I' pour obtenir une solution de valeur $\mathcal{A}(I')$. Alors, par hypothèse, nous avons :

$$|\mathcal{A}(I') - OPT(I')| \leq k.$$

En utilisant le lemme 5.2.25, il vient :

$$|(k+1)\mathcal{A}(I) - (k+1)\, OPT(I)| \leq k.$$

D'où,

$$|\mathcal{A}(I) - OPT(I)| \leq \frac{k}{k+1}.$$

Mais puisque nous ne considérons que des entiers, nous obtenons :

$$|\mathcal{A}(I) - OPT(I)| \leq 0.$$

C'est une contradiction puisque ce dernier résultat nous indique que nous avons calculé en temps polynomial une solution optimale pour le problème MIN-SURFACE.

$\boxed{5.2.26}$

11. Ce type de propriété est déjà employé dans [GJ79]

5.3 Famille de couples de sous-ensembles

5.3.1 Introduction

Nous présentons dans cette section une nouvelle généralisation du problème MATRICE INF-TRIANGULAIRE[12]. Face à notre réelle difficulté quant à la complexité du problème MATRICE INF-TRIANGULAIRE, l'idée que nous souhaitons développer ici est celle d'arrangement de couples de sous-ensembles. Une des motivations de cette approche est le remplacement de la «*forme triangulaire*» par une notion plus générale basée sur la comparaison des cardinaux de deux ensembles (c.f. définition 4.2.3). Nous appelons ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES ce problème dont voici l'énoncé.

PROBLÈME 5.3.1 (ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES)
DONNÉES : Une famille $\mathcal{F} = \{(S_i,T_i) \mid 1 \leq i \leq n\}$ de couples de sous-ensembles appartenant à $\mathcal{S} \times \mathcal{T}$, où $\mathcal{S}$ et $\mathcal{T}$ sont deux ensembles disjoints de même cardinal.
QUESTION : Existe-t-il une permutation π de $[\![n]\!]$ telle que :

$$\text{card}\Big(\bigcup_{1\leq j\leq i} S_{\pi(j)}\Big) \geq \text{card}\Big(\bigcup_{1\leq j\leq i} T_{\pi(j)}\Big)$$

pour $1 \leq i \leq n$?

Un exemple de résolution du problème ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES est donné sur la figure 5.6.

Le problème ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES contient le problème MATRICE INF-TRIANGULAIRE. En effet, soit $A = [a_{i,j}]$ une (0,1)-matrice d'ordre n. Définissons les ensembles $\mathcal{S}$ et $\mathcal{T}$ par :

$$\mathcal{S} = \{s_1,s_2,\ldots,s_n\}$$
$$\mathcal{T} = \{t_1,t_2,\ldots,t_n\}$$

et la famille $\mathcal{F} = \{(S_i,T_i) \mid 1 \leq i \leq n\}$, $S_i \subseteq \mathcal{S}$ et $T_i \subseteq \mathcal{T}$ pour $1 \leq i \leq n$, de couples de sous-ensembles par :

$$S_i = \{s_i\}$$
$$T_i = \{t_j \mid 1 \leq j \leq n \ \wedge \ a_{i,j} = 1\}$$

pour tout $1 \leq i \leq n$. Autrement dit, tous les S_i, $1 \leq i \leq n$, sont des singletons et T_i, $1 \leq i \leq n$, contient les indices des colonnes des éléments non nuls de la ligne i de la (0,1)-matrice A. Alors, il existe deux matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$ si et seulement si il existe une permutation π de $[\![n]\!]$ telle que :

$$\text{card}\Big(\bigcup_{1\leq j\leq i} S_{\pi(j)}\Big) \geq \text{card}\Big(\bigcup_{1\leq j\leq i} T_{\pi(j)}\Big)$$

12. problème 4.1.1 page 58.

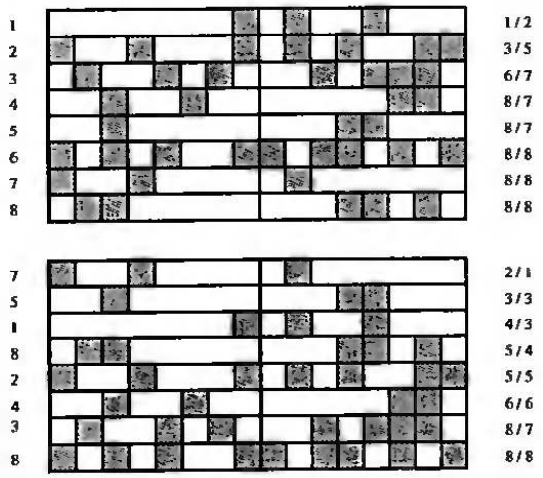


FIG. 5.6 – *Illustration du problème* ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES. *La famille de couples de sous-ensembles est schématisée par deux tableaux accolés. La permutation recherchée est représentée dans la partie inférieure de la figure. Pour chaque ligne, nous indiquons à droite les cardinaux des unions des sous-ensembles cumulés.*

pour $1 \leq i \leq n$ (une illustration de cette remarque est donnée en figure 5.7).

Remarquons que la famille $\mathcal{F}' = \{T_i \mid 1 \leq i \leq n\}$ est la famille des sous-ensembles associés à la matrice A. Nous avons déjà montré[13] qu'il existe deux matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \triangle_n$ si et seulement si il existe un arrangement croissant de $\mathcal{F}'$. Rappelons qu'un arrangement croissant de la famille $\mathcal{F}'$ est une permutation π de $[\![n]\!]$ telle que :

$$\text{card}\Big(\bigcup_{\pi(j)\leq i} T_j\Big) \leq i$$

pour tout $1 \leq i \leq n$. Mais, puisque les sous-ensembles S_i, $1 \leq i \leq n$, sont des singletons disjoints, nous avons :

$$\text{card}\Big(\bigcup_{1\leq j\leq i} S_{\pi(j)}\Big) \geq \text{card}\Big(\bigcup_{1\leq j\leq i} T_{\pi(j)}\Big) \quad \Rightarrow \quad i \geq \text{card}\Big(\bigcup_{1\leq j\leq i} T_{\pi(j)}\Big)$$

pour tout $1 \leq i \leq n$.

En d'autres termes, s'il existe un algorithme en temps polynomial pour le problème ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES, alors il existe également un algorithme en temps polynomial pour le problème MATRICE INF-TRIANGULAIRE. Malheureusement, ce nouveau problème est difficile et nous prouvons dans la sous-section suivante qu'il est NP-complet.

5.3.2 Complexité

Nous avons montré précédemment que les problèmes ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES et MATRICE INF-TRIANGULAIRE sont équivalents si les sous-ensembles

13. Proposition 4.2.5 page 63

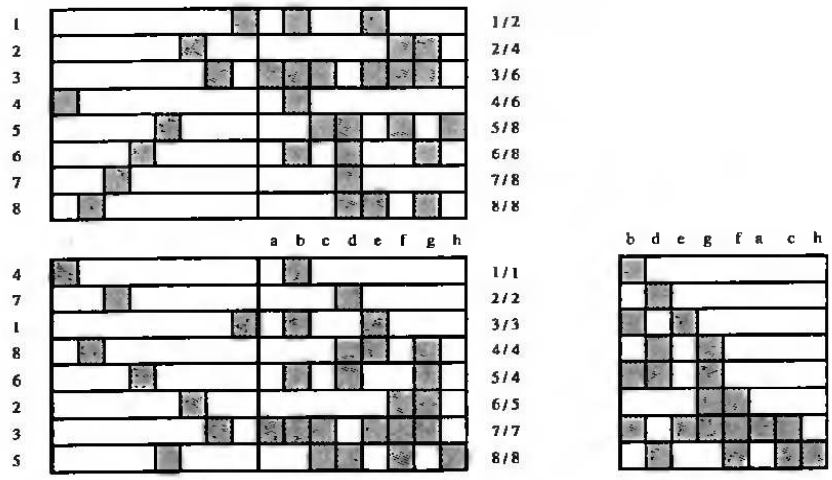


FIG. 5.7 – *Illustration du problème* ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES *dans le cas où les sous-ensembles S_i, $1 \leq i \leq n$, sont des singletons disjoints. Une permutation des colonnes de la matrice d'incidence des sous-ensembles T_i, $1 \leq i \leq n$, est indiquée dans la partie droite de la figure. Nous vérifions que la matrice ainsi obtenue est inférieurement triangulaire.*

S_i pour $1 \leq i \leq n$ sont des singletons disjoints. Nous utilisons une réduction depuis le problème x3c [GJ79] pour montrer que le problème ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES est NP-complet dans le cas général.

PROPOSITION 5.3.2 ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES *est un problème* NP-*complet.*

PREUVE

Le problème ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème x3c donnée par une famille $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$, $|C_i| = 3$ pour $1 \leq i \leq m$, de sous-ensembles d'un $3q$-ensemble $X = \{x_1, x_2, \ldots, x_{3q}\}$. Soient $\mathcal{S}'$, $\mathcal{S}''$ et $\mathcal{T}'$ les ensembles disjoints définis par :

$$\mathcal{S}' = \{x_{i,j} \mid 1 \leq i \leq 3q \text{ et } 1 \leq j \leq 2\}$$
$$\mathcal{S}'' = \{a_i \mid 1 \leq i \leq 3(m-q)\}$$
$$\mathcal{T}' = \{C_{i,j} \mid 1 \leq i \leq m \text{ et } 1 \leq j \leq 3\}.$$

Les ensembles de bases $\mathcal{S}$ et $\mathcal{T}$ sont définis par :

$$\mathcal{S} = \mathcal{S}' \cup \mathcal{S}''$$
$$\mathcal{T} = \mathcal{T}' \cup X.$$

En d'autres termes, pour chaque élément $x_i \in X$, l'ensemble $\mathcal{S}$ contient deux copies distinctes : les éléments $x_{i,1}$ et $x_{i,2}$. De plus, $\mathcal{S}$ contient les éléments a_i pour $1 \leq i \leq 3(m-q)$. Pour chaque sous-ensemble $C_i \in \mathcal{C}$, l'ensemble $\mathcal{T}$ contient trois copies

distinctes : les éléments $C_{i,1}$, $C_{i,2}$ et $C_{i,3}$. De plus, $\mathcal{T}$ contient l'ensemble X. Nous vérifions facilement que $|\mathcal{S}| = |\mathcal{T}| = 3(m+q)$.

L'instance correspondante du problème ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES est donnée par une famille $\mathcal{F}$ de $m + 3(m-q)$ couples de sous-ensembles. Pour chaque sous-ensemble $C_i = \{x_{i_1}, x_{i_2}, x_{i_3}\}$, la famille $\mathcal{F}$ contient le couple (S_i, T_i) défini par :

$$\begin{aligned} S_i &= \{x_{i_1,j}, x_{i_2,j}, x_{i_3,j} \mid 1 \leq j \leq 2\} \\ T_i &= \{C_{i,j} \mid 1 \leq j \leq 3\} \cup \{x_{i_1}, x_{i_2}, x_{i_3}\}. \end{aligned}$$

Pour $m+1 \leq i \leq m+3(m-q)$, la famille $\mathcal{F}$ contient le sous-ensemble (S_i, T_i) où $S_i = \{a_{i-m}\}$ et $T_i = X$. Clairement, nous avons $|\mathcal{F}| = 4m - 3q$. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une sous-famille $\mathcal{C}' \subseteq \mathcal{C}$ de q sous-ensembles disjoints deux à deux. Soit $\mathcal{C}' = \{C_{i_1}, C_{i_2}, \ldots, C_{i_q}\}$ cet ensemble. Considérons une permutation π de $[\![n]\!]$ définie par :

$$\begin{cases} 1 \leq \pi(i) \leq q \quad \text{pour } i \in \{i_1, i_2, \ldots, i_q\} \\ q+1 \leq \pi(i) \leq 3(m-q)+q \quad \text{pour } m+1 \leq i \leq m+3(m-q) \\ 3(m-q)+q+1 \leq \pi(i) \leq 3(m-q)+m \quad \text{pour } 1 \leq i \leq m \text{ et } i \notin \{i_1, i_2, \ldots, i_q\}. \end{cases}$$

Montrons que la permutation π ainsi définie satisfait la propriété demandée.

1. Pour $1 \leq i \leq q$,

$$\operatorname{card}\Big(\bigcup_{1 \leq j \leq i} S_{\pi(j)}\Big) = \operatorname{card}\Big(\bigcup_{1 \leq j \leq i} T_{\pi(j)}\Big) = 6i.$$

car les sous-ensembles de $\mathcal{C}'$ sont disjoints deux à deux.

2. Pour $q+1 \leq i \leq 3(m-q)+q$,

$$\operatorname{card}\Big(\bigcup_{1 \leq j \leq i} S_{\pi(j)}\Big) = 6q + i - q \geq \operatorname{card}\Big(\bigcup_{1 \leq j \leq i} T_{\pi(j)}\Big) = 6q.$$

car $T_{\pi(j)} = X$.

3. Pour $3(m-q)+q+1 \leq i \leq 3(m-q)+m$,

$$\operatorname{card}\Big(\bigcup_{1 \leq j \leq i} S_{\pi(j)}\Big) = 3(m+q) \geq \operatorname{card}\Big(\bigcup_{1 \leq j \leq i} T_{\pi(j)}\Big)$$

car $|\mathcal{S}| = |\mathcal{T}| = 3(m+q)$.

Inversement, supposons qu'il existe une permutation π de $[\![n]\!]$ telle que :

$$\operatorname{card}\Big(\bigcup_{1 \leq j \leq i} S_{\pi(j)}\Big) \geq \operatorname{card}\Big(\bigcup_{1 \leq j \leq i} T_{\pi(j)}\Big) \tag{5.1}$$

pour $1 \leq i \leq n$. Soit j l'entier positif défini par $j = \min\{\pi(i) \mid m+1 \leq i \leq m+3(m-q)\}$. Remarquons, dans un premier temps, que $j > 1$ car $|S_i| = 1 < |T_i| = |X| = 3q$ pour $m+1 \leq i \leq m+3(m-q)$. Soit $\mathcal{C}' = \{C_{i_1}, C_{i_2}, \ldots, C_{i_{j-1}}\} \subseteq \mathcal{C}$ la sous-famille définie par :

$$\{i_1, i_2, \ldots, i_{j-1}\} = \{i \mid 1 \leq i \leq m \ \wedge \ \pi(i) < j\}.$$

Montrons par induction que la sous-famille C' est composée de sous-ensembles disjoints. Pour ce faire, nous notons n_i, $1 \leq i < j$, le nombre d'éléments du sous-ensemble $S_{\pi(i)}$ qui n'appartiennent pas à l'union des sous-ensembles $S_{\pi(k)}$ pour $k < i$. La propriété est vérifiée pour $i = 2$ car

$$\begin{aligned}(5.1) &\Leftrightarrow 6 + 2n_2 \geq (3+3) + 3 + n_2 \\ &\Leftrightarrow n_2 \geq 3.\end{aligned}$$

Supposons que la propriété soit vérifiée pour tout $i < j - 1$. Alors,

$$\begin{aligned}(5.1) &\Leftrightarrow 6i + 2n_{i+1} \geq (3i + 3i) + 3 + n_{i+1} \\ &\Leftrightarrow n_{i+1} \geq 3.\end{aligned}$$

Par conséquent, la sous-famille C' est composée de sous-ensembles disjoints. Mais, $j \leq q+1$, car il existe au plus q sous-ensembles disjoints dans C. Montrons que nous avons en fait $j = q + 1$. En considérant l'équation (5.1) au rang j, il vient :

$$\begin{aligned}(5.1) &\Leftrightarrow 6(j-1) + 1 \geq 3(j-1) + 3q \\ &\Leftrightarrow j \geq q + \frac{2}{3}.\end{aligned}$$

Mais, puisque j est un entier, nous avons $j \geq \lceil q + \frac{2}{3} \rceil = q + 1$. Par suite, la famille C contient q sous-ensembles disjoints. $\boxed{5.3.2}$

Cependant, une nouvelle fois, la proposition 5.3.2 ne nous permet pas de conclure que le problème MATRICE INF-TRIANGULAIRE soit NP-complet. En effet, les sous-ensembles S_i pour $1 \leq i \leq n$ ne sont pas des singletons dans l'instance. La proposition suivante résume la «*dépendance*» entre les deux problèmes.

PROPOSITION 5.3.3 MATRICE INF-TRIANGULAIRE est un problème NP*-complet si et seulement si* ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES *est un problème* NP*-complet lorsque* $|\mathcal{T}| = |\mathcal{S}| = n$ *et les sous-ensembles* S_i *pour* $1 \leq i \leq n$ *sont des singletons disjoints.*

5.4 Partition

5.4.1 Introduction

Les problèmes de *partition* sont les extensions naturelles des problèmes sur les graphes et les matrices. Dans le cas des graphes, il s'agit de partitionner l'ensemble des sommets en K sous-ensembles disjoints (l'entier positif K peut ou non faire partie de l'instance) et de vérifier que chaque sous-ensemble de sommets vérifie une propriété donnée. Par exemple, les problèmes suivants sont tous NP-complets :

- PARTITION EN TRIANGLES [GJ79],
- PARTITION EN SOUS-GRAPHES ISOMORPHES [KH78],
- PARTITION EN SOUS-GRAPHES HAMILTONIENS [Val79],
- PARTITION EN FORÊTS (M.R. Garey et D.S. Johnson - non publié),

- PARTITION EN CLIQUES [Kar72],
- PARTITION EN COUPLAGES PARFAITS [Sch78].

Nous nous proposons dans cette section d'étudier les problèmes de partition dans le cadre des sous-matrices inf / exact-triangulaire.

5.4.2 Partition en sous-matrices inf-triangulaires

Nous nous intéressons au problème de la partition d'une (0,1)-matrice en sous-matrices inf-triangulaires. Nous appelons PARTITION SOUS-MATRICES INF-$\mathbb{T}$ le problème générique.

PROBLÈME 5.4.1 (PARTITION SOUS-MATRICES PRINCIPALES INF-$\mathbb{T}$)
DONNÉES : Une (0,1)-matrice A d'ordre n, un ensemble de (0,1)-matrices triangulaires $\mathbb{T}$ et un entier positif K.
QUESTION : Existe-t-il une matrice de permutation P d'ordre n telle que :

$$PAP^T \leq \begin{bmatrix} T_{m_1} & \mathbf{1}_{m_1,m_2} & \cdots & \mathbf{1}_{m_1,m_K} \\ \mathbf{1}_{m_2,m_1} & T_{m_2} & & \mathbf{1}_{m_2,m_K} \\ \vdots & & \ddots & \vdots \\ \mathbf{1}_{m_K,m_1} & \mathbf{1}_{m_K,m_2} & \cdots & T_{m_K} \end{bmatrix}$$

avec $T_i \in \mathbb{T}$ pour $1 \leq i \leq K$ et $\sum_{1 \leq i \leq K} m_i = n$?

Il suffit[14] en fait de considérer uniquement ce problème pour $\mathbb{T} = \{\triangle_i \mid 1 \leq i \leq n\}$ et $\mathbb{T} = \{\triangledown_i \mid 1 \leq i \leq n\} \cup \{\triangle_i \mid 1 \leq i \leq n\}$. Nous appelons respectivement PARTITION SOUS-MATRICES INF-$\triangle$ et PARTITION SOUS-MATRICES INF-$\triangle$ ces deux problèmes. Nous montrons maintenant qu'ils sont tous les deux NP-complets.

PROPOSITION 5.4.2 PARTITION SOUS-MATRICES INF-$\triangle$ *est un problème* NP-*complet.*

PREUVE
Il suffit de remarquer que pour $K = 1$, le problème est équivalent au problème ARRANGEMENT LINÉAIRE TRIANGULAIRE. La proposition 4.4.8 page 75 permet de conclure. $\boxed{5.4.2}$

Nous utilisons pour le problème PARTITION SOUS-MATRICES INF-$\triangle$ une réduction depuis le problème K-COLORIAGE: étant donnés un graphe $G = (V,E)$ et un entier positif K, existe-t-il une fonction φ de V dans $[\![K]\!]$ telle que $\varphi(u) \neq \varphi(v)$ si $\{u,v\} \in E$? Le problème est NP-complet [Kar72] et ceci pour tout $K \geq 3$ ou si $K = 3$ et G est un graphe planaire avec $\Delta(G) \leq 4$ [GJS76]. Le problème est également NP-complet si G est un graphe circulaire [Ung88]. Par contre, il est résoluble en temps polynomial si G est un graphe triangulé [Gav74a].

PROPOSITION 5.4.3 PARTITION SOUS-MATRICES INF-$\triangle$ *est un problème* NP-*complet même si* $K = 3$.

14. c.f. section 4.6 page 99.

PREUVE

Le problème PARTITION SOUS-MATRICES INF-$\triangle$ appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème 3-COLORIAGE donnée par un graphe $G = (V,E)$ d'ordre n. L'instance correspondante du problème PARTITION SOUS-MATRICES INF-$\triangle$ est donnée par la matrice d'adjacence $A = A(G)$ du graphe G et par un entier positif $K = 3$.

Supposons qu'il existe un 3-coloriage du graphe G. Soit $V = V_1 \cup V_2 \cup V_3$ la partition induite par ce coloriage et notons $n_1 = |V_1|$, $n_2 = |V_2|$ et $n_3 = |V_3|$. Considérons la matrice de permutation P d'ordre n telle que :

1. les n_1 premières lignes et colonnes de la matrice PAP^T sont indicées par les éléments de V_1;
2. les lignes et colonnes $n_1 + 1$ à $n_1 + n_2$ de la matrice PAP^T sont indicées par les éléments de V_2;
3. les n_3 dernières lignes et colonnes de la matrice PAP^T sont indicées par les éléments de V_3.

Alors, nous avons :

$$PAP^T \leq \begin{bmatrix} \mathbf{0}_{n_1} & \mathbf{1}_{n_1,n_2} & \mathbf{1}_{n_1,n_3} \\ \mathbf{1}_{n_2,n_1} & \mathbf{0}_{n_2} & \mathbf{1}_{n_2,n_3} \\ \mathbf{1}_{n_2,n_1} & \mathbf{1}_{n_3,n_2} & \mathbf{0}_{n_3} \end{bmatrix} \leq \begin{bmatrix} \triangle_{n_1} & \mathbf{1}_{n_1,n_2} & \mathbf{1}_{n_1,n_3} \\ \mathbf{1}_{n_2,n_1} & \triangle_{n_2} & \mathbf{1}_{n_2,n_3} \\ \mathbf{1}_{n_2,n_1} & \mathbf{1}_{n_3,n_2} & \triangle_{n_3} \end{bmatrix}.$$

Inversement, supposons qu'il existe une matrice de permutation P d'ordre n telle que :

$$PAP^T \leq \begin{bmatrix} \triangle_{n_1} & \mathbf{1}_{n_1,n_2} & \mathbf{1}_{n_1,n_3} \\ \mathbf{1}_{n_2,n_1} & \triangle_{n_2} & \mathbf{1}_{n_2,n_3} \\ \mathbf{1}_{n_2,n_1} & \mathbf{1}_{n_3,n_2} & \triangle_{n_3} \end{bmatrix}.$$

Puisque A est (0,1)-matrice symétrique, alors $\triangle_{n_1}$, $\triangle_{n_2}$ et $\triangle_{n_3}$ sont également des matrices symétriques. De plus, puisque le graphe G est sans boucle, il vient $\triangle_{n_1} = \mathbf{0}_{n_1,n_1}$, $\triangle_{n_2} = \mathbf{0}_{n_2,n_2}$ et $\triangle_{n_3} = \mathbf{0}_{n_3,n_3}$. Par conséquent, le graphe G est 3-coloriable. $\boxed{5.4.3}$

REMARQUE 5.4.4 Dans le cas des (0,1)-matrices symétriques, $K = 3$ est une borne optimale pour le problème PARTITION SOUS-MATRICES INF-$\triangle$. En d'autres termes, ce problème est résoluble en temps polynomial pour $K = 2$. En effet, dans ce dernier cas, le problème PARTITION SOUS-MATRICES INF-$\triangle$-MATRICE consiste à décider si A est la matrice d'adjacence d'un graphe biparti (ou, de façon équivalente, si A est la matrice d'adjacence d'un graphe 2-coloriable), problème facilement résoluble en temps polynomial [Jun99]. ◇

5.4.3 Partition en sous-matrices exact-triangulaires

Nous étudions maintenant deux problèmes relatifs à la partition en sous-matrices exact-triangulaires. Nous appelons PARTITION SOUS-MATRICES EXACT-T le problème sous sa forme générale.

PROBLÈME 5.4.5 (PARTITION SOUS-MATRICES EXACT-$\mathbb{T}$)
DONNÉES : Une (0,1)-matrice A d'ordre n, un ensemble de (0,1)-matrices triangulaires $\mathbb{T}$ et un entier positif K.
QUESTION : Existe-t-il une matrice de permutation P d'ordre n telle que :

$$PAP^T = \begin{bmatrix} T_{m_1} & * & \cdots & * \\ * & T_{m_2} & & * \\ \vdots & & \ddots & \vdots \\ * & * & \cdots & T_{m_K} \end{bmatrix}$$

avec $T_i \in \mathbb{T}$ pour $1 \leq i \leq K$ et $\sum_{1 \leq i \leq K} m_i = n$?

Comme précédemment, il suffit une nouvelle fois de considérer uniquement les cas $\mathbb{T} = \{\triangle_i \mid 1 \leq i \leq n\}$ et $\mathbb{T} = \{\triangledown_i \mid 1 \leq i \leq n\} \cup \{\triangle_i \mid 1 \leq i \leq n\}$. Nous appelons respectivement PARTITION SOUS-MATRICES EXACT-$\triangle$ et PARTITION SOUS-MATRICES EXACT-$\triangle$ ces deux problèmes. Nous montrons maintenant qu'ils sont NP-complets.

PROPOSITION 5.4.6 PARTITION SOUS-MATRICES EXACT-$\triangle$ *est un problème* NP-*complet même si* $K = 3$.

PREUVE
Le problème PARTITION SOUS-MATRICES EXACT-$\triangle$ appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème 3-COLORIAGE donnée par un graphe $G = (V,E)$. Notons $A = [a_{i,j}]$ la matrice d'adjacence de G. L'instance correspondante du problème PARTITION SOUS-MATRICES EXACT-$\triangle$ est donnée par une (0,1)-matrice $B = [b_{i,j}]$ d'ordre n définie par :

- $b_{i,i} = 1$ pour tout $1 \leq i \leq n$.
- $b_{i,j} = 1$ si et seulement si $a_{i,j} = 0$ et $i > j$.

La construction peut s'effectuer en temps polynomial.

Remarquons dans un premier temps que, pour toute matrice de permutation P d'ordre n, la matrice PAP^T est de la forme :

$$PAP^T = \begin{bmatrix} \mathbf{0}_{m_1} & * & * \\ * & \mathbf{0}_{m_2} & * \\ * & * & \mathbf{0}_{m_3} \end{bmatrix}$$

si et seulement si la matrice PBP^T est de la forme :

$$QBQ^T = \begin{bmatrix} \triangle_{m_1} & * & * \\ * & \triangle_{m_2} & * \\ * & * & \triangle_{m_3} \end{bmatrix}$$

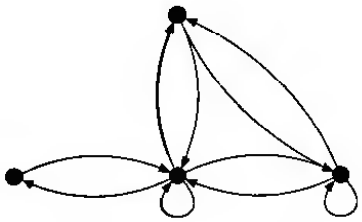
avec $m_1+m_2+m_3 = n$. Or, la matrice PAP^T est de la forme proposée si et seulement si le graphe G est 3-coloriable. $\boxed{5.4.6}$

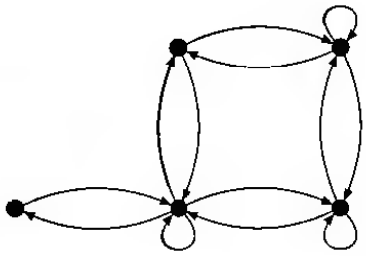
Nous allons maintenant montrer que PARTITION SOUS-MATRICES EXACT-$\triangle$ est un problème NP-complet. Considérons tout d'abord le graphe orienté G donné ci-dessous.

En notant $A(G)$ la matrice d'adjacence du graphe orienté G, nous vérifions immédiatement qu'il existe une matrice de permutation P d'ordre 3 telle que :

$$PA(G)P^T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \triangle_3.$$

Plus généralement, nous disons qu'un graphe orienté G d'ordre n est isomorphe à $\triangle_n$ s'il existe une matrice de permutation P d'ordre n telle que $PA(G)P^T = \triangle_n$, où $A(G)$ est la matrice d'adjacence du graphe G. Par exemple, le graphe G précédent est isomorphe à $\triangle_3$ et les deux graphes suivants sont respectivement isomorphes à $\triangle_4$ et $\triangle_5$.





Nous vérifions facilement que le problème PARTITION SOUS-MATRICES EXACT-$\triangle$ est équivalent au problème suivant : étant donnés un graphe orienté $G = (V,E)$ d'ordre n et un entier positif K, existe-t-il une partition de V en K sous-ensembles disjoints :

$$V = V_1 \cup V_2 \cup \cdots \cup V_K$$

telle que le sous-graphe induit par V_i pour $1 \leq i \leq K$ soit isomorphe à $\triangle_{|V_i|}$? C'est cette version équivalente du problème que nous montrons NP-complète dans la proposition suivante. Nous utilisons pour cela une réduction depuis le problème COUPLAGE 3D.

PROBLÈME 5.4.7 (COUPLAGE 3D)
DONNÉES : Un ensemble $\mathcal{T} \subseteq X \times Y \times Z$, où X, Y et Z sont trois ensembles disjoints ayant chacun m éléments.
QUESTION : Existe-t-il un couplage dans $\mathcal{T}$, *i.e.* un sous-ensemble $\mathcal{T}' \subseteq \mathcal{T}$, $|\mathcal{T}'| = q$, tel que si (x,y,z) et (x',y',z') sont deux triplets de $\mathcal{T}'$, alors $x \neq x'$, $y \neq y'$ et $z \neq z'$.

THÉORÈME 5.4.8 *[Kar72]* COUPLAGE 3D *est un problème* NP*-complet.*

PROPOSITION 5.4.9 PARTITION SOUS-MATRICES EXACT-$\triangle$ *est un problème* NP*-complet.*

PREUVE

Le problème PARTITION SOUS-MATRICES EXACT-$\triangle$ appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème COUPLAGE 3D donnée par trois ensembles X, Y et Z de même cardinalité ($X = \{x_i \mid 1 \leq i \leq m\}$, $Y = \{y_i \mid 1 \leq i \leq m\}$ et $Z = \{z_i \mid 1 \leq i \leq m\}$) et par un sous-ensemble $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\} \subseteq U \times V \times W$.

L'instance correspondante du problème PARTITION SOUS-MATRICES EXACT-$\triangle$ est donnée par un graphe $G = (V,E)$ d'ordre $9n+3m$ et par un entier positif $K = 3n+m$. Le graphe G est défini par :

$$V = X \cup Y \cup Z \cup \{u_{i,j} \mid 1 \leq i \leq m \text{ et } 1 \leq j \leq 9\}$$
$$E = \{(y_i,y_i) \mid 1 \leq i \leq m\} \cup \{(z_i,z_i) \mid 1 \leq i \leq m\} \cup \bigcup_{1 \leq \ell \leq n} E_\ell$$

où chaque sous-ensemble E_ℓ, $1 \leq \ell \leq n$, est associé à un triplet de l'ensemble $\mathcal{T}$ (c.f. graphe de la figure 5.8). Plus précisément, pour chaque triplet $T_\ell = \{x_i,y_j,z_k\} \in \mathcal{T}$, le sous-ensemble E_ℓ est défini par :

$$\begin{aligned}
E_\ell = \{&(x_i,u_{\ell,2}),(u_{\ell,2},x_i),(y_j,u_{\ell,5}),(u_{\ell,5},y_j),(z_k,u_{\ell,8}),(u_{\ell,8},z_k)\} && \cup \\
&\{(u_{\ell,1},u_{\ell,2}),(u_{\ell,2},u_{\ell,1}),(u_{\ell,2},u_{\ell,3}),(u_{\ell,3},u_{\ell,2})\} && \cup \\
&\{(u_{\ell,4},u_{\ell,5}),(u_{\ell,5},u_{\ell,4}),(u_{\ell,5},u_{\ell,6}),(u_{\ell,6},u_{\ell,5})\} && \cup \\
&\{(u_{\ell,7},u_{\ell,8}),(u_{\ell,8},u_{\ell,7}),(u_{\ell,8},u_{\ell,9}),(u_{\ell,9},u_{\ell,8})\} && \cup \\
&\{(u_{\ell,3},u_{\ell,6}),(u_{\ell,6},u_{\ell,3}),(u_{\ell,6},u_{\ell,9}),(u_{\ell,9},u_{\ell,6})\} && \cup \\
&\{(u_{\ell,1},u_{\ell,1}),(u_{\ell,2},u_{\ell,2}),(u_{\ell,5},u_{\ell,5}),(u_{\ell,6},u_{\ell,6}),(u_{\ell,8},u_{\ell,8}),(u_{\ell,9},u_{\ell,9})\}.
\end{aligned}$$

Remarquons que tous les sommets y_i et z_i pour $1 \leq i \leq m$ ont une boucle. Nous vérifions facilement :

$$|V| = 9n + 3m$$
$$|E| = 28n + 2m$$

La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe un sous-ensemble $\mathcal{T}' \subseteq \mathcal{T}$ telle que si (x,y,z) et (x',y',z') sont deux triplets de $\mathcal{T}'$, alors $x \neq x'$, $y \neq y'$ et $z \neq z'$. Pour chaque sous-ensemble $T_\ell = \{x_i,y_j,z_k\} \in \mathcal{T}$,

1. si $T_\ell \notin \mathcal{T}'$, nous formons les sous-ensembles $V_\ell[1]$, $V_\ell[2]$ et $V_\ell[3]$:

$$V_\ell[1] = \{u_{\ell,1},u_{\ell,2},u_{\ell,3}\}$$
$$V_\ell[2] = \{u_{\ell,4},u_{\ell,5},u_{\ell,6}\}$$
$$V_\ell[3] = \{u_{\ell,7},u_{\ell,8},u_{\ell,9}\}$$

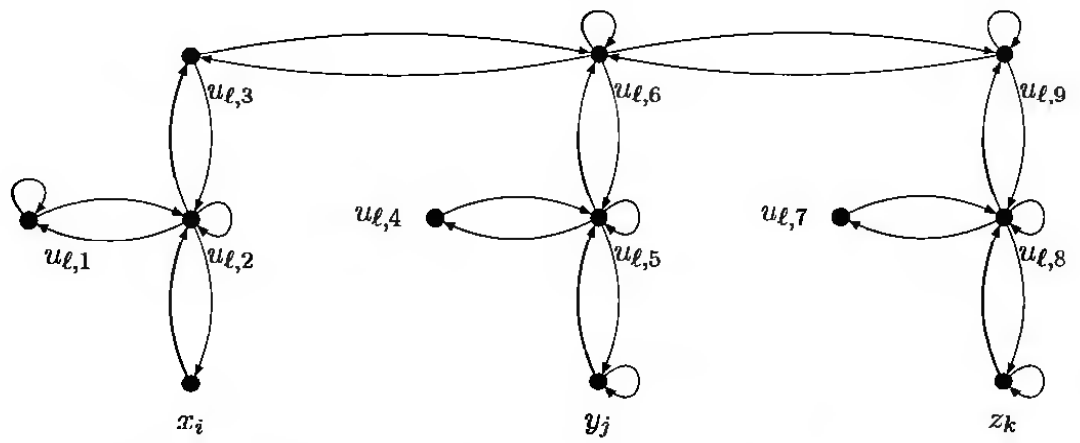


FIG. 5.8 – *Le sous-graphe associé au triplet* $T_\ell = \{x_i,y_j,z_k\}$.

2. si $T_\ell \in \mathcal{T}'$, nous formons les sous-ensembles $V_\ell[1]$, $V_\ell[2]$, $V_\ell[3]$ et $V_\ell[4]$:

$$\begin{aligned} V_\ell[1] &= \{x_i,u_{\ell,1},u_{\ell,2}\} \\ V_\ell[2] &= \{y_j,u_{\ell,4},u_{\ell,5}\} \\ V_\ell[3] &= \{z_k,u_{\ell,7},u_{\ell,8}\} \\ V_\ell[4] &= \{u_{\ell,3},u_{\ell,6},u_{\ell,9}\}. \end{aligned}$$

Nous vérifions facilement que les K sous-ensembles ainsi définis induisent tous des sous-graphes isomorphes à $\triangle_3$.

Inversement, supposons qu'il existe une partition de V en K sous-ensembles disjoints :

$$V = V_1 \cup V_2 \cup \cdots \cup V_K$$

telle que chaque sous-ensemble V_i pour $1 \leq i \leq K$ induise un sous-graphe isomorphe à $\triangle_{|V_i|}$. Remarquons dans un premier temps qu'il n'existe dans G aucun sous-graphe isomorphe à $\triangle_i$ pour $i \geq 4$. Par conséquent, puisque $|V| = 9n + 3m$ et $K = 3n + m$, chaque sous-ensemble V_i pour $1 \leq i \leq K$ induit un sous-graphe isomorphe à $\triangle_3$.

Soit G_ℓ le sous-graphe associé au triplet T_ℓ (c.f. figure 5.8). Supposons tout d'abord que $\{x_i,u_{\ell,1},u_{\ell,2}\}$ soit un élément de la partition de V. Alors, exactement un des ensembles $\{u_{\ell,3},u_{\ell,6},u_{\ell,9}\}$ et $\{u_{\ell,3},u_{\ell,5},u_{\ell,6}\}$ est un élément de la partition de V. Or, $\{u_{\ell,3},u_{\ell,5},u_{\ell,6}\}$ ne peut pas être un élément de la partition de V car le sommet $u_{\ell,4}$ se retrouve ainsi isolé. Par conséquent, $\{u_{\ell,3},u_{\ell,6},u_{\ell,9}\}$ est un élément de la partition de V. Par suite, $\{y_j,u_{\ell,4},u_{\ell,5}\}$ et $\{z_k,u_{\ell,7},u_{\ell,8}\}$ sont également des éléments de la partition de V. Inversement, supposons que $\{x_i,u_{\ell,1},u_{\ell,2}\}$ ne soit pas un élément de la partition de V. Alors, $\{u_{\ell,1},u_{\ell,2},u_{\ell,3}\}$, $\{u_{\ell,4},u_{\ell,5},u_{\ell,6}\}$ et $\{u_{\ell,7},u_{\ell,8},u_{\ell,9}\}$ sont des éléments de la partition de V.

En utilisant des arguments comparables pour les sommets y_j et z_k, nous en déduisons que :

- soit $\{x_i,u_{\ell,1},u_{\ell,2}\}$, $\{y_j,u_{\ell,4},u_{\ell,5}\}$ et $\{z_k,u_{\ell,7},u_{\ell,8}\}$ sont des éléments de la partition de V,

- soit $\{x_i,u_{\ell,1},u_{\ell,2}\}$, $\{y_j,u_{\ell,4},u_{\ell,5}\}$ et $\{z_k,u_{\ell,7},u_{\ell,8}\}$ ne sont pas des éléments de la partition de V.

Par conséquent, puisque cette remarque s'applique à chaque triplet de $\mathcal{T}$, nous en concluons qu'il existe un sous-ensemble $\mathcal{T}' \subseteq \mathcal{T}$ de m triplets disjoints composante à composante. $\boxed{5.4.9}$

REMARQUE 5.4.10 La preuve de la proposition 5.4.9 s'inspire «*librement*» de la preuve du problème PARTITION EN TRIANGLES [GJ79]. Étant donné un graphe $G = (V,E)$, il s'agit de décider s'il existe une partition de l'ensemble des sommets en sous-ensembles disjoints de 3 sommets chacun telle que chaque sous-ensemble induit un triangle. Le problème plus général PARTITION EN SOUS-GRAPHES ISOMORPHES est également NP-complet [KH78]. ◇

5.5 Conclusion et problèmes ouverts

Nous avons présenté dans ce chapitre plusieurs extensions du problème MATRICE INF-TRIANGULAIRE[15]. Toutefois, aucun des problèmes considérés ne nous a permis de déterminer la complexité du problème MATRICE INF-TRIANGULAIRE; tous les problèmes considérés s'avèrent NP-complets. Ce problème reste donc posé.

Nous avons tout d'abord montré que les problèmes MATRICE INF-TRIANGULAIRE et INF-SILHOUETTE MONOTONE CROISSANT STRICT[16] restreint à la classe des (0,1)-matrices carrées sont équivalents (proposition 5.1.7). Par conséquent, l'étude de la complexité du problème INF-SILHOUETTE pour différentes valeurs du paramètre $|\pi|$ semble donc essentielle. Nous avons vu que le problème INF-SILHOUETTE est NP-complet même si $|\pi| = c$, où c est une constante (proposition 5.1.15). La question reste posée si $|\pi| = n$, où n est l'ordre de la matrice. Plus généralement, nous posons la question suivante :

QUESTION 6 *Quelle est la complexité du problème* INF-SILHOUETTE *si* $|\pi| = n - c$, *où n est l'ordre la matrice et c est une constante ?*

Nous avons ensuite présenté le problème MATRICE INF-TRIANGULAIRE comme un problème de compression d'une (0,1)-matrice et montré que MIN-SURFACE est un problème NP-complet dans le cas général[17]. Cependant, nous n'avons donné que des résultats partiels concernant les relations entre les problèmes MIN-SURFACE et MATRICE INF-TRIANGULAIRE.

QUESTION 7 *Quelle est la complexité du problème* MIN-SURFACE *si la matrice A est carrée et est donnée sous forme triangulaire,* i.e. $A \leq \triangle_n$ *?*

D'une façon générale, les liens entre les problèmes MATRICE INF-TRIANGULAIRE et MIN-SURFACE restent à établir. Nous avons montré que le problème MIN-SURFACE reste

15. problème 4.1.1 page 58.
16. Le problème INF-SILHOUETTE est NP-complet dans le cas général (proposition 5.1.15).
17. Proposition 5.2.18 page 143.

NP-complet même si le nombre d'éléments non nuls sur chaque colonne de la matrice A est borné par une constante.

QUESTION 8 *Quelle est la complexité du problème* MIN-SURFACE *si le nombre d'éléments non nuls sur chaque ligne de la matrice A est borné par une constante?*

totalement unimodulaire

Une matrice A est *totalement unimodulaire* si toute sous-matrice carrée de A a pour déterminant 0 ou ± 1. Il est immédiat qu'une matrice totalement unimodulaire est une $(0, \pm 1)$-matrice. Les matrices totalement unimodulaires sont, par exemple, très importantes en programmation linéaire. Rappelons qu'un graphe G est biparti si et seulement si sa matrice d'incidence est totalement unimodulaire [GM79, ADH98]. Aussi, la proposition suivante est immédiate.

PROPOSITION 5.5.1 *Si* ARRANGEMENT LINÉAIRE MIN/MAX *est un problème* NP*-complet même si G est un graphe biparti, alors* MIN-SURFACE *est un problème* NP*-complet même si A est une matrice totalement unimodulaire.*

Rappelons qu'un graphe est co-biparti s'il est le complément d'un graphe biparti. Nous avons donc également de façon équivalente.

PROPOSITION 5.5.2 *Si* ARRANGEMENT LINÉAIRE MAX/MAX *est un problème* NP*-complet même si G est un graphe co-biparti, alors* MIN-SURFACE *est un problème* NP*-complet même si A est une matrice totalement unimodulaire.*

Cependant, la question concernant la complexité du problème ARRANGEMENT LINÉAIRE MIN/MAX lorsque G est un graphe biparti reste posée[18]. Un graphe biparti ou co-biparti est *parfait* [Ber75]. Nous avons montré que dans le cas des graphes de séparation - ces graphes sont parfaits - les problèmes ARRANGEMENT LINÉAIRE MAX/MAX, ARRANGEMENT LINÉAIRE MAX/MIN et ARRANGEMENT LINÉAIRE MIN/MAX sont résolubles en temps polynomial. Notons enfin que même si le problème ARRANGEMENT LINÉAIRE MIN/MAX est résoluble en temps polynomial dans le cas des graphes bipartis, ceci ne prouverait pas que le problème MIN-SURFACE est résoluble en temps polynomial dans le cas des (0,1)-matrices totalement unimodulaires.

Enfin, nous avons montré que ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES est un problème NP-complet. Nous avons de plus remarqué que ce problème est équivalent au problème MATRICE INF-TRIANGULAIRE lorsque les sous-ensembles S_i sont des singletons disjoints.

QUESTION 9 *Quelle est la complexité du problème* ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES *si les sous-ensembles S_i sont des singletons disjoints?*

Plus généralement, le problème ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES est-il NP-complet si les tous les sous-ensembles S_i sont disjoints et de même cardinal?

18. Remarquons que si MIN-SURFACE est un problème NP-complet même si A est la matrice d'incidence d'un hypergraphe d'intervalle, alors MIN-SURFACE est un problème NP-complet même si A est une matrice totalement unimodulaire.

Chapitre 6

Structures secondaires d'ARN et familles de 2-intervalles

Sommaire

6.1 Introduction **163**
6.2 Structure secondaire **166**
6.3 Famille de 2-intervalles **168**
6.4 Graphe d'intersection **171**
6.5 Sous-famille de 2-intervalles disjoints **175**
6.6 Recherche de motifs **191**
6.7 Conclusion et problèmes ouverts **207**

Nous présentons dans ce chapitre quelques résultats de complexité relatifs aux familles de 2-intervalles dans le cadre des structures secondaires d'ARN. Étant donnés une famille de 2-intervalles $\mathcal{F}$ et un modèle d'intersection $\mathcal{R}$, nous nous intéressons essentiellement (1) *au calcul d'une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de cardinalité maximale dont les éléments sont comparables deux à deux par une relation de $\mathcal{R}$ et* (2) *à la recherche d'un* «motif générique» *dans $\mathcal{F}$.*

6.1 Introduction

Au niveau de l'ARN, les signaux biologiques sont définis dans une large mesure par une combinaison de structures spatiales. La structure est établie par des appariements intra-chaîne. Une suite de tels appariements forme une *hélice*. C'est l'arrangement relatif de ces différentes hélices qui définit la forme spatiale de la structure. ***hélice***

Lors de l'étude d'un génome, il est particulièrement important de pouvoir localiser un signal donné. L'approche classique consiste à concevoir un programme dédié pour chaque structure. Il existe par exemple des programmes spécialisés pour identifier les *introns auto-catalytiques de groupe I*[1] [LDM94] ou les ARNt [EML96] dans les banques de séquences. Nous parlons dans ce cas de programmes *had hoc*. Les principaux avantages de cette approche sont :

– rapidité du programme,

1. c.f. chapitre suivant «*Reconnaissance automatique des introns auto-catalytiques de groupe I*»

- spécialisation,
- facilité à traiter les «*cas exceptionnels*» du problème.

Néanmoins, les programmes *had hoc* souffrent d'un certain nombre de défauts parmi lesquels nous pouvons citer :

- temps de développement important,
- faible réutilisation du code (en général),
- peu évolutif,
- manque de souplesse.

Si l'approche *had hoc* est toujours largement répandue aujourd'hui, d'autres travaux ont été développés permettant non plus de décrire «*comment*» chercher, mais plutôt «*quoi*» chercher. Autrement dit, il s'agit de décrire dans un premier temps une structure spatiale particulière, puis de rechercher cette description dans une séquence. Un telle approche est dite *déclarative*. C'est par exemple le cas du programme `Palingol` [BKV96]. `Palingol` est un langage de description de structure spatiale muni d'un algorithme de recherche. Le langage de description permet de définir une liste de *propriétés* (*contraintes*) que doit posséder un objet pour être considéré comme une occurrence du motif. Ensuite, le moteur de recherche effectue le criblage d'une banque de séquence pour trouver les occurrences du motif.

Comme nous l'avons esquissé précédemment, l'approche déclarative se décompose essentiellement en deux étapes *indépendantes* :

1. calcul de l'ensemble des hélices potentielles;
2. recherche d'un sous-ensemble d'hélices.

Le première étape consiste donc à calculer «*toutes*» les hélices dans la séquence qui sont susceptibles d'appartenir à la structure spatiale recherchée. Bien sur, plusieurs stratégies sont possibles: algorithme naïf, extension de l'algorithme KMR [KMR72], ... Typiquement, toutes les hélices possibles dans une séquence ne doivent pas forcément être considérées comme une hélice potentielle. En particulier, les «*inclusions strictes*» d'hélices n'ont que peu de sens biologique et sont de fait en général supprimées.

La deuxième étape utilise un algorithme de type «*séparation et évaluation*» [BKV96] pour rechercher une combinaison des hélices potentielles précédemment calculées afin de former la structure spatiale souhaitée.

Notre but n'est pas dans ce chapitre de proposer un algorithme pour ce problème, mais plutôt d'étudier quelques problèmes de complexité sous-jacents. En particulier, nous ne nous intéresserons qu'à la seconde étape, à savoir la recherche d'un sous-ensemble d'hélices correspondant à un motif.

Nous commençons par une très courte introduction sur les structures secondaires des ARN. Après quelques brefs rappels sur les *graphes d'intersection* et les *graphes d'intervalles* en particulier, nous introduisons la notion de 2-*intervalle*. Intuitivement, un 2-intervalle est un couple de deux intervalles disjoints et le graphe d'intersection d'une famille de 2-intervalles est appelé un *graphe de 2-intervalles* [TH79].

2-intervalle

Lorsque deux 2-intervalles sont disjoints[2], nous disons qu'ils sont *comparables*. Nous nous intéressons essentiellement aux différents «*modèles de comparaison*» entre

modèle de comparaison

2. Deux 2-intervalles sont disjoints si l'intersection des intervalles qui les composent est l'ensemble

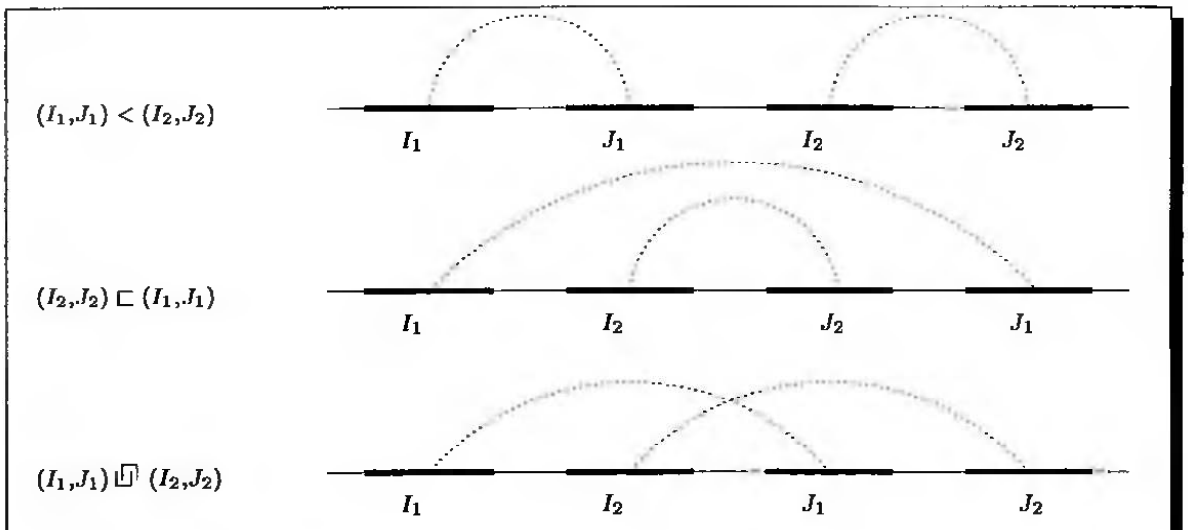


FIG. 6.1 – *Relations possibles entre 2-intervalles.*

2-intervalles disjoints. Plus précisément, étant donnés deux 2-intervalles $D_1 = (I_1,J_1)$ et $D_2 = (I_2,J_2)$, nous voyons sur la figure 6.1 que plusieurs arrangements des intervalles qui les composent sont possibles. Nous introduisons donc les relations binaires[3] $<$, $\sqsubset$ et $\sqcap$ entre 2-intervalles disjoints. Un modèle de comparaison, généralement noté $\mathcal{R}$ dans la suite, est un sous-ensemble non vide de l'ensemble des relations binaires $\{<, \sqsubset, \sqcap\}$. Nous disons que deux 2-intervalles sont $\mathcal{R}$-comparables s'ils sont comparables par l'une des relations binaires de $\mathcal{R}$. Nous tenterons tout au long de ce chapitre d'examiner la complexité des problèmes pour différents modèles de comparaison. $<, \sqsubset, \sqcap$

Nous étudierons le problème de la recherche de la plus grande sous-famille de 2-intervalles $\mathcal{R}$-comparables, où $\mathcal{R}$ est un modèle de comparaison donné dans l'instance. Nous montrerons que le problème est NP-complet[4] pour les modèles de comparaison $\mathcal{R} = \{<, \sqsubset, \sqcap\}$ et $\mathcal{R} = \{\sqsubset, \sqcap\}$. Cependant, il est résoluble en temps polynomial pour les modèles de comparaison $\mathcal{R} = \{<, \sqsubset\}$, $\mathcal{R} = \{<\}$, $\mathcal{R} = \{\sqsubset\}$ et $\mathcal{R} = \{\sqcap\}$. Nous n'avons pas été en mesure de déterminer la complexité de ce problème pour le modèle de comparaison $\mathcal{R} = \{<, \sqcap\}$.

Enfin, nous étudierons dans une dernière partie le problème de la recherche d'un *motif* dans une famille de 2-intervalles pour différents modèles de comparaison. Il s'agit cette fois-ci de déterminer s'il existe une occurrence d'un motif respectant un modèle de comparaison dans une famille de 2-intervalles. Par exemple, dans le cas du modèle de comparaison $\mathcal{R} = \{<, \sqsubset\}$, les motifs sont donnés par des mots de Dyck[5]. Nous montrerons en particulier que la recherche d'un motif donné dans une famille de 2-intervalles est un problème NP-complet pour les modèle de comparaison $\mathcal{R} = \{<, \sqsubset, \sqcap\}$ et $\mathcal{R} = \{\sqsubset, \sqcap\}$. En revanche, le problème est résoluble en

vide.

3. Mentionnons dès à présent que le symbole '$<$' sera utilisé à la fois pour la comparaison entre 2-intervalles et pour la précédence entre intervalle.

4. Nous présenterons néanmoins quelques cas particuliers résolubles en temps polynomial.

5. Cet exemple correspond au cas de la recherche d'une structure secondaire d'ARN particulière sans pseudo-nœud.

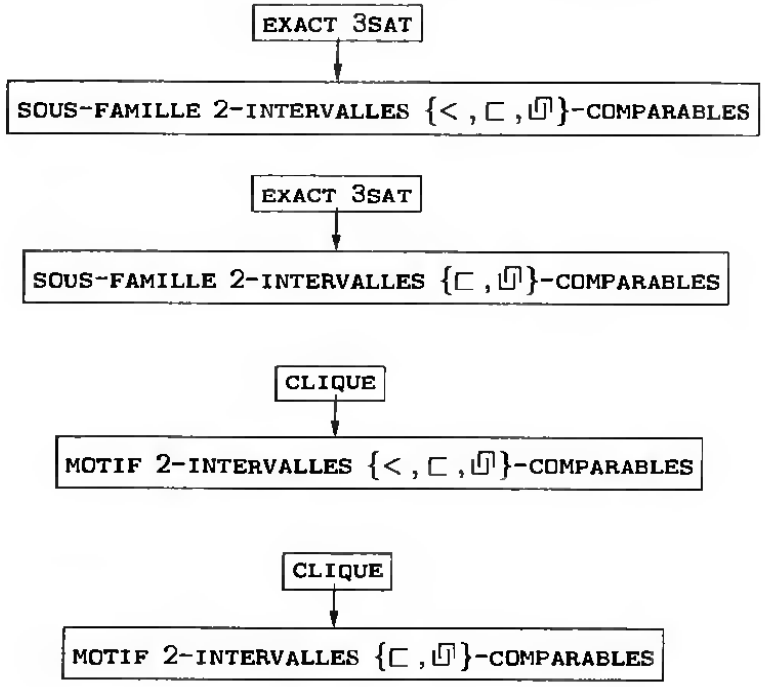


FIG. 6.2 – *Schémas des réductions polynomiales présentées dans ce chapitre.*

temps polynomial pour les modèles d'intersection $\mathcal{R} = \{<\}$, $\mathcal{R} = \{\sqsubset\}$ et $\mathcal{R} = \{⧉\}$. Nous n'avons pas été en mesure d'apporter une réponse quant à la complexité de ce problème pour les modèles d'intersection $\mathcal{R} = \{<, ⧉\}$ et $\mathcal{R} = \{<, \sqsubset\}$.

6.2 Structure secondaire

6.2.1 Présentation

repliement

Les interactions favorables au *repliement* sont principalement les liaisons hydrogènes entre bases complémentaires. Un ensemble de liaisons hydrogènes entre deux bases complémentaires forme une *paire de bases*. Les paires de bases les plus communément rencontrées dans les ARN sont les paires $G - C$ et $A - U$, et, dans une moindre mesure, $G - U$. D'autres éléments ont un rôle dans la détermination du repliement d'énergie minimale : contraintes stéréo-chimiques, torsions des liaisons phosphodiester, forces électrostatiques, effet de solvant, température, *etc* ...

structure secondaire

Une *structure secondaire* d'une séquence s est un ensemble S de paires de base $(i \cdot j)$ avec $1 \leq i < j \leq |s|$ tel que, pour tout $(i \cdot j)$, $(i' \cdot j') \in S$, nous avons :

$$i = i' \quad \Leftrightarrow \quad j = j'.$$

Autrement dit, chaque base est appariée avec au plus une base (exclusion des *interactions tertiaires* [ZS84]). Un exemple de structure secondaire est donné en figure 6.3.

pseudo-nœud

Une restriction d'ordre purement technique est habituellement imposée aux structures secondaires : elles ne doivent pas contenir de *pseudo-nœud*. Un pseudo-nœud

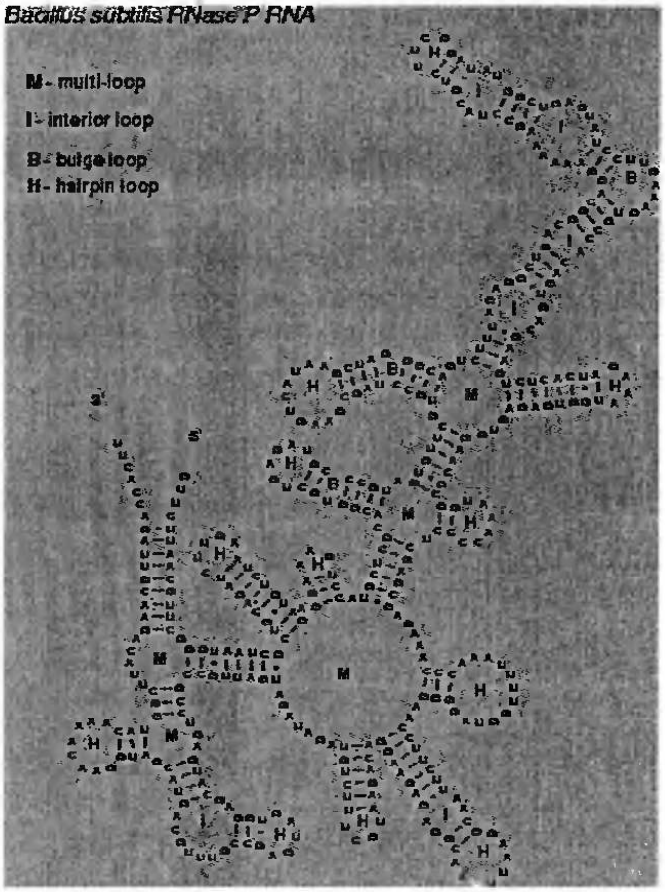


FIG. 6.3 – *Une structure secondaire d'ARN.*

est constitué de deux paires de bases qui se chevauchent : $(i \cdot j)$,$(i' \cdot j') \in S$ avec $i < i' < j < j'$. En plus de limiter le nombre de structures secondaires à considérer, cette condition a pour conséquence directe que toute paire $(i \cdot j)$ dans un structure secondaire sépare la molécule en deux parties indépendantes. C'est sur la base de cette remarque que de nombreux algorithmes de programmation dynamique ont été proposés : [NPGK78] maximise le nombre de paires de bases appariées; il a ensuite été modifié pour minimiser l'énergie thermodynamique des boucles et paires de segments [NJ80] et pour tenir compte de contraintes structurelles [ZS81].

Une structure secondaire sans pseudo-nœud peut facilement être générée par une *grammaire hors contexte* [Sea92]. Les *grammaires hors contextes stochastiques* ont été utilisées pour modéliser des modèles probabilistes [SBH+94, ED94]. L'algorithme de prédiction de structures secondaires d'ARN `MFOLD` recherche les structures d'énergie minimale en utilisant des paramètres thermodynamiques.

Cependant, les pseudo-nœuds sont des structures importantes des ARN [tDPD92]. Par exemple, les introns auto-catalytiques de groupe I contiennent un pseudo-nœud [LDM94]. Il est par contre difficile de les modéliser. En particulier, ils ne peuvent pas être décrits par une grammaire hors-contexte, ni prédits par le programme `MFOLD`. Un cadre formel plus élaboré est donc nécessaire, mais, le niveau suivant dans la hiérarchie de Chomsky est constitué des *grammaires contextuelles* dont l'analyse syntaxique est un problème NP-complet. Néanmoins, plusieurs travaux sont consacrés à la modélisation des pseudo-nœuds [Lef97].

6.2.2 Références

Un chapitre de [DEKM98] est consacré à l'étude des structures secondaires des ARN. Une description plus formelle est présentée, par exemple, dans [WS78, Wat78, HSS98].

6.3 Famille de 2-intervalles

6.3.1 Ensemble partiellement ordonné

comparable
incomparable

Soit $P = (X, \leq_P)$ un ensemble partiellement ordonné. Les éléments x,$y \in X$ sont *comparables* dans P, noté $x \sim y$, si $x \leq_P y$ ou $u \leq_P x$. Inversement, les éléments x et y sont *incomparables*, noté $x \parallel y$, si $x \not\leq_P y$ et $y \not\leq_P x$. Le graphe $G = (X,E)$ où $\{x,y\} \in E$ si et seulement si $x \sim y$ est appelé le *graphe de comparaison* de P. Un élément x est *couvert* par y (ou y couvre x), noté $x \lessdot_P y$, si $x \leq_P y$ et si, pour tout $z \in X$ tel que $x \leq_P z \leq_P y$, $x = z$ ou $y = z$. Cette relation binaire définit le *diagramme de Hasse* de P, et le graphe associé est appelé le *graphe de couverture* de P.

(anti)chaîne

Une *chaîne* (resp. *antichaîne*) dans P est un sous-ensemble X' de X d'éléments deux à deux comparables (resp. incomparables). La *longueur* d'une chaîne $X' \subseteq X$ est $|X'| - 1$, et la *largeur* d'une antichaîne $X' \subseteq X$ est $|X'|$. La *hauteur* de P, notée $h(P)$, est la longueur maximale d'une chaîne dans P. La *largeur* de P, notée $w(P)$, est la taille maximale d'une antichaîne dans P.

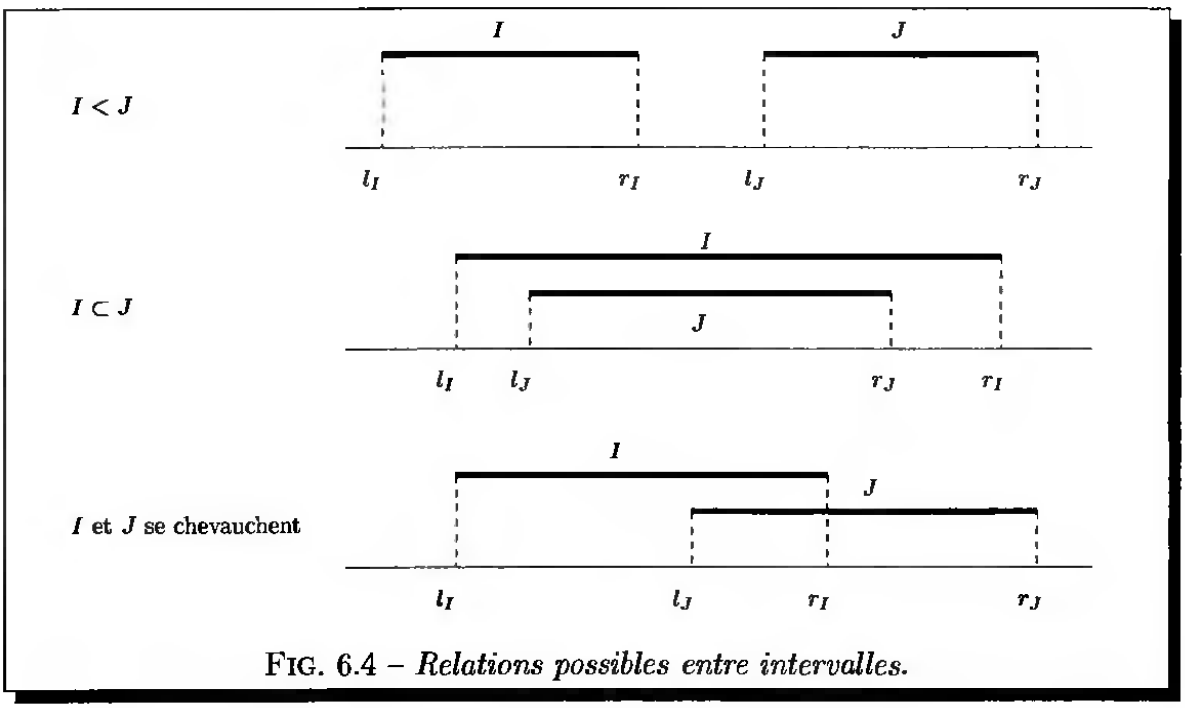


FIG. 6.4 – *Relations possibles entre intervalles.*

6.3.2 Intervalle

Un *intervalle* est l'ensemble des points sur la droite réelle compris entre deux points extrémités. En général, si I est un intervalle, nous notons l_I (resp r_I) son point extrémité gauche (resp. droit). Deux ordres partiels sont naturellement associés aux intervalles sur la droite réelle. Le premier, généralement noté $<$, est associé à la relation de *précédence* entre les intervalles disjoints: ***intervalle*** ***précédence***

$$I < J \quad \text{si et seulement si} \quad r_I < l_J.$$

Autrement dit, $I < J$ si l'intervalle I est «*strictement à gauche*» de l'intervalle J. Le second, généralement noté $I \subseteq J$, est associé à la relation d'*inclusion* entre intervalles: ***inclusion***

$$I \subseteq J \quad \text{si et seulement si} \quad l_J \geq l_I \quad \text{et} \quad r_I \geq r_J.$$

En d'autres termes, $I \subseteq J$ si I est un sous-intervalle de l'intervalle J. Deux intervalles I et J incomparables par $<$ et $\subseteq$ sont dits *chevauchants*. ***chevauchement***

Ces définitions sont illustrées par la figure 6.4. L'intersection et l'union de deux intervalles I et J sont notées respectivement $I \cap J$ et $I \cup J$ et sont définies par:

$$I \cap J = \{x \mid l_I \leq x \leq r_I \ \text{ et } \ l_J \leq x \leq r_J\}$$
$$I \cup J = \{x \mid l_I \leq x \leq r_I \ \text{ ou } \ l_J \leq x \leq r_J\}$$

Deux intervalles I et J sont disjoints ($I \cap J = \emptyset$) si et seulement si ils sont comparables par la relation $<$, *i.e.* $I < J$ ou $J < I$.

6.3.3 2-intervalle

Un 2-intervalle est un couple de deux intervalles disjoints. Par convention, nous ***2-intervalle***

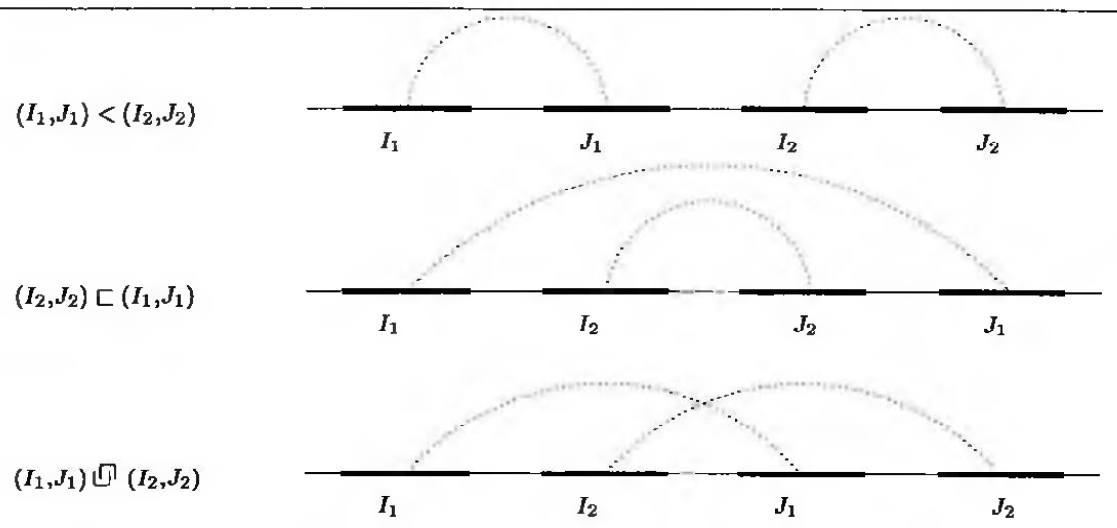


FIG. 6.5 – *Relations possibles entre 2-intervalles.*

notons toujours un 2-intervalle par un couple (I,J) avec $I < J$, où $<$ désigne ici la relation de précédence entre intervalles. Deux 2-intervalles $D_1 = (I_1,J_1)$ et $D_2 = (I_2,J_2)$ sont *comparables* si leur intersection $D_1 \cap D_2$ définie par :

$$D_1 \cap D_2 = (I_1 \cap I_2) \cup (I_1 \cup J_2) \cup (J_1 \cap I_2) \cup (J_1 \cup J_2).$$

est l'ensemble vide.

Soient $D_1 = (I_1,J_1)$ et $D_2 = (I_2,J_2)$ deux 2-intervalles comparables. Nous vérifions facilement sur la figure 6.5 que ces deux 2-intervalles sont dans l'une des situations (ou situations symétriques) données. Nous considérerons donc dans la suite les relations binaires $<$, $\sqsubset$ et $\between$ entre 2-intervalles définies par :

- $D_1 < D_2$ si et seulement[6] si $J_1 < I_2$;
- $D_2 \sqsubset D_1$ si et seulement si $I_1 < I_2$ et $J_2 < J_1$;
- $D_1 \between D_2$ si et seulement si $I_1 < I_2$ et $J_1 < J_2$.

Remarquons que si deux 2-intervalles D_1 et D_2 sont comparables par $<$, $\sqsubseteq$ ou $\between$, alors ils sont disjoints. De plus, ces trois relations sont incompatibles. Il est également important de souligner que $<$ et $\sqsubset$ sont des relations 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. $D_1 \sqsubset D_3$). Par suite, si $\mathcal{F}$ est une famille de 2-intervalles, les paires $(\mathcal{F}, <)$ et $(\mathcal{F}, \sqsubset)$ sont des ensemble partiellement ordonnés (stricts). Remarquons enfin que $\between$ n'est pas une relation transitive et que, par conséquent, la paire $(\mathcal{F}, \between)$ *n'est pas* un ensemble partiellement ordonné.

Soit $R \in \{<, \sqsubset, \between\}$ une relation, nous notons $D_1 \sim_R D_2$ (resp. $D_1 \parallel_R D_2$) si D_1 et D_2 sont *comparables* (resp. *incomparables*) par R. Remarquons que $D_1 \cap D_2 = \emptyset$ si et seulement si il existe une relation $R \in \{<, \sqsubset, \between\}$ telle que $D_1 \sim_R D_2$. Nous pouvons définir différents *modèles de comparaison* en ne considérant qu'un sous-ensemble (non vide) de $\{<, \sqsubset, \between\}$.

DÉFINITION 6.3.1 (MODÈLE DE COMPARAISON) *Soient $\mathcal{F}$ une famille de 2-interval-*

6. Les deux symboles '$<$' désignent des relations différentes. Le premier est une relation entre 2-intervalles, tandis que le second est une relation entre intervalles.

les et $\mathcal{R}$ une partie non vide de l'ensemble des relations binaires $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$. Deux 2-intervalles D_i et D_j de $\mathcal{F}$ sont $\mathcal{R}$-comparables, noté $D_i \sim_{\mathcal{R}} D_j$, s'il sont comparables par une des relations binaires du sous-ensemble $\mathcal{R}$:

$$D_i \sim_{\mathcal{R}} D_j = \emptyset \qquad \textit{si et seulement si} \quad \exists R \in \mathcal{R}, \quad D_i \sim_R D_j.$$

L'ensemble $\mathcal{R}$ est appelé un modèle de comparaison. ***modèle de comparaison***

Par souci de clarté, nous dirons en général que deux 2-intervalles sont «*comparables*» en lieu et place de «$\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-*comparables*».

DÉFINITION 6.3.2 (COUPLE D'INTERVALLES EMBOÎTÉS) *Un* couple d'intervalles emboîtés *est un couplage d'intervalles (I,J) avec $J \subseteq I$.* ***emboîtés***

DÉFINITION 6.3.3 (INTERVALLE EXTÉRIEUR / INTÉRIEUR) *Soit $D = (I,J)$ un 2-intervalle. L'*intervalle extérieur *(fermé) de D, noté $ext(D)$, est défini par :* ***intervalle extérieur***

$$ext(D) = [l_I, r_J]$$

*et l'*intervalle intérieur *(ouvert) de D, noté $int(D)$, est défini par :* ***intervalle intérieur***

$$int(D) =]r_I, l_J[.$$

Remarquons que, pour tout 2-intervalle D, nous avons toujours $ext(D) \supseteq int(D)$. Autrement dit, le couple $(ext(D), int(D))$ est un couple d'intervalles emboîtés. Si D est un 2-intervalle, nous notons en général $\widetilde{D}$ le couple d'intervalles $(ext(D), int(D))$. Par extension, étant donnée une famille $\mathcal{F}$ de 2-intervalles, $\widetilde{\mathcal{F}}$ est la famille de couples d'intervalles emboîtés définie par :

$$\widetilde{\mathcal{F}} = \{\widetilde{D} \mid D \in \mathcal{F}\}.$$

Soit $\mathcal{F}$ une famille de 2-intervalles. Le *support* de $\mathcal{F}$, noté $supp(\mathcal{F})$, est l'ensemble des intervalles sous-jacents à $\mathcal{F}$: ***support***

$$supp(\mathcal{F}) = \{I, J \mid (I,J) \in \mathcal{F}\}$$

La famille $\mathcal{F}$ est à *support unitaire* si tous les intervalles de son support sont de longueur 1; elle est à *support disjoint* si tous les intervalles de son support sont disjoints : ***unitaire*** ***disjoint***

$$\forall I, J \in supp(\mathcal{F}), \qquad I \cap J \neq \emptyset \quad \Rightarrow \quad I = J$$

Un exemple d'une famille de 2-intervalles à support disjoint est donné sur la figure 6.6. Bien entendu, si la famille $\mathcal{F}$ est à support disjoint, nous pouvons toujours supposer qu'elle est également à support unitaire.

6.4 Graphe d'intersection

6.4.1 Graphe d'intersection

Soit $\mathcal{F}$ une famille de sous-ensembles. Le *graphe d'intersection* [Gol80, MM99] de $\mathcal{F}$, généralement noté $G = \Omega(\mathcal{F})$, est obtenu en associant à chaque sommet ***graphe d'intersection***

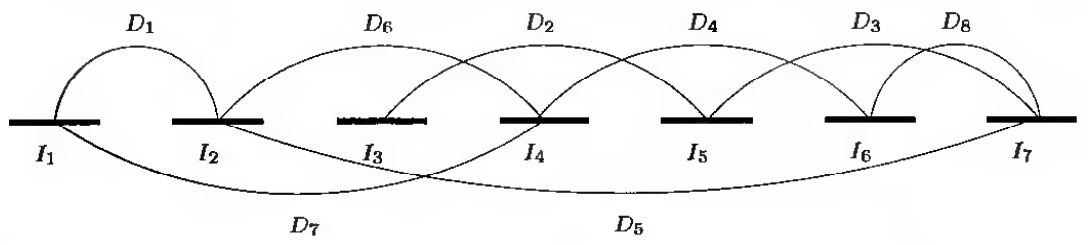


FIG. 6.6 – *Une famille de 2-intervalles à support disjoint.*

de G un sous-ensemble de $\mathcal{F}$; deux sommets sont adjacents si les sous-ensembles correspondants intersectent. Lorsque $\mathcal{F}$ peut être une famille arbitraire, la classe des graphes d'intersection obtenus est l'ensemble des graphes (non orientés). Autrement dit, tout graphe est un graphe d'intersection [Mar45].

Des problèmes plus intéressants apparaissent lorsque certaines restrictions sont imposées sur la famille $\mathcal{F}$ et/ou G. Plus formellement, soient $\mathfrak{G}$ un ensemble de graphes et Δ un ensemble d'ensembles. Nous notons $\mathcal{F} \cong \Omega(\Delta)$ si tout graphe $G \in \mathfrak{G}$ est isomorphe à un graphe d'intersection $G' = \Omega(\mathcal{F})$ pour un famille de sous-ensembles $\mathcal{F}$ de Δ et, réciproquement, tout graphe $G' = \Omega(\mathcal{F})$ pour une famille de sous-ensembles $\mathcal{F}$ de Δ est isomorphe à un graphe G de $\mathfrak{G}$. Un ensemble de graphes $\mathfrak{G}$ est une *classe d'intersection* [Sch85]s'il existe un ensemble d'ensembles Δ tel que $\mathfrak{G} \cong \Omega(\Delta)$.

classe d'intersection

Par exemple, les *graphes triangulés* sont les graphes d'intersection d'une famille de sous-arbres d'un arbre [Gav74b], les *graphes d'intervalle* sont les graphes d'intersection d'une famille d'intervalles sur la droite réelle [Gol80] et les *graphes circulaires* sont les graphes d'intersection d'une famille d'arcs sur un cercle, ...

Nous faisons dans un premier temps quelques brefs rappels sur les graphes d'intervalle. Ensuite, nous présentons les graphes de 2-intervalles, *i.e.* les graphes d'intersection d'une famille de 2-intervalles.

6.4.2 Graphe d'intervalles

Les *graphes d'intervalles* constituent sans soute la classe la plus étudiée des graphes d'intersection. En effet, les graphes d'intervalles ont de nombreuses applications, notamment en ordonnancement, en biologie moléculaire et même en psychologie. L'ouvrage de M.C. Golumbic [Gol80] contient un chapitre consacré aux graphes d'intervalle. Voir également [MM99].

Le graphe d'intersection d'une famille d'intervalles est appelé un *graphe d'intervalles* (c.f. figure 6.7). Si de plus tous les intervalles ont pour longueur 1, le graphe est appelé un *graphe d'intervalles unitaires.* Un *graphe d'intervalles propres* est le graphe d'intersection d'une famille d'intervalles où aucun intervalle n'est strictement inclus dans un autre. F.S. Robert a montré que les classes des graphes d'intervalles unitaires et des graphes d'intervalles propres coïncident.

graphe d'intervalles

Introduisons quelques définitions avant de caractériser les graphes d'intervalles. Un graphe est *triangulé* si tout cycle dont la longueur est strictement plus grande que 3 contient une *corde.* Les graphes triangulés sont importants puisque nous allons voir que tout graphe d'intervalles est triangulé. L'inverse n'est pas vrai. Un graphe

triangulé

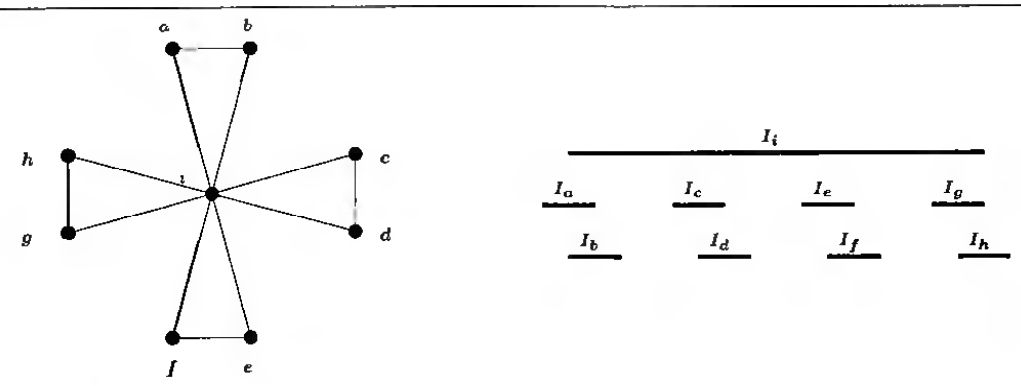


FIG. 6.7 – *À gauche, un graphe d'intervalles, et à droite une représentation.*

$G = (V,E)$ est un *graphe de comparaison* si nous pouvons donner à chaque arête une orientation de sorte que le graphe orienté $G' = (V; E')$ obtenu vérifie : ***comparaison***

$$(u,v) \in E' \wedge (v,w) \in E' \Rightarrow (u,w) \in E'.$$

Une telle orientation est appelée une *orientation transitive*. Soit G un graphe d'ordre n, les *cliques maximales* $M_1, M_2, \ldots, M_m$ constituent un ensemble de cliques qui couvrent tous les sommets de G et m est le plus petit entier possible. Par suite, la *matrice des cliques maximales* de G est une matrice $A = [a_{i,j}]$ dont les lignes correspondent aux cliques maximales et les colonnes aux sommets de G ; $a_{i,j} = 1$ si et seulement si la clique M_i contient le sommet u_j. Une (0,1)-matrice possède la *propriété des 1 consécutifs pour les colonnes* s'il existe une permutation des lignes telle que, pour chaque colonne, les 1 apparaissent consécutivement. Par exemple, la matrice A suivante a la propriété des 1 consécutifs : ***orientation transitive*** ***clique maximale*** ***matrice des cliques maximales*** ***propriété des 1 consécutifs***

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

THÉORÈME 6.4.1 ([GH64]) *Soit $G = (V,E)$ un graphe. Les conditions suivantes sont équivalentes.*

1. *G est un graphe d'intervalles;*
2. *G est un graphe triangulé et son complément $\overline{G}$ est un graphe de comparaison;*
3. *Les cliques maximales de G peuvent être linéairement ordonnées telles que pour chaque sommet u de G, les cliques maximales contenant u apparaissent consécutivement;*
4. *La matrice des cliques maximales de G a la propriété des 1 consécutifs pour les colonnes.*

À partir de ce résultat, des algorithmes efficaces ont été proposés pour reconnaître les graphes d'intervalles.

THÉORÈME 6.4.2 ([BL76]) *Les graphes d'intervalles sont reconnaissables en temps linéaire.*

PQ-arbre

L'algorithme linéaire de K.S. Booth and G.S. Leuker [BL76] est basé sont l'utilisation de PQ-arbres. Sa mise en œuvre reste toutefois assez délicate et une méthode simplifiée est présentée dans [Hsu92]. Plus récemment, un algorithme beaucoup plus simple - *sans PQ-arbres* - a été proposé dans [HMPV00].

parfait

Le graphes d'intervalles sont des *graphes parfaits* [Ber75]. Aussi, de nombreux problèmes NP-complets dans le cas général sont résolubles en temps polynomial dans le cas des graphes d'intervalles : STABLE, CLIQUE, ISOMORPHISME, k-COLORIAGE, ... [Gol80].

6.4.3 Graphe de t-intervalle

graphe de t-intervalle

Les *graphes de t-intervalle*[7] sont une généralisation des graphes d'intervalles. Un graphe $G = (V,E)$ est un graphe de t-intervalle s'il existe une fonction f qui associe à chaque sommet $u \in V$ un ensemble de t intervalles sur la droite réelle avec $\{u,v\} \in E$ si et seulement si $f_i(u) \cap f_j(u) \neq \emptyset$ pour un couple (i,j), $1 \leq i,j \leq t$. Étant donné un graphe $G = (V,E)$, le *nombre d'intervalles* , noté $i(G)$, est le plus petit entier positif t pour lequel G est un graphe de t-intervalle. La fonction f est appelée une t-représentation de G. Les graphes d'intervalles sont donc exactement les graphes G pour lesquels $i(G) = 1$. De même, les graphes d'arc circulaire vérifient $i(G) \leq 2$.

nombre d'intervalles

t-représentation

Une attention considérable a été portée à l'estimation du nombre d'intervalles de certaines classes de graphes.

THÉORÈME 6.4.3 ([TH79]) *Si G est un graphe d'ordre n, alors* $i(G) \leq \lceil \frac{n}{3} \rceil$

THÉORÈME 6.4.4 ([TH79]) $i(K_{m,n}) = \left\lceil \frac{mn+1}{m+n} \right\rceil$

caterpillar

W.T. Trotter et F. Harary [TH79] donnent un résultat précis lorsque G est un arbre. Rappelons qu'un arbre T est un *caterpillar* si l'arbre T' obtenu en supprimant toutes les feuilles de T est un chemin[8]. Si T est un arbre, alors $i(T) = 1$ si T est un caterpillar, et $i(T) = 2$ sinon.

En 1980, M. C. Golumbic [Gol80] proposait le problème suivant : caractériser les graphes G pour lesquels $i(G) = k$, où k est une constante supérieure ou égale à 2. En 1984, D.B. West et D.B. Schmoys [WS84] ont montré que la reconnaissance des graphes G tels que $i(G) = k$, $k \geq 2$, est un problème NP-complet. Sauf si P = NP, il n'existe donc aucune caractérisation «*simple*» des graphes d'intervalles multiples ($k \geq 2$).

Étant donnée une famille $\mathcal{F} = \{D_i \mid 1 \leq j \leq m\}$ de 2-intervalles, le graphe d'intersection $G = \Omega(\mathcal{F})$ est un graphe de 2-intervalles. Plus précisément, G est un graphe de 2-intervalles pour le modèle de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsupset\!\!\!\sqsubset\}$; c'est-à-dire que deux 2-intervalles D_i et D_j de $\mathcal{F}$ s'intersectent ($D_i \cap D_j = \emptyset$) s'ils sont incomparables par les relations $<$, $\sqsubset$ et $\sqsupset\!\!\!\sqsubset$:

$$\forall D_i,D_j \in \mathcal{F}, \qquad D_i \cap D_j = \emptyset \quad \text{si et seulement si} \quad D_i \parallel_R D_j \text{ pour } R \in \{<, \sqsubset, \sqsupset\!\!\!\sqsubset\}$$

Nous pouvons définir des sous-classes des graphes de 2-intervalles en restreignant le modèle d'intersection. En d'autres termes, nous considérons que deux 2-intervalles

7. ou *graphe d'intervalles multiples*.

8. T est un caterpillar si et seulement si T ne contient pas une sous-division de $K_{1,3}$ en sous-arbre.

s'intersectent s'il sont incomparables pour toutes les relations du modèle d'intersection $\mathcal{R}$. Par exemple, le graphe d'intersection d'une famille de 2-intervalles pour le modèle de comparaison $\mathcal{R} = \{<\}$ est simplement un graphe d'intervalles.

Étant donnée une famille $\mathcal{F}$ de 2-intervalles, le graphe $G = \Omega(\mathcal{F})$ est dit à *support disjoint* si $\mathcal{F}$ est une famille de 2-intervalles à support disjoint; il est dit à *support unitaire* si $\mathcal{F}$ est à support unitaire. Si $G = \Omega(\mathcal{F})$ est un graphe de 2-intervalles à support disjoint, alors il est à support unitaire. En effet, puisque tous les intervalles du support sont disjoints, nous pouvons toujours supposer qu'ils sont de longueur 1. L'inverse n'est pas vrai.

support disjoint

support unitaire

6.5 Sous-famille de 2-intervalles disjoints

6.5.1 Introduction

Nous étudions dans cette section le problème de la recherche d'une sous-famille de cardinalité maximale de 2-intervalles $\mathcal{R}$-comparables (pour un modèle de comparaison $\mathcal{R}$ donné). Autrement dit, étant donnés une famille $\mathcal{F}$ de 2-intervalles et un modèle de comparaison non vide $\mathcal{R} \subseteq \{<, \sqsubset, \sqsubset\!\!\sqsubset\}$, il s'agit de calculer une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de cardinalité maximale telle que pour tout $D_i, D_j \in \mathcal{F}'$, il existe une relation $R \in \mathcal{R}$ pour laquelle nous ayons $D_i \sim_R D_j$. Nous appelons SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES ce problème sous sa forme générale dont voici l'énoncé.

> PROBLÈME 6.5.1 (SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES)
> DONNÉES : Une famille $\mathcal{F} = \{D_i \mid 1 \leq i \leq m\}$ de 2-intervalles, un entier positif K et un modèle de comparaison $\mathcal{R}$.
> QUESTION : Existe-t-il une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$, $|\mathcal{F}'| = K$, de 2-intervalles $\mathcal{R}$-comparables.

Nous étudions dans la suite ce problème pour différents modèles de comparaison $\mathcal{R}$ obtenus à partir des combinaisons des relations $<$, $\sqsubset$ et $\sqsubset\!\!\sqsubset$ sur les 2-intervalles. Nous avons déjà vu que le modèle de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsubset\!\!\sqsubset\}$ est directement associé aux graphes de 2-intervalles. Nous montrons en particulier dans les sous-sections suivantes les résultats suivants :

- Le problème SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES est NP-complet pour les modèles de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsubset\!\!\sqsubset\}$ et $\mathcal{R} = \{\sqsubset, \sqsubset\!\!\sqsubset\}$. Le résultat pour le premier modèle de comparaison implique que le problème STABLE est NP-complet même si G est un graphe de 2-intervalles.
- Le problème SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES est résoluble en temps polynomial pour les modèles de comparaison $\mathcal{R} = \{<, \sqsubset\}$, $\mathcal{R} = \{<\}$, $\mathcal{R} = \{\sqsubset\}$ et $\mathcal{R} = \{\sqsubset\!\!\sqsubset\}$. Par exemple, le cas $\mathcal{R} = \{<\}$ est équivalent à la recherche d'un ensemble stable de cardinalité maximale dans un graphe d'intervalles.

Nous malheureusement n'avons pas été en mesure de déterminer la complexité du problèmes SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES pour le modèle de comparaison $\mathcal{R} = \{<, \sqsubset\!\!\sqsubset\}$. Nous igorons également la complexité du problème SOUS-FAMILLE

Modèle de comparaison	Complexité	Commentaires
$\mathcal{R} = \{<, \sqsubset, \sqsupset\}$	NP-complet	Le problème est NP-complet même si $\mathcal{F}$ est une famille à support unitaire. Cependant, il est résoluble en temps polynomial si $\mathcal{F}$ est une famille de 2-intervalles à support disjoint.
$\mathcal{R} = \{\sqsubset, \sqsupset\}$	NP-complet	Le problème est NP-complet même si $\mathcal{F}$ est une famille à support unitaire. Il est polynomialement équivalent au problème $\{\sqsubset, \sqsupset\}$-COUPLAGE si $\mathcal{F}$ est une famille de 2-intervalles à support disjoint.
$\mathcal{R} = \{<, \sqsubset\}$	$O(n^2)$	
$\mathcal{R} = \{<, \sqsupset\}$	?	Le problème est résoluble en temps polynomial si $\mathcal{F}$ est une famille de 2-intervalles $\{<, \sqsubset, \sqsupset\}$-comparables. Il est polynomialement équivalent au problème $\{<, \sqsupset\}$-COUPLAGE si $\mathcal{F}$ est une famille à support disjoint.
$\mathcal{R} = \{<\}$	$O(n^2)$	
$\mathcal{R} = \{\sqsubset\}$	$O(n^2)$	
$\mathcal{R} = \{\sqsupset\}$	$O(n^2 \log n)$	

FIG. 6.8 – *Complexité de la recherche d'une sous-famille de cardinalité maximale pour différents modèles de comparaison $\mathcal{R}$.*

2-INTERVALLES $\mathcal{R}$-COMPARABLES pour le modèle d'intersection $\{\sqsubset, \sqsupset\}$ lorsque $\mathcal{F}$ est une famille de 2-intervalles à support disjoint. Le tableau de la figure 6.8 résume la complexité de ce problème pour différents modèles de comparaison.

6.5.2 Sous-famille de 2-intervalles $\mathcal{R} = \{<, \sqsubset, \sqsupset\}$-comparables

Nous commençons par le cas le plus général. Nous montrons que SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES est un problème NP-complet pour le modèle de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsupset\}$. Autrement dit, sauf si P = NP, il n'existe aucun algorithme en temps polynomial qui calcule une sous-famille de cardinalité maximale de 2-intervalles $\{<, \sqsubset, \sqsupset\}$-comparables.

PROPOSITION 6.5.2 SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \sqsupset\}$-COMPARABLES *est un problème* NP*-complet.*

PREUVE

Le problème SOUS-FAMILLE 2-INTERVALLES $(<, \sqsubset, \sqsupset)$-COMPARABLES appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème EXACT 3SAT donnée par une formule booléenne ϕ sous forme normale conjonctive d'ordre 3, c'est-à-dire avec exactement trois littéraux par clauses. Nous notons $C_1, C_2, \cdots, C_m$

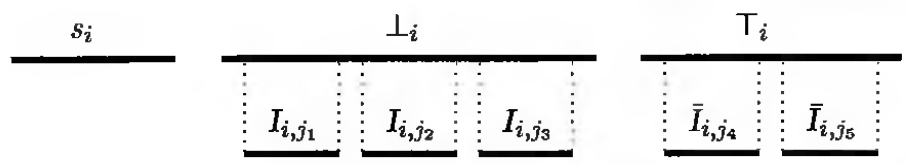


FIG. 6.9 – *Intervalles associés à une variable booléenne x_i dans le cas particulier où $N(x_i) = 3$ et $N(\overline{x}_i) = 2$: le littéral x_i apparaît dans les clauses C_{j_1}, C_{j_2} et C_{j_3}, tandis que le littéral $\overline{x}_i$ apparaît dans les clauses C_{j_4} et C_{j_2}.*

les clauses et $x_1, x_2, \ldots, x_n$ les variables booléennes. Dans la suite de la preuve, nous notons $N(x_i)$ et $N(\overline{x}_i)$ respectivement le nombre de fois où la variable x_i et sa négation apparaissent parmi les clauses. Pour chaque variable booléenne x_i, nous définissons les ensembles d'intervalles disjoints deux à deux $\mathcal{I}(x_i)$ et $\mathcal{I}(\overline{x}_i)$:

$$\mathcal{I}(x_i) = \{I_{i,j} \mid x_i \in C_j\}$$
$$\mathcal{I}(\overline{x}_i) = \{\overline{I}_{i,j} \mid \overline{x}_i \in C_j\}.$$

Remarquons que les ensembles $\mathcal{I}(x_i)$ et $\mathcal{I}(\overline{x}_i)$, $1 \leq i \leq n$, contiennent respectivement $N(x_i)$ et $N(\overline{x}_i)$ intervalles. De plus, toujours pour chaque variable booléenne x_i, nous définissons trois intervalles disjoints notés s_i, $\perp_i$ et $\top_i$ tels que :

1. $s_i < \perp_i < \top_i$
2. $I_{i,j} \subset \perp_i$ pour tout $I_{i,j} \in \mathcal{I}(x_i)$
3. $\overline{I}_{i,j} \subset \top_i$ pour tout $\overline{I}_{i,j} \in \mathcal{I}(\overline{x}_i)$

La figure 6.9 illustre cette construction pour une variable booléenne x_i. Tous ces intervalles sont placés sur la droite réelle en respectant l'ordre de précédence $\perp_i < s_j$ pour $1 \leq i < j \leq n$. Enfin, définissons un ensemble de m intervalles disjoints $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ tels que $c_1 < c_2 < \cdots < c_m < s_1$.

Nous définissons maintenant l'instance correspondante du problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \sqsupset\}$-COMPARABLES. Elle est donnée par une famille $\mathcal{F}$ de 2-intervalles et par un entier positif K. Pour chaque variable booléenne x_i, la famille $\mathcal{F}$ contient les 2-intervalles $(s_i, \perp_i)$ et $(s_i, \top_i)$. Remarquons que l'intervalle s_i joue ici un rôle de «*sélecteur*». En effet, une sous-famille de 2-intervalles $\{<, \sqsubset, \sqsupset\}$-comparables ne peut contenir à la fois les 2-intervalles $(s_i, \perp_i)$ et $(s_i, \top_i)$ puisqu'ils s'intersectent en s_i. Nous désignons les littéraux de la clause C_i par $\lambda_{i,k}$, $1 \leq k \leq 3$. Pour chaque clause $C_i = (\lambda_{i,1} \vee \lambda_{i,2} \vee \lambda_{i,3})$, la famille $\mathcal{F}$ contient les trois 2-intervalles (c_i, I_1), (c_i, I_2) et (c_i, I_3) où les intervalles I_k, $1 \leq k \leq 3$, sont définis par $I_k = I_{j,i}$ si $\lambda_{i,k} = x_j$ et $I_k = \overline{I}_{j,i}$ si $\lambda_{i,k} = \overline{x}_j$ pour $1 \leq k \leq 3$. Remarquons que l'intervalle c_i joue lui aussi un rôle de «*sélecteur*» puisque toute sous-famille de 2-intervalles $\{<, \sqsubset, \sqsupset\}$-comparables ne peut contenir à la fois les 2-intervalles (c_i, I_1), (c_i, I_2) et (c_i, I_3) puisqu'ils intersectent en c_i. La figure 6.13 illustre cette construction pour la formule booléenne $\phi = (x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$. Enfin, nous définissons l'entier positif $K = n + m$. Nous vérifions facilement que la construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une interprétation satisfaisante f de la formule booléenne ϕ. Sans perte de généralité, nous pouvons supposer que chaque clause est satisfaite

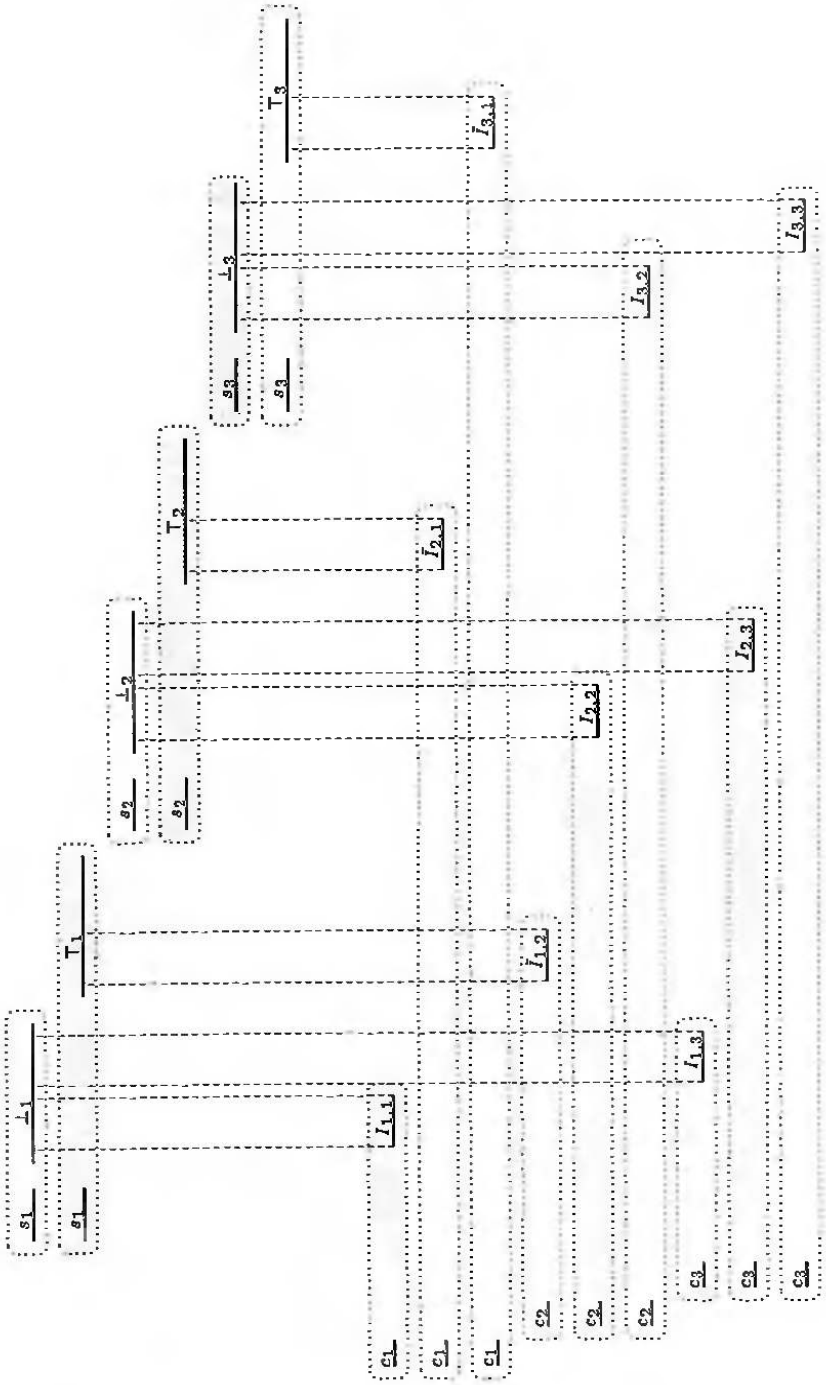


FIG. 6.10 – *Construction utilisée dans la preuve de la proposition 6.5.2 pour la formule booléenne* $\phi = (x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$. *Les boites grisées donnent une interprétation satisfaisante* f *de* ϕ : $f(x_1) =$ *vrai*, $f(x_2) =$ *vrai et* $f(x_3) =$ *faux. Remarquons que le choix* $f(x_3) =$ *vrai est également possible.*

par son premier littéral. Considérons la sous-famille $\mathcal{F}' \subset \mathcal{F}$ définie par :

- Pour chaque variable booléenne x_i, la sous-famille $\mathcal{F}'$ contient le 2-intervalle $(s_i,\top_i)$ si $f(x_i) =$ vrai et $(s_i,\bot_i)$ si $f(x_i) =$ faux;
- Pour chaque clause $C_i = (\lambda_{i,1} \vee \lambda_{i,2} \vee \lambda_{i,3})$, La sous-famille $\mathcal{F}'$ contient le 2-intervalle $(c_i,I_{j,i})$ si $\lambda_{i,1} = x_j$ et le 2-intervalle $(c_i,\overline{I}_{j,i})$ si $\lambda_{i,1} = \overline{x}_j$ (rappelons que chaque clause est satisfaite par son premier littéral).

La sous-famille $\mathcal{F}'$ contient un 2-intervalle pour chaque variable booléenne et un 2-intervalle pour chaque clause. En conséquence, nous avons $|\mathcal{F}'| = m+n$. Remarquons maintenant que les 2-intervalles de la sous-famille $\mathcal{F}'$ correspondant aux variables booléennes sont tous $\{<, \sqsubset, \sqsupset\}$-comparables. De plus, pour chaque clause C_i, la sous-famille $\mathcal{F}'$ contient un 2-intervalle de la forme (c_i,I). Si $I = I_{j,i}$ (resp. $I = \overline{I}_{j,i}$) alors le littéral x_j (resp. $\overline{x}_j$) satisfait la clause C_i, et par suite $(s_j,\top_j) \in \mathcal{F}'$ (resp. $(s_j,\bot_j) \in \mathcal{F}'$). Mais, $I_{j,i} \cap \top_j = \emptyset$ (resp. $\overline{I}_{j,i} \cap \bot_j = \emptyset$). D'où, $(c_i,I_{j,i}) \cap (s_j,\top_j) = \emptyset$ (resp. $(c_i,\overline{I}_{j,i}) \cap (s_j,\bot_j) = \emptyset$). Aussi, puisque ce raisonnement s'applique pour chaque clause de la formule booléenne ϕ, nous en concluons que les 2-intervalles de la sous-famille $\mathcal{F}'$ sont tous $\{<, \sqsubset, \sqsupset\}$-comparables deux à deux.

Inversement, supposons qu'il existe une sous-famille $\mathcal{F}' \subset \mathcal{F}$, $|\mathcal{F}'| \geq K = m + n$, de 2-intervalles disjoints. Pour chaque variable booléenne x_i, la sous-famille $\mathcal{F}'$ contient au plus un des 2-intervalles $(s_i,\top_i)$ et $(s_i,\bot_i)$. De même, pour chaque clause C_i, la sous-famille $\mathcal{F}'$ contient au plus un des 2-intervalles (c_i,I_1), (c_i,I_2) et (c_i,I_3). Par suite, la sous-famille $\mathcal{F}'$ contient exactement $m+n$ 2-intervalles disjoints. Soit f une interprétation de la formule booléenne ϕ définie par $f(x_i) =$ vrai si $(s_i,\top_i) \in \mathcal{F}'$ et $f(x_i) =$ faux si $(s_i,\bot_i) \in \mathcal{F}'$. Nous vérifions facilement que f est une interprétation satisfaisante de la formule booléenne ϕ. En effet, pour chaque variable booléenne x_i, si $(s_i,\top_i) \in \mathcal{F}'$ alors il n'existe pas dans $\mathcal{F}'$ un 2-intervalle de la forme $(c_j,\overline{I}_{i,j})$ car $\overline{I}_{i,j} \subset \top_i$. $\boxed{6.5.2}$

Un graphe de 2-intervalles $G = \Omega(\mathcal{F})$ est le graphe d'intersection d'une famille de 2-intervalles pour le modèle de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsupset\}$. Autrement dit, deux sommets quelconques de G ne sont pas adjacents si et seulement si les 2-intervalles associés sont $\{<, \sqsubset, \sqsupset\}$-comparables. Aussi, un ensemble stable dans G correspond à une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles $\{<, \sqsubset, \sqsupset\}$-comparables. En utilisant la proposition 6.5.2, le corollaire suivant est immédiat.

COROLLAIRE 6.5.3 STABLE *est un problème* NP*-complet même si G est un graphe de 2-intervalles*

Remarquons que dans la proposition 6.5.2, le problème EXACT 3SAT n'est utilisé que pour simplifier la preuve. Fondamentalement, c'est le problème SAT que nous utilisons, *i.e.* le nombre de littéraux par clause est sans importance. Nous partons de cette remarque pour montrer que le problème reste NP-complet si $\mathcal{F}$ est une famille de 2-intervalles à support unitaire. Nous rappelons dans un premier temps un résultat concernant une version restreinte du problème EXACT 3SAT.

THÉORÈME 6.5.4 ([PAP94]) 3SAT *est un problème* NP*-complet même si chaque variable apparaît au plus trois fois et chaque littéral apparaît au plus deux fois.*

PROPOSITION 6.5.5 SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES *est un problème* NP-*complet même si* $\mathcal{F}$ *est une famille de 2-intervalles à support unitaire.*

PREUVE

Remarquons tout d'abord que dans la preuve de la proposition 6.5.2, tous les intervalles peuvent être de longueur 1, sauf les intervalles $\perp_i$ et $\top_i$ associés à chaque variable x_i. En effet, les intervalles $\perp_i$ et $\top_i$ contiennent respectivement les intervalles de type I_{i,j_k} et $\overline{I}_{i,j_\ell}$. Cependant, l'inclusion n'est pas nécessaire, seule l'intersection est importante dans notre preuve. Il suffit alors de prendre comme problème NP-complet de départ le problème 3SAT restreint aux cas où chaque variable apparaît au plus trois fois et chaque littéral au plus deux fois. La construction des intervalles associés à chaque variable est alors donnée par la figure 6.11. Le reste de la preuve est inchangé. $\boxed{6.5.5}$

Pouvons-nous simplifier d'avantage le problème et maintenir la NP-complétude? En particulier, le problème est-il NP-complet si $\mathcal{F}$ est un famille de 2-intervalles à support disjoint? Nous allons montrer que sous cette hypothèse, le problème est résoluble en temps polynomial.

PROPOSITION 6.5.6 *Le problème* SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES *est résoluble en temps polynomial si* $\mathcal{F}$ *est une famille de 2-intervalles à support disjoint.*

PREUVE

Notons $supp(\mathcal{F}) = \{I_1, I_2, \cdots, I_n\}$. Considérons le graphe $G = (V,E)$ d'ordre $|\mathcal{F}| = n$ où $V = supp(\mathcal{F})$ et $E = \{\{I_i, I_j\} \mid (I_i, I_j) \in \mathcal{F} \text{ ou } (I_j, I_i) \in \mathcal{F}\}$. Autrement dit, chaque sommet représente un intervalle du support et il existe une arête entre deux intervalles du support s'ils forment un 2-intervalle de $\mathcal{F}$. Nous vérifions facilement qu'une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-comparables correspond à un couplage dans le graphe G. Or, un couplage de cardinalité maximale dans G est calculable en temps $O(n^3)$, voir par exemple [Jun99] ou [PS82]. $\boxed{6.5.6}$

Nous retrouverons ce type d'approche pour simplifier l'étude de problèmes sur les graphes de 2-intervalles dans l'annexe B page 233.

6.5.3 Sous-famille de 2-intervalles $\{<, \sqsubset\}$-comparables

Nous nous intéressons dans cette sous-section au cas particulier où deux 2-intervalles D_i et D_j sont comparables s'ils sont comparables par les relations $<$ ou $\sqsubset$. Autrement dit, les 2-intervalles D_i et D_j sont $\{<, \sqsubset\}$-comparables si et seulement si l'une des deux conditions suivantes est satisfaite :

1. $D_i < D_j$ ou $D_j < D_i$;
2. $D_i \sqsubset D_j$ ou $D_j \sqsubset D_i$.

Ce cas particulier est sans doute le plus intéressant du point de vue biologique puisqu'il correspond à une structure secondaire sans pseudo-nœuds d'un ARN. Nous avons déjà signalé que cette restriction (l'absence de pseudo-nœuds) n'est utilisée dans les programmes calculant une structure secondaire optimale d'un ARN que pour garantir une complexité en temps acceptable (le problème est NP-complet en toute généralité [LP00]). Il n'est donc pas vraiment étonnant de voir que le

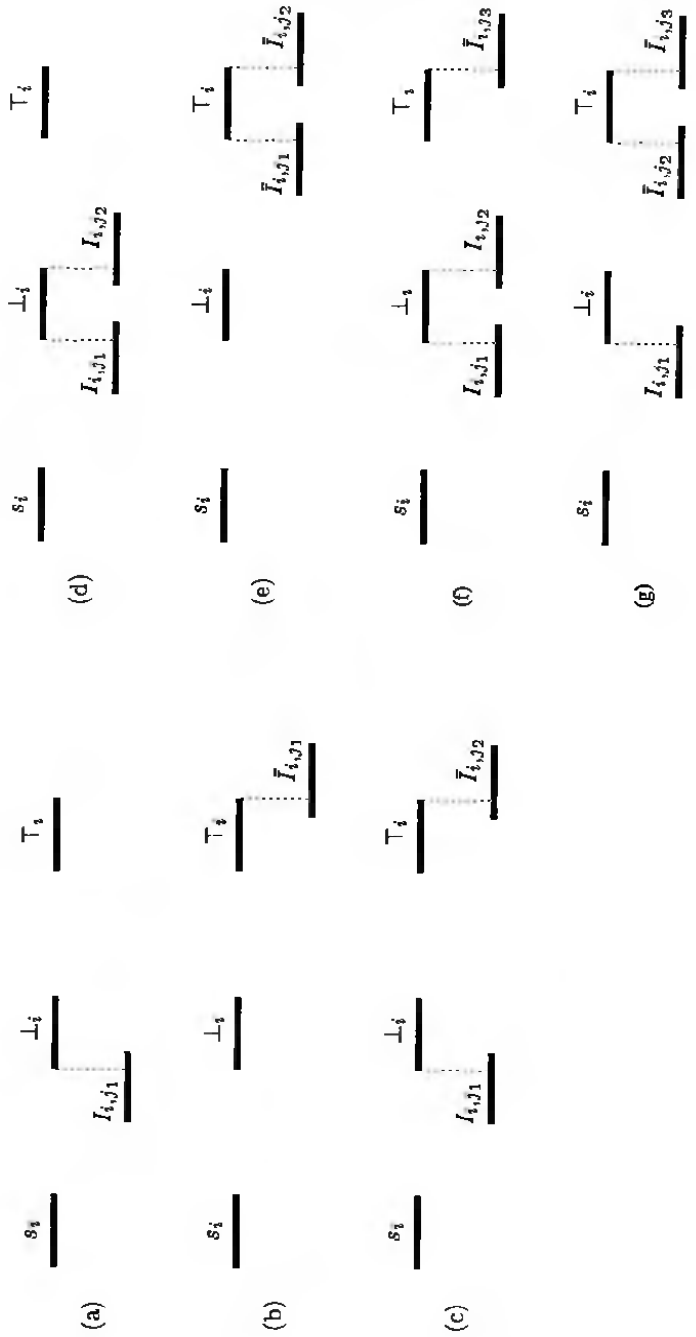


FIG. 6.11 – *Intervalles associés à une variable booléenne x_i dans le cas où chaque variable apparaît au plus trois fois et chaque littéral apparaît au plus deux fois : (a) le littéral x_i apparaît une fois et le littéral $\overline{x}_i$ n'apparaît pas, (b) le littéral x_i n'apparaît pas et le littéral $\overline{x}_i$ apparaît une fois, (c) le littéral x_i apparaît une fois et le littéral $\overline{x}_i$ apparaît une fois, (d) le littéral x_i apparaît deux fois et le littéral $\overline{x}_i$ n'apparaît pas, (e) le littéral x_i n'apparaît pas et le littéral $\overline{x}_i$ apparaît deux fois, (f) le littéral x_i apparaît deux fois et le littéral $\overline{x}_i$ apparaît une fois et (g) le littéral x_i apparaît une fois et le littéral $\overline{x}_i$ apparaît deux fois.*

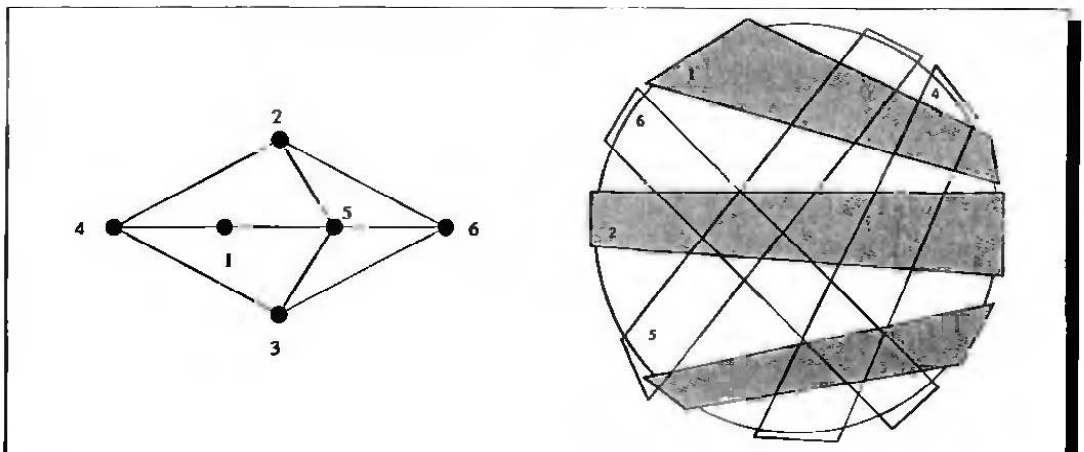


FIG. 6.12 – *À gauche un graphe de trapézoïdes circulaires, et à droite une représentation.*

problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES est résoluble en temps polynomial.

trapézoïde circulaire

graphe de trapézoïdes circulaires

graphe de croisement

Nous introduisons dans un premier temps quelques définitions. Un *trapézoïde circulaire* est la région d'un cercle comprise entre deux cordes qui ne se croisent pas. Le graphe d'intersection d'une famille de trapézoïdes circulaires est appelé un *graphe de trapézoïdes circulaires* [FMW97] (figure 6.12). Afin de montrer que la recherche d'un ensemble stable pondéré maximum est résoluble en temps quadratique pour les graphes de trpézoïdes circulaires, S. Felsner, R. M. Müller et L. Wernisch introduisent une nouvelle classe de graphes appelée *graphes de croisement*. Soit $\mathcal{F} = \{(S_i,T_i) \mid 1 \leq i \leq m\}$ une famille de couples d'intervalles emboîtés : *i.e.* T_i est un sous-intervalle de S_i pour $1 \leq i \leq m$. Un graphe $G = (V,E)$ d'ordre m est un graphe de croisement s'il existe une bijection φ de V dans $[\![m]\!]$ telle que $\{u_i,u_j\} \notin E$ si et seulement si l'une des deux conditions suivantes est satisfaite :

1. $S_{\varphi(u)} < S_{\varphi(v)}$ ou $S_{\varphi(v)} < S_{\varphi(u)}$;
2. $S_{\varphi(u)} \subset T_{\varphi(v)}$ ou $S_{\varphi(v)} \subset T_{\varphi(u)}$.

THÉORÈME 6.5.7 ([FMW97]) *Le calcul d'un ensemble stable pondéré maximum dans un graphe de croisement est résoluble en temps $O(m^2)$, où m est le nombre de couples d'intervalles emboîtés.*

Rappelons que, étant donnée une famille $\mathcal{F} = \{D_i \mid 1 \leq i \leq m\}$ de 2-intervalles, la famille $\widetilde{\mathcal{F}}$ est définie par $\widetilde{\mathcal{F}} = \{\widetilde{D} \mid D \in \mathcal{F}\}$ avec $\widetilde{D} = (ext(D), int(D))$. Nous avons déjà remarqué que $\widetilde{\mathcal{F}}$ est une famille d'intervalles emboîtés : *i.e.* $int(D_i) \subseteq ext(D_i)$ pour $1 \leq i \leq m$. Le lemme suivant est immédiat.

LEMME 6.5.8 *Soit $\mathcal{F} = \{D_i \mid 1 \leq i \leq m\}$ une famille de 2-intervalles. Il existe une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$, $|\mathcal{F}'| = K$, de 2-intervalles $\{<, \sqsubset\}$-comparables si et seulement si il existe un ensemble stable de taille K dans le graphe de croisement de la famille $\widetilde{\mathcal{F}}$.*

Le théorème 6.5.7 et le lemme 6.5.8 nous permettent de conclure quant à la complexité du problème.

COROLLAIRE 6.5.9 *Le problème* SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES *est résoluble en temps* $O(m^2)$, *où* $m = |\mathcal{F}|$.

Remarquons que le corollaire précédent est valable même si nous associons un poids ω à chaque 2-intervalle de la famille $\mathcal{F}$ et si nous recherchons une sous-famille de 2-intervalles $\{<, \sqsubset\}$-comparables dont la somme des poids soit maximale. Cette possibilité nous permet en particulier - dans le cas des structures secondaires d'ARN - d'affecter un poids à chaque 2-intervalle - hélice potentielle - qui reflète par exemple la qualité de ses appariements.

6.5.4 Sous-famille de 2-intervalles $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-comparables

À la différence du problème abordé dans la sous-section précédente, la recherche d'une famille de cardinalité maximale de 2-intervalles $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-comparables est un problème difficile dans le cas général puisque nous allons montrer qu'il est NP-complet.

PROPOSITION 6.5.10 SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES *est un problème* NP-*complet.*

PREUVE

Nous reprenons essentiellement la preuve de la proposition 6.5.2, seul l'ordre de certains intervalles est modifié. Plus précisément :

1. $s_i < \bot_i < \top_i$ pour $1 \leq i \leq n$
2. $s_n < \bot_n < \top_n < \cdots < \bot_2 < \top_2 < \bot_1 < \top_1$
3. $c_1 < c_2 < \cdots < c_m < s_1$.

Remarquons alors que si D_i et D_j sont deux 2-intervalles de la famille $\mathcal{F}$, nous n'avons jamais $D_i < D_j$ ou $D_j < D_i$. Autrement dit, si D_1 et D_2 sont comparables, alors ils sont $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-comparables. Nous pouvons maintenant reprendre la preuve de la proposition 6.5.2 et conclure qu'il existe une sous-famille de $K = m+n$ 2-intervalles $(\sqsubset, \sqsubset\!\!\sqsubset)$-comparables si et seulement si il existe une interprétation satisfaisante de la formule booléenne ϕ. Une illustration de la modification de la preuve est donnée en figure 6.13. $\boxed{6.5.10}$

À l'instar pour la proposition 6.5.5, nous pouvons utiliser une version restreinte du problème EXACT 3SAT et conclure quant à la complexité du problème lorsque $\mathcal{F}$ est une famille de 2-intervalles à support unitaire.

COROLLAIRE 6.5.11 SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES *est un problème* NP-*complet même si* $\mathcal{F}$ *est une famille de* 2-*intervalles à support unitaire.*

Nous ignorons cependant la complexité du problème SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES lorsque $\mathcal{F}$ est une famille de 2-intervalles à support disjoint. Ce problème est en fait un problème de couplage dans un graphe.

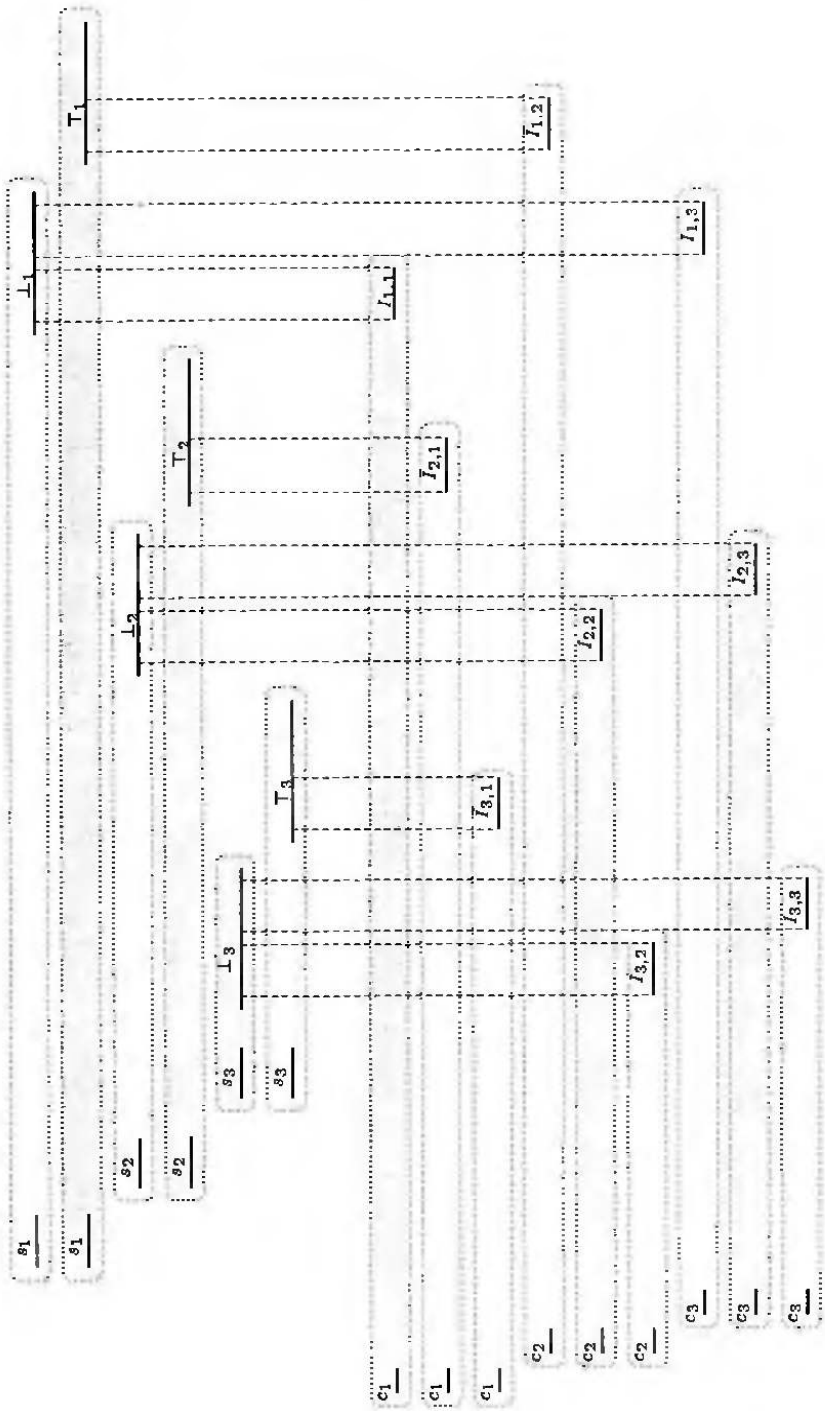


FIG. 6.13 – *Construction utilisée dans la preuve de la proposition 6.5.10 pour la formule booléenne* $\phi = (x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$. *Les boites grisées donnent une interprétation satisfaisante* f *de* ϕ : $f(x_1) =$ *vrai*, $f(x_2) =$ *vrai et* $f(x_3) =$ *faux. Remarquons que le choix* $f(x_3) =$ *vrai est également possible.*

PROBLÈME 6.5.12 ($\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COUPLAGE)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n, une bijection ϕ de V dans $[\![n]\!]$ et un entier positif K.
QUESTION : Existe-t-il un couplage $\mathcal{M} \subseteq E$, $|\mathcal{M}| \geq K$, tel que pour deux arêtes distinctes quelconques $\{u_i,u_j\}$ et $\{u_k,u_\ell\}$ de $\mathcal{M}$, nous n'ayons jamais $\phi(u_i) < \phi(u_j) < \phi(u_k) < \phi(u_\ell)$?

En associant à chaque sommet un intervalle (l'ensemble des intervalles disjoints deux à deux est ordonné sur la droite suivant la bijection ϕ) et à chaque arête un 2-intervalle, nous obtenons l'équivalence entre les problèmes SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES restreint à la classe des familles de 2-intervalles à support disjoint et $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COUPLAGE.

PROPOSITION 6.5.13 *Les problèmes* SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES *restreint aux familles de 2-intervalles à support disjoint et* $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COUPLAGE *sont polynomialement équivalents.*

6.5.5 Sous-famille de 2-intervalles $\{<\}$-comparables

Nous avons déjà remarqué que $(\mathcal{F}, <)$ est un ensemble partiellement ordonné car $<$ est une relation transitive. Par conséquent, nous pouvons calculé une sous-famille de 2-intervalles $\{<\}$-comparables en recherchant une clique dans le graphe de comparaison de l'ensemble partiellement ordonné $(\mathcal{F}, <)$. S. Even, A. Pnueli et A. Lempel ont donné un algorithme quadratique pour la recherche de la clique de cardinalité maximale dans un graphe de comparaison [EPL72]. Un algorithme linéaire pour le même problème est décrit dans [Gol80].En remarquant que nous pouvons construire le graphe de comparaison de $(\mathcal{F}, <)$ en temps quadratique, nous obtenons le résultat suivant.

PROPOSITION 6.5.14 *Le problème* SOUS-FAMILLE 2-INTERVALLES $\{<\}$-COMPARABLES *est résoluble en temps $O(m^2)$, où m est le nombre de 2-intervalles de la famille $\mathcal{F}$.*

Signalons qu'à l'instar du problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES (sous-section 6.5.3), l'algorithme est de même complexité si nous associons un poids ω à chaque 2-intervalle de la famille $\mathcal{F}$. Remarquons que le problème est SOUS-FAMILLE 2-INTERVALLES $\{<\}$-COMPARABLES en temps linéaire si la famille $\mathcal{F}$ est donnée par le graphe de comparaison de $(\mathcal{F}, <)$.

6.5.6 Sous-famille de 2-intervalles $\{\sqsubset\}$-comparables

De même que pour le problème SOUS-FAMILLE 2-INTERVALLES $\{<\}$-COMPARABLES, il s'agit une nouvelle fois de déterminer une clique de cardinalité maximale dans un graphe de comparaison ([EPL72, Gol80]).

PROPOSITION 6.5.15 *Le problème* SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset\}$-COMPARABLES *est résoluble en temps $O(m^2)$, où m est le nombre de 2-intervalles de la famille $\mathcal{F}$.*

Remarquons que le problème SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset\}$-COMPARABLES est résoluble en temps linéaire si la famille $\mathcal{F}$ est donnée par le graphe de comparaison de $(\mathcal{F}, \sqsubset)$. Encore une fois, l'algorithme est de même complexité si nous associons un poids ω à chaque 2-intervalle de la famille $\mathcal{F}$.

6.5.7 Sous-famille de 2-intervalles $\{\sqsupset\}$-comparables

Nous montrons ici que la recherche d'une sous-famille de cardinalité maximale de 2-intervalles $\{\sqsupset\}$-comparables est un problème résoluble en temps polynomial. Nous commençons par quelques définitions et notations.

trapézoïde circulaire

Rappelons qu'un *trapézoïde circulaire* est la region d'un cercle comprise entre deux cordes qui ne se croisent pas. Nous disons que deux trapézoïdes circulaires *se croisent* s'ils s'intersectent *à l'intérieur* du cercle. Nous définissons maintenant une sous-classe des graphes de trapézoïdes circulaires. Un graphe $G = (V,E)$ est un *graphe de croisement de trapézoïdes circulaires* si nous pouvons associer à chaque sommet un 2-intervalle et deux sommets sont adjacents si et seulement si les deux 2-intervalles associés se croisent.

graphe de croisement de trapézoïdes circulaires

REMARQUE 6.5.16 Soit $\mathcal{T}$ un ensemble de trapézoïdes circulaires, $G = (\mathcal{T},E)$ le graphe de trapézoïdes circulaires associé et $G' = (\mathcal{T},E')$ le graphe de croisement de trapézoïdes circulaires associé. Alors, $E' \subseteq E$. ◇

Un exemple est donné en figure 6.14.

LEMME 6.5.17 *Soit $\mathcal{F}$ une famille de 2-intervalles et $G_{\mathcal{F}}$ le graphe de croisement de trapézoïdes circulaires associé. Alors, il existe une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles $\{\sqsupset\}$-comparables deux à deux si et seulement si il existe une clique de taille $|\mathcal{F}'|$ dans $G_{\mathcal{F}}$.*

Une illustration de ce lemme est donnée en figure 6.15. Soient $G = (V,E)$ un graphe de croisement de trapézoïdes circulaires et $u \in V$ un sommet quelconque. Nous notons $G(u)$ le graphe défini sur l'ensemble de sommets $N(u) \cup \{u\}$ où deux sommets sont adjacents si et seulement si les 2-intervalles associés *ne se croisent pas*. Remarquons en particulier que u est toujours un sommet isolé dans $G(u)$.

LEMME 6.5.18 *Soient $G = (V,E)$ un graphe de croisement de trapézoïdes circulaires et $u \in V$ un sommet quelconque. Alors, le graphe $G(u)$ est un graphe trapézoïdal.*

Une illustration de ce lemme est donné en figure 6.16. Soient $G = (V,E)$ un graphe de croisement de trapézoïdes circulaires et $u \in V$ un sommet quelconque. Nous notons $G_{N(u)}$ le sous-graphe induit dans G par l'ensemble $N(u) \cup \{u\}$, c'est-à-dire $G_{N(u)} = G \mid (N(u) \cup \{u\})$.

LEMME 6.5.19 *Soient $G = (V,E)$ un graphe de croisement de trapézoïdes circulaires et $u \in V$ un sommet quelconque. Alors, il existe une clique de taille K dans le graphe induit $G_{N(u)}$ si et seulement si il existe un ensemble stable de taille K dans le graphe trapézoïdal $G(u)$.*

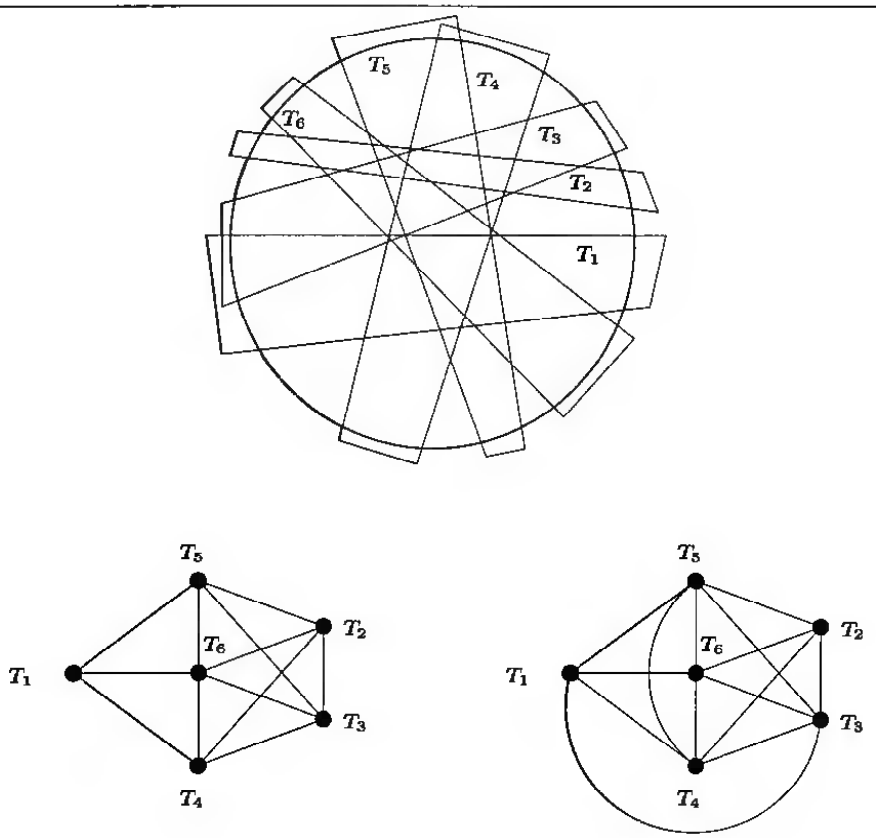


FIG. 6.14 – *En haut, un ensemble de trapézoïdes circulaires. À gauche, le graphe de croisement de trapézoïdes circulaires associé et à droite le graphe de trapézoïdes circulaires associé. Remarquons que le graphe de croisement de trapézoïdes circulaires est obtenus à partir du graphe de trapézoïdes circulaires en supprimant des arêtes. Plus précisément, nous vérifions sur la figure que les trapézoïdes circulaires T_1 et T_3 d'une part et T_4 et T_5 d'autre part s'intersectent mais ne se croisent pas.*

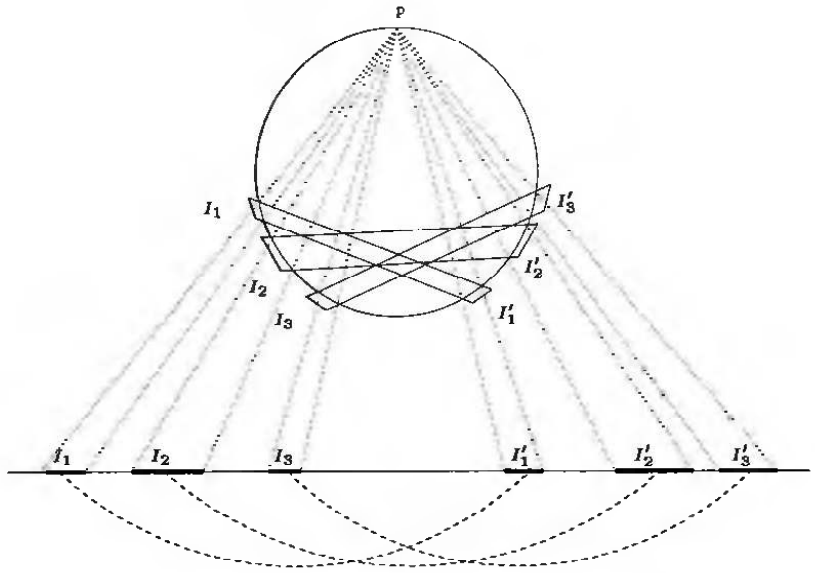


FIG. 6.15 – *Une clique dans un graphe de croisement de trapézoïdes circulaires et une famille de 2-intervalles {⧉}-comparables.*

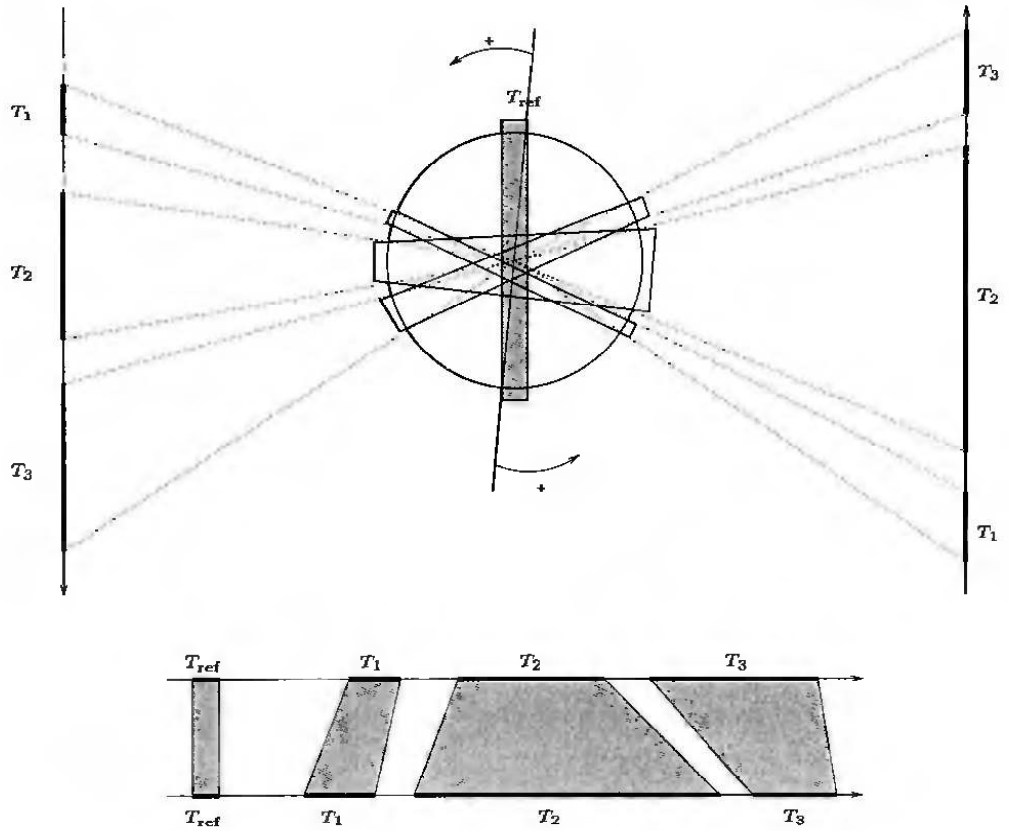


FIG. 6.16 – *Le graphe trapézoïdal $G(u)$.*

En utilisant une représentation géométrique des graphes trapézoïdaux, S. Felsner, R. Müller et L. Wernisch [FMW97] donnent un algorithme en temps $O(n \log n)$ pour la recherche d'un ensemble stable de cardinalité maximale. Le lemme suivant est immédiat.

LEMME 6.5.20 *Soient $G = (V,E)$ un graphe de croisement de trapézoïdes circulaires et $V' \subseteq V$ un sous-ensemble de sommets. Si V' induit une clique dans G, alors V' induit une clique dans $G_{N(u)}$ pour tout $u \in V'$.*

PROPOSITION 6.5.21 *Le problème* CLIQUE *est résoluble en temps $O(n^2 \log n)$ pour les graphes de croisement de trapézoïdes circulaires.*

PREUVE

Nous utilisons l'algorithme 8 ci-après; nous supposons disposer d'une procédure appelée *max-clique-graphe-trapez* qui prend en paramètre un graphe trapézoïdal et qui retourne une clique de cardinalité maximale.

Algorithm 8:

max-clique-graphe-croisement-trapez-circ

Entrée: Un graphe de croisement de trapézoïdes circulaires $G = (V,E)$ d'ordre n.

Sortie: Une clique de cardinalité maximale.

début

1 **pour** $u \in V$ **faire**

2 Construire le graphe trapézoïdal $G(u)$

3 $K_u \leftarrow$ *max-clique-graphe-trapez*$(G(u))$

4 Retourner un ensemble K_u de cardinalité maximale.

fin

Pour chaque sommet $u \in V$, la construction du graphe trapézoïdal $G(u)$ peut s'effectuer en temps $O(n)$ et le calcul d'une clique de cardinalité maximale dans $G(u)$ est en temps $O(n \log n)$ [FMW97]. $\boxed{6.5.21}$

COROLLAIRE 6.5.22 *Le problème* SOUS-FAMILLE 2-INTERVALLES $\{\sqsupset\}$-COMPARABLES *est résoluble en temps $O(n^2 \log n)$..*

6.5.8 Sous-famille de 2-intervalles $\{<, \sqsupset\}$-comparables

Nous n'avons malheureusement pas été en mesure de déterminer la complexité du problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsupset\}$-COMPARABLES dans le cas général. Nous ignorons également la complexité du problème lorsque $\mathcal{F}$ est une famille de 2-intervalles à support unitaire ou disjoint.

Nous montrons tout d'abord que le problème est résoluble en temps polynomial si $\mathcal{F}$ est une famille de 2-intervalles $\{<, \sqsubset, \sqsupset\}$-comparables. Un graphe $G = (V,E)$ est un *graphe d'inclusion* d'une famille $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ de sous-ensembles non vides si (1) $V = \mathcal{F}$ et (2) $\{S_i, S_j\} \in E$ si et seulement si $S_i \subset S_j$ ou $S_j \subset S_i$. Un graphe $G = (V,E)$ est donc un graphe d'inclusion si et seulement si il est un *graphe*

graphe d'inclusion

de comparaison d'un ensemble partiellement ordonné $(X, <)$; *i.e.* (1) $V = X$ et (2) $\{x_i,x_j\} \in E$ si et seulement si $x_i < x_j$ ou $x_j < x_i$. B. Dushnik et W. Miller ont prouvé [DM41] qu'un graphe G est un *graphe d'inclusion d'une famille d'intervalles* si et seulement si G et son complement $\overline{G}$ sont des graphes d'inclusion. Une conséquence de ce résultat est qu'un graphe G est un graphe d'inclusion d'une famille d'intervalles si et seulement si G est un *graphe de permutation.*

PROPOSITION 6.5.23 *Le problème* SOUS-FAMILLE 2-INTERVALLES $\{<, \sqcap\}$-COMPARABLES *est résoluble en temps* $O(m^2)$ *si* $\mathcal{F} = \{D_1,D_2,\ldots,D_m\}$ *est une famille de 2-intervalles* $\{<, \sqsubset, \sqcap\}$*-comparables.*

PREUVE

Considérons la famille d'intervalles $\mathcal{I} = \{ext(D_i) \mid D_i \in \mathcal{F}\}$ et soit $G = (V,E)$ le graphe d'inclusion associé à $\mathcal{I}$. Pour tout $ext(D_i), ext(D_j) \in \mathcal{I}$, $i \neq j$, nous avons $\{ext(D_i), ext(D_j)\} \notin E$ si et seulement si exactement une des conditions suivantes est satisfaite :

1. $D_i < D_j$;
2. $D_j < D_i$;
3. $D_i \sqcap D_j$;
4. $D_j \sqcap D_i$.

Il suffit pour cela de rappeler que $\mathcal{F}$ est une famille de 2-intervalles $\{<, \sqsubset, \sqcap\}$-comparables, c'est-à-dire que les 2-intervalles de $\mathcal{F}$ sont disjoints deux à deux. Par conséquent, une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles $\{<, \sqcap\}$-comparables deux à deux correspond à un ensemble stable dans le graphe G. La construction du graphe G peut s'effectuer en temps $O(m^2)$ et la recherche d'un ensemble stable de cardinalité maximale dans un graphe de permutation[9] peut s'effectuer en temps $O(m \log m)$ [LS93]. $\boxed{6.5.23}$

Nous terminons en donnant un problème de couplage dans un graphe polynomialement équivalent au problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqcap\}$-COMPARABLES lorsque $\mathcal{F}$ est une famille de 2-intervalles à support disjoint.

PROBLÈME 6.5.24 ($\{<, \sqcap\}$-COUPLAGE)
DONNÉES : Un graphe $G = (V,E)$ d'ordre n, une bijection ϕ de V dans $[\![n]\!]$ et un entier positif K.
QUESTION : Existe-t-il un couplage $\mathcal{M} \subseteq E$, $|\mathcal{M}| \geq K$, tel que pour deux arêtes distinctes quelconques $\{u_i,u_j\}$ et $\{u_k,u_\ell\}$ de $\mathcal{M}$, nous n'ayons jamais $\phi(u_i) < \phi(u_k) < \phi(u_\ell) < \phi(u_j)$?

En associant une nouvelle fois à chaque sommet un intervalle (l'ensemble des intervalles disjoints deux à deux est ordonné sur la droite suivant la bijection ϕ) et à chaque arête un 2-intervalle, nous obtenons l'équivalence entre les problèmes

9. Nous avons précédemment vu qu'un graphe est un graphe d'inclusion d'une famille d'intervalles si et seulement il est un graphe de permutation.

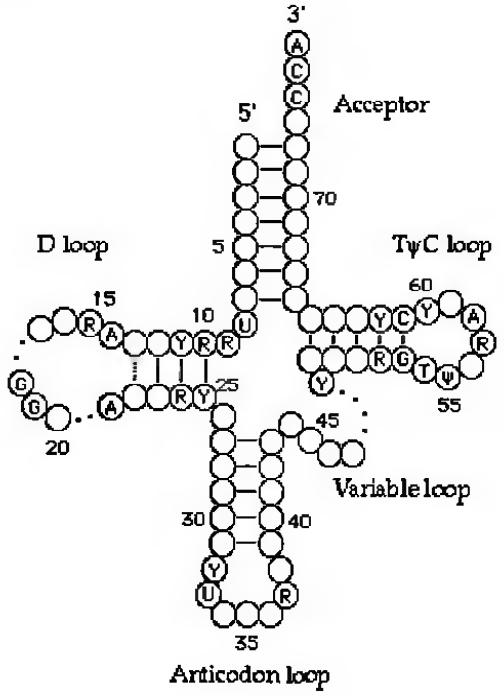


FIG. 6.17 –

SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsupset\}$-COMPARABLES restreint à la classe des familles de 2-intervalles à support disjoint et $\{<, \sqsupset\}$-COUPLAGE.

PROPOSITION 6.5.25 *Les problèmes* SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsupset\}$-COMPARABLES *restreint aux familles de 2-intervalles à support disjoint et* $\{<, \sqsupset\}$-COUPLAGE *sont polynomialement équivalents.*

6.6 Recherche de motifs

6.6.1 Introduction

Nous étudions dans cette section une extension du problème classique de la recherche d'un motif dans un texte au cas des familles de 2-intervalles. La motivation première étant la recherche d'un algorithme de classification automatique des séquences d'ARN suivant la présence ou non d'un «*motif*» particulier dans la structure secondaire associée (c.f. programme `Palingol`). Plus précisément, il s'agit de détecter s'il existe une sous-famille d'hélices potentielles dont l'arrangement correspondrait au motif désiré.

Considérons par exemple la structure secondaire d'un ARN de transfert donnée en figure 6.17. Nous pouvons représenter cette structure par un motif sur l'alphabet $\Sigma = \{a,b,c,d\}$ puisqu'il y a quatre hélices. Nous parcourons la séquence dans le sens 5' - 3' et nous associons à chaque brin d'une même hélice une même lettre. Aussi, le motif associé est ici, par exemple, *abbccdda*. Étant donnés un motif et un ensemble d'hélices potentielles, nous souhaitons calculer (à l'instar d'un programme type `Palingol`) un sous-ensemble d'hélices correspondant au motif. Autrement dit,

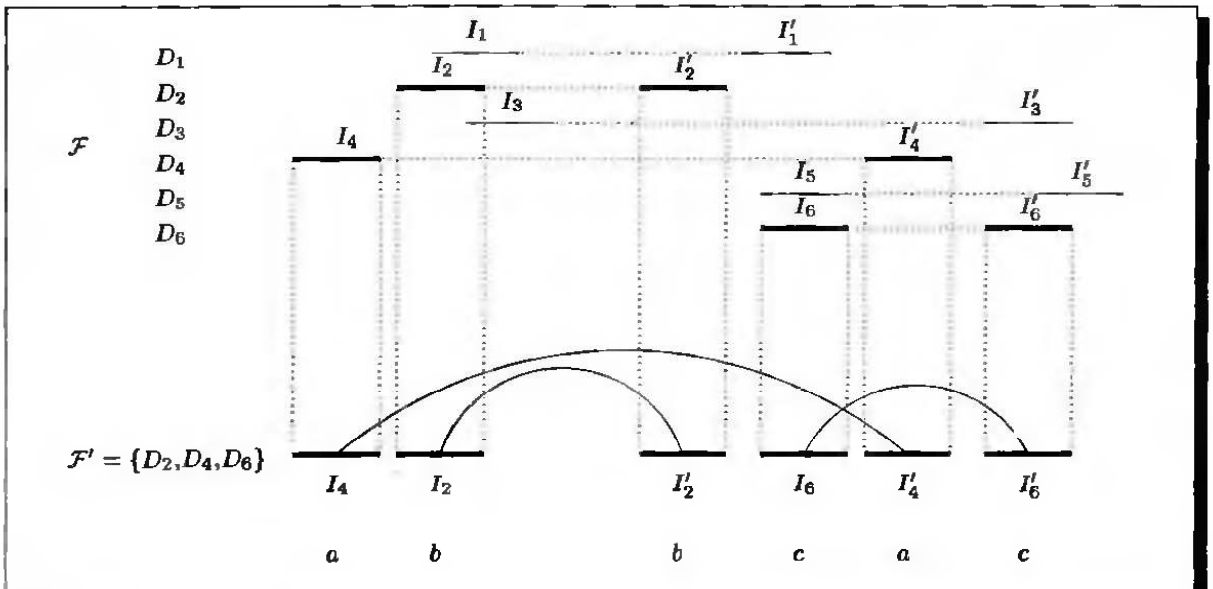


FIG. 6.18 – *Une famille de 2-intervalle et une sous-famille qui correspond au motif abbcac.*

le problème consiste à sélectionner un certain nombre d'hélices dont la structure secondaire induite correspond au motif. Notre but n'est pas ici de proposer une alternative aux programmes type `Palingol`, mais plutôt de présenter quelques résultats partiels de complexité sur le problème sous-jacent. En effet, nous ne considérons pas par exemple les longueurs des hélices, les contraintes structurelles sur les hélices, les distances relatives entre hélices, *etc*... Aussi, nous ne discuterons pas de l'algorithme type *séparation et évaluation* utilisé par les programme type `Palingol`.

Nous commençons par une présentation informelle du problème dans le cadre des familles de 2-intervalles. Considérons la famille de 2-intervalles $\mathcal{F} = \{D_i \mid 1 \leq i \leq 6\}$ représentée dans la partie supérieure de la figure 6.18. Supposons maintenant que nous recherchions un «*motif*» de la forme *abbcac*. Autrement dit, nous voulons associer les lettres a, b et c à trois 2-intervalles disjoints de $\mathcal{F}$ qui vérifient :

- $D(b) \sqsubset D(a)$ car les deux occurrences de la lettre b sont entre la première et la seconde occurrence de la lettre a;
- $D(b) < D(c)$ car les deux occurrences de la lettre b précèdent les deux occurrences de la lettre c;
- $D(a) \sqcap D(c)$ car *acac* est un sous-mot de *abbcac*.

Nous associons la lettre a au 2-intervalle D_4, b à D_2 et c à D_6 et nous vérifions sur la figure 6.18 que :

- $D_2 \sqsubset D_4$;
- $D_2 < D_6$;
- $D_4 \sqcap D_6$.

Nous introduisons dans un premier temps le formalisme nécessaire puis le problème sous sa forme générale. Nous montrons ensuite que le problème est NP-complet dans le cas général (pour le modèle de comparaison $\mathcal{R} = \{<, \sqsubset, \sqcap\}$) et dans le cas du modèle de comparaison $\mathcal{R} = \{\sqsubset, \sqcap\}$.

6.6.2 Définitions

Soit $\mathcal{F} = \{D_i \mid 1 \leq i \leq m\}$ une famille de 2-intervalles $\mathcal{R}$-comparables pour un modèle de comparaison $\mathcal{R}$ donné. Remarquons dans un premier temps que tous les intervalles du support de la famille $\mathcal{F}$ sont comparables par la relation de précédence $<$[10]. Par exemple, sur l'exemple de la figure 6.18, les intervalles du support de la sous-famille $\mathcal{F}'$ satisfont : $I_4 < I_2 < I'_2 < I_6 < I'_4 < I'_6$.

Nous notons *ind* une bijection de $supp(\mathcal{F})$ dans $[\![2m]\!]$ définie par :

$$ind(I) = 1 + \text{card}(\{I' \mid I' \in supp(\mathcal{F}) \text{ et } I' < I\}).$$

La bijection *ind* est bien définie car tous les intervalles du support sont comparables deux à deux par la relation $<$. Intuitivement, la bijection *ind* correspond à une numérotation des intervalles du support ordonnés de gauche à droite. De plus, nous notons $\pi_{\mathcal{F}}$ (ou plus simplement π s'il n'y a pas d'ambiguïté) une application surjective de $supp(\mathcal{F})$ dans $\mathcal{F}$ définie par $\pi(I) = D_i$ si $I \in D_i$. Autrement dit, $\pi(I) = D_i$ si I est un des deux intervalles du 2-intervalle D_i. Enfin, nous notons $\varphi_{\mathcal{F}}$ (ou plus simplement φ s'il n'y a pas d'ambiguïté) l'application surjective de $[\![2m]\!]$ dans $\mathcal{F}$ définie par $\varphi(i) = \pi(I)$ avec $ind(I) = i$ pour $1 \leq i \leq 2m$.

DÉFINITION 6.6.1 (REPRÉSENTANT CANONIQUE) *Soit $\mathcal{F} = \{D_i \mid 1 \leq i \leq m\}$ une famille de 2-intervalles $\mathcal{R}$-comparables pour un modèle de comparaison $\mathcal{R}$ donné. Le* représentant canonique *de la famille $\mathcal{F}$ est le mot $\mu = a_1 a_2 \cdots a_{2m}$ sur l'alphabet $\mathcal{F}$ défini par $a_i = \varphi(i)$ pour $1 \leq i \leq 2m$.* ***représentant canonique***

Afin de définir formellement le motif de recherche, nous introduisons une relation d'équivalence notée $\equiv$. Soient $u = u_1 u_2 \cdots u_n$ et $v = v_1 v_2 \cdots v_n$ deux mots. Les mots u et v sont équivalents modulo $\equiv$, noté $u \equiv v$, s'il existe une bijection f de $\{u_i \mid 1 \leq i \leq |u|\}$ dans $\{v_i \mid 1 \leq v \leq |v|\}$ tel que :

$$f(u_1)f(u_2)\cdots f(u_n) = v.$$

Nous notons $f(u) = v$ en lieu et place de $f(u_1)f(u_2)\cdots f(u_n) = v$. La relation $\equiv$ est une relation d'équivalence et nous notons $[u]_{\equiv}$ la classe d'équivalence modulo $\equiv$ d'un mot u.

DÉFINITION 6.6.2 (REPRÉSENTANT) *Soit $\mathcal{F}$ un famille de 2-intervalles $\mathcal{R}$-comparables pour un modèle de comparaison $\mathcal{R}$ donné. Un mot u est un* représentant *de la famille $\mathcal{F}$ si $u \equiv \mu$, où μ est le représentant canonique de la famille $\mathcal{F}$.* ***représentant***

EXEMPLE 18 *Reprenons l'exemple de la sous-famille $\mathcal{F}' = \{D_2, D_4, D_6\}$ de la figure 6.18. Dans un premier temps, nous avons $supp(\mathcal{F}') = \{I_2, I'_2, I_4, I'_4, I_6, I'_6\}$. La bijection ind est définie par :*

$$\begin{array}{lll} ind(I_4) = 1 & ind(I_2) = 2 & ind(I'_2) = 3 \\ ind(I_6) = 4 & ind(I'_4) = 5 & ind(I'_6) = 6 \end{array}$$

10. Le symbole '$<$' désigne ici une relation entre intervalles.

L'application π est définie par :

$$\pi(I_2) = D_2 \qquad \pi(I'_2) = D_2$$
$$\pi(I_4) = D_4 \qquad \pi(I'_4) = D_4$$
$$\pi(I_6) = D_6 \qquad \pi(I'_6) = D_6$$

Enfin, l'application φ est définie par :

$$\varphi(1) = D_4 \qquad \varphi(2) = D_2 \qquad \varphi(3) = D_2$$
$$\varphi(4) = D_6 \qquad \varphi(5) = D_4 \qquad \varphi(6) = D_6$$

Le représentant canonique associé à la famille $\mathcal{F}'$ est le mot $\mu = D_4D_2D_2D_6D_4D_6$. Le mot $u = abbcac$ est un représentant de $\mathcal{F}'$ en posant $f(a) = D_4$, $f(b) = D_2$ et $f(c) = D_6$. ◇

Un mot u est un représentant d'une famille de 2-intervalles si et seulement si chaque lettre de u apparaît exactement deux fois. Nous disons que le mot u est «*bien formé*» s'il satisfait cette propriété élémentaire. Suivant le modèle de comparaison $\mathcal{R}$ considéré, tous les mots bien formés ne sont pas des représentants d'une famille de 2-intervalles $\mathcal{R}$-comparables. Par exemple, si $\mathcal{R} = \{<, \sqsubset\}$, tous les représentants sont des mots de Dyck.

La recherche d'une occurrence d'un mot u dans une famille $\mathcal{F}$ de 2-intervalles consiste donc à calculer une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles disjoints relativement à un modèle de comparaison $\mathcal{R}$ donné telle que le mot u soit un représentant de la sous-famille $\mathcal{F}'$; *i.e.* u appartient à la classe d'équivalence modulo $\equiv$ du représentant canonique de la sous-famille $\mathcal{F}'$. Nous appelons MOTIF 2-INTERVALLE $\mathcal{R}$-COMPARABLES ce problème.

PROBLÈME 6.6.3 (MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES)
DONNÉES : Une famille $\mathcal{F} = \{D_i \mid 1 \leq i \leq m\}$ de 2-intervalles, un modèle de comparaison $\mathcal{R}$ et un mot u.
QUESTION : Existe-t-il une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles $\mathcal{R}$-comparables de représentant u, *i.e.* $\mu_{\mathcal{F}'} \equiv u$?

EXEMPLE 19 *Reprenons la famille $\mathcal{F} = \{D_i \mid 1 \leq i \leq 6\}$ de la figure 6.18. La sous-famille $\mathcal{F}'$ est solution du problème* MOTIF 2-INTERVALLES $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES *pour le mot $u = abbcac$. La sous-famille $\mathcal{F}'' = \{D_3, D_4\}$ est solution du problème* MOTIF 2-INTERVALLES $\{\sqsubset\}$-COMPARABLES *pour le mot $u = abba$.* ◇

De même que pour la recherche d'une sous-famille de 2-intervalles disjoints de cardinalité maximale, nous étudions la complexité du problème MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES pour plusieurs modèles de comparaison $\mathcal{R}$. Bien sur, la complexité du problème varie encore une fois suivant le modèle de comparaison considéré. Nous montrons que le problème est NP-complet pour les modèles de comparaison $\mathcal{R} = \{<, \sqsubset, \sqsubset\!\!\sqsubset\}$ et $\mathcal{R} = \{\sqsubset, \sqsubset\!\!\sqsubset\}$, et résoluble en temps polynomial pour les modèles de comparaison $\mathcal{R} = \{<\}$, $\mathcal{R} = \{\sqsubset\}$ et $\mathcal{R} = \{\sqsubset\!\!\sqsubset\}$. Cependant, les sous-sections suivantes ne présentent que des résultats partiels. En particulier, la question concernant la complexité du problème MOTIF 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES reste posée.

6.6.3 Sous-famille de 2-intervalles $\mathcal{R}$-comparables et recherche de motifs

Nous pouvons raisonnablement nous demander si les problèmes MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES et SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES ne sont pas deux formulations différentes d'un même problème. C'est en fait le cas pour les modèles de comparaison $\mathcal{R} = \{<\}$, $\mathcal{R} = \{\sqsubset\}$ et $\mathcal{R} = \{\text{⧉}\}$ comme le montre le lemme suivant.

LEMME 6.6.4 *Soient $\mathcal{F}$ une famille de 2-intervalles, $R \in \{<, \sqsubset, \text{⧉}\}$ une relation et u un mot compatible avec R. Alors, il existe une occurrence de u dans $\mathcal{F}$ si et seulement si il existe une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$, $|\mathcal{F}'| = \frac{1}{2}|u|$, de 2-intervalles $\{R\}$-comparables dont le représentant canonique est équivalent à u modulo $\equiv$.*

C'est la raison pour laquelle les problèmes MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES et SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES sont équivalents pour les modèles de comparaison $\mathcal{R} = \{<\}$, $\mathcal{R} = \{\sqsubset\}$ et $\mathcal{R} = \{\text{⧉}\}$.

Cependant, cette équivalence cesse d'être valide dès que $|\mathcal{R}| > 1$. Mais, même si le fait qu'un problème SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES pour un modèle d'intersection $\mathcal{R}$ donné soit NP-complet ne prouve en rien que le problème MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES pour le même modèle d'intersection $\mathcal{R}$ soit également NP-complet, il nous laisse supposer que ce dernier est vraisemblablement tout aussi difficile. C'est en partie la raison pour laquelle la NP-complétude des problèmes MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES pour les modèles de comparaison $\mathcal{R} = \{<, \sqsubset, \text{⧉}\}$ et $\mathcal{R} = \{\sqsubset, \text{⧉}\}$ est - en quelque sorte - un «*résultat attendu*». Concernant le problème MOTIF 2-INTERVALLES $\{<, \sqsubset, \text{⧉}\}$-COMPARABLES, nos résultats tendent même à prouver que le problème est «*plus difficile*» que le problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \text{⧉}\}$-COMPARABLES en ce sens que nous prouverons qu'il est NP-complet même si $\mathcal{F}$ est une famille de 2-intervalles à support disjoint[11].

La complexité du problème MOTIF 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES est une question plus intéressante; le problème pourrait être NP-complet bien que nous ayons déjà remarqué que le problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES soit lui résoluble en temps polynomial.

6.6.4 Modèle de comparaison $\mathcal{R} = \{<, \sqsubset, \text{⧉}\}$

Nous allons montrer que MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES est un problème NP-complet. Autrement dit, sauf si P = NP, il n'existe aucun algorithme en temps polynomial qui recherche une occurrence d'un mot u dans une famille de 2-intervalles. Nous commençons par une notation. Pour tout mot $u = a_1 a_2 \cdots a_n$, nous notons $\widetilde{u}$ le mot inverse : *i.e.* $\widetilde{u} = a_n \cdots a_2 a_1$.

11. Nous avons déjà remarqué (proposition 6.5.6) que le problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \text{⧉}\}$-COMPARABLES est résoluble en temps polynomial si $\mathcal{F}$ est une famille de 2-intervalles à support disjoint. Nous prouverons également dans la suite que MOTIF 2-INTERVALLES $\{\sqsubset, \text{⧉}\}$-COMPARABLES est un problème NP-complet même si $\mathcal{F}$ est une famille de 2-intervalles à support disjoints, mais nous ignorons jusqu'à présent si le problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \text{⧉}\}$-COMPARABLES reste ou non NP-complet sous cette hypothèse.

PROPOSITION 6.6.5 MOTIF 2-INTERVALLES $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES *est un problème* NP-*complet.*

PREUVE

Le problème MOTIF 2-INTERVALLES $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème CLIQUE donnée par un graphe $G = (V,E)$ d'ordre n et par un entier positif K. Notons m la nombre d'arêtes du graphe G, $V = \{u_1,u_2,\ldots,u_n\}$ et $u_i < u_j$ si et seulement si $i < j$. À chaque sommet $u \in V$, nous associons un ensemble $\mathcal{I}(u)$ de $n + 2(m+1)$ intervalles disjoints :

$$\mathcal{I}(u) = \mathcal{A}(u) \cup \mathcal{B}(u) \cup \mathcal{A}'(u)$$

où

$$\begin{aligned}\mathcal{A}_u &= \{A_u(i) \mid 1 \leq i \leq m+1\}\\ \mathcal{B}_u &= \{B_u(i) \mid 1 \leq i \leq n\}\\ \mathcal{A}'_u &= \{A'_u(i) \mid 1 \leq i \leq m+1\}.\end{aligned}$$

Pour tout $u \in V$, les intervalles de $\mathcal{I}(u)$ sont ordonnés sur la droite réelle par les relations suivantes (figure 6.19) :

$$A_u(1) < \cdots < A_u(m+1) < B_u(1) < \cdots < B_u(n) < A'_u(m+1) < \cdots < A'_u(1).$$

De plus,

$$\forall u_i,u_j \in V, i \neq j, \qquad A'_{u_i}(1) < A_{u_j}(1) \quad \text{si et seulement si} \quad i < j.$$

L'instance correspondante du problème MOTIF 2-INTERVALLES $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES est donnée par une famille $\mathcal{F}$ de 2-intervalles et par un mot p. Pour chaque sommet $u \in V$, la famille $\mathcal{F}$ contient les 2-intervalles $(A_u(i),A'_u(i))$ pour $1 \leq i \leq m+1$. Intuitivement, ces 2-intervalles vont nous permettre d'«*isoler*» un certain nombre de sommets. Pour chaque arête $e = \{u_i,u_j\} \in E$, $i < j$, la famille $\mathcal{F}$ contient le 2-intervalle $(B_{u_i}(j),B_{u_j}(i))$. Enfin, le mot p est défini par :

$$p = \alpha_1\beta_1\widetilde{\alpha_1}\alpha_2\beta_2\widetilde{\alpha_2}\cdots\alpha_K\beta_K\widetilde{\alpha_K}$$

où $|\alpha_i| = m+1$ et $|\beta_i| = K-1$ pour $1 \leq i \leq K$. De plus, $\beta_i[j] = \beta_{j+1}[i]$[12] pour $1 \leq i \leq j \leq K-1$. Remarquons que chaque intervalle appartient à au plus un 2-intervalle. De plus, le mot p est *bien formé*. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une clique $V' \subseteq V$ de taille K dans G. Soit $V' = \{u_{i_1},u_{i_2},\cdots,u_{i_K}\}$ cet ensemble. Sans perte de généralité, nous pouvons supposer $i_1 < i_2 < \cdots < i_K$. Notons $E' \subseteq E$ l'ensemble des arêtes de la clique, *i.e.* $E' = \{\{u_i,u_j\} \mid u_i \in V' \text{ et } u_j \in V'\}$. Considérons la sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles définie de la façon suivante :

1. pour chaque sommet $u \in V'$, la sous-famille $\mathcal{F}'$ contient les 2-intervalles $(A_u(i),A'_u(i))$ pour $1 \leq i \leq m+1$;

12. $\beta_i[j]$ (resp. $\beta_{j+1}[i]$) désigne la j^{eme} (resp. i^{eme}) lettre du mot β_i (resp. β_{j+1}).

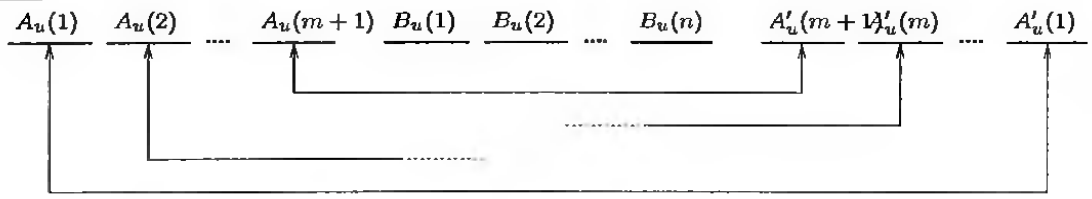


FIG. 6.19 – *Illustration de la construction utilisée dans la preuve de la proposition 6.6.5. À chaque sommet $u \in V$ est associé un ensemble $\mathcal{I}(u)$ de $n + 2(m+1)$ intervalles. Les $m+1$ premiers intervalles constituent l'ensemble $\mathcal{A}_u$, les n intervalles suivants l'ensemble $\mathcal{B}_u$, et les $m+1$ derniers intervalles l'ensemble $\mathcal{A}'_u$. Les 2-intervalles sont représentés par des doubles flèches.*

2. pour chaque arête $\{u_i,u_j\} \in E'$, $i < j$, la sous-famille $\mathcal{F}'$ contient le 2-intervalle $(B_{u_i}(j),B_{u_j}(i))$.

Nous vérifions $|\mathcal{F}'| = K(m+1) + \frac{1}{2}K(K-1)$ car nous associons à chaque sommet de la clique un ensemble de $m+1$ 2-intervalles et à chaque arête de la clique un 2-intervalle. De plus, par construction, le représentant canonique μ associé à la sous-famille $\mathcal{F}'$ est de la forme :

$$\mu = \alpha_1\beta_1\widetilde{\alpha_1}\alpha_2\beta_2\widetilde{\alpha_2}\cdots\alpha_K\beta_K\widetilde{\alpha_K}$$

où $|\alpha_i| = m+1$ pour $1 \leq i \leq K$. De plus, $|\beta_i| = K-1$ pour $1 \leq i \leq K$ car V' est une clique de taille K dans G. Montrons que le représentant canonique μ vérifie également $\beta_i[j] = \beta_{j+1}[i]$ pour $1 \leq i \leq j \leq K-1$. Soit $\{u_i,u_j\} \in E'$ avec $i < j$. Alors, nous avons :

$$\pi(B_{u_i}(j)) = \pi(B_{u_j}(i))$$

car $(B_{u_i}(j),B_{u_j}(i))$ est un 2-intervalle de $\mathcal{F}'$, et *a fortiori* de $\mathcal{F}$. Par suite,

$$\varphi(ind(B_{u_i}(j))) = \varphi(ind(B_{u_j}(i))).$$

Or, il est facile de vérifier que nous avons les égalités :

$$\varphi(ind(B_{u_i}(j))) = \beta_i[j]$$
$$\varphi(ind(B_{u_j}(i))) = \beta_{j+1}[i].$$

Par conséquent, $\beta_i[j] = \beta_{j+1}[i]$. Nous en concluons que les mots p et μ sont équivalents modulo $\equiv$.

Inversement, supposons qu'il existe une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles $\{<, \sqsubset, \sqsubset\!\!\sqsubset\}$-comparables telle que $\mu \equiv p$, où μ est le motif canonique associé à la sous-famille $\mathcal{F}'$. Remarquons dans un premier temps que $|\alpha_i| = |\widetilde{\alpha_i}| = m+1$ pour $1 \leq i \leq K$. Or, il n'y a que m arêtes dans le graphe G. Par conséquent, chaque sous-mot $\alpha_i\widetilde{\alpha_i}$, $1 \leq i \leq K$, correspond à un ensemble de $m+1$ 2-intervalles associés à un même sommet $u \in V$, *i.e.* les 2-intervalles $(A_u(i),A'_u(i))$ pour $1 \leq i \leq m+1$. Nous notons $V' \subseteq V$, $|V'| = K$, le sous-ensemble de sommets dont les 2-intervalles sont associés à un sous-mot $\alpha_i\widetilde{\alpha_i}$, $1 \leq i \leq K$. Par conséquent, chaque couple de lettres $(\beta_i[j],\beta_{j+1}[i])$, $1 \leq i \leq K$ et $1 \leq j \leq K-1$, correspond à une arête du graphe

G entre deux sommets de V'. Puisque $|\beta_i| = K - 1$ pour $i \leq i \leq K$, l'ensemble V' induit un graphe complet d'ordre K dans G. $\boxed{6.6.5}$

REMARQUE 6.6.6 Le problème MOTIF 2-INTERVALLES $\{<, \sqsubset, \sqsupset\}$-COMPARABLES est NP-complet même si $\mathcal{F}$ est une famille de 2-intervalles à support disjoint. ◇

6.6.5 Modèle de comparaison $\mathcal{R} = \{\sqsubset, \sqsupset\}$

Nous nous intéressons ici au cas particulier où deux 2-intervalles sont comparables s'ils sont comparables par les relations $\sqsubset$ ou $\sqsupset$. Nous montrons ici que le problème de recherche associé est NP-complet en utilisant une nouvelle fois une réduction depuis le problème CLIQUE. Introduisons tout d'abord une notation : étant donnés deux ensembles d'intervalles $\mathcal{I}$ et $\mathcal{I}'$, nous notons $\mathcal{I} < \mathcal{I}'$ si et seulement si chaque intervalle de $\mathcal{I}$ précède tous les intervalles de $\mathcal{I}'$: $\mathcal{I} < \mathcal{I}'$ si et seulement si $I < I'$ pour tout $I \in \mathcal{I}$ et $I' \in \mathcal{I}'$

PROPOSITION 6.6.7 MOTIF 2-INTERVALLES $\{\sqsubset, \sqsupset\}$-COMPARABLES *est un problème* NP-*complet.*

PREUVE

Le problème MOTIF 2-INTERVALLES $\{\sqsubset, \sqsupset\}$-COMPARABLES appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème CLIQUE donnée par un graphe $G = (V,E)$ d'ordre n et par un entier positif K. Nous notons $V = \{u_1, u_2, \cdots, u_n\}$ l'ensemble des sommets et $u_i < u_j$ si et seulement si $i < j$. De même, nous notons $E = \{e_1, e_2, \cdots, e_m\}$ et $e_i < e_j$ si et seulement si $i < j$. L'ordre sur les sommets et les arêtes est arbitraire. Introduisons maintenant les différents éléments de notre construction. À chaque sommet $u_i \in V$, nous associons un ensemble $\mathcal{I}_i$ de $m + 2$ intervalles disjoints :

$$\mathcal{I}_i = \{I_{i,j} \mid 0 \leq j < m + 2\}.$$

Pour des raisons de commodité, nous notons $\mathcal{I}_V$ l'ensemble des intervalles associés aux sommets du graphe G :

$$\mathcal{I}_V = \bigcup_{u_i \in V} \mathcal{I}_i.$$

Nous ordonnons les intervalles de chaque ensemble $\mathcal{I}_i$, $1 \leq i \leq n$, par les relations :

$$I_{i,0} < I_{i,1} < \cdots < I_{i,m+1}.$$

Les intervalles de $\mathcal{I}_V$ sont complètement ordonnés par les relations :

$$\mathcal{I}_1 < \mathcal{I}_2 < \cdots < \mathcal{I}_n.$$

À chaque arête $e_i \in E$, nous associons un ensemble $\mathcal{I}'_i$ de $n + 2$ intervalles disjoints :

$$\mathcal{I}'_i = \{J_{i,j} \mid 0 \leq j \leq n + 1\}.$$

Nous notons $\mathcal{I}_E$ l'ensemble des intervalles associés aux arêtes de G :

$$\mathcal{I}_E = \bigcup_{e_i \in V} \mathcal{I}'_i.$$

Nous ordonnons les intervalles de chaque ensemble $\mathcal{I}'_i$, $1 \leq i \leq m$, par les relations :

$$J_{i,0} < J_{i,1} < \cdots < J_{i,n+1}.$$

Les intervalles de $\mathcal{I}_E$ sont complètement ordonnés par les relations :

$$\mathcal{I}'_1 < \mathcal{I}'_2 < \cdots < \mathcal{I}'_m.$$

Soient $\mathcal{I}_X$ et $\mathcal{I}_Y$ les ensembles d'intervalles disjoints définis par :

$$\begin{aligned}\mathcal{I}_X &= \{X_{i,j} \mid 1 \leq i \leq n \text{ et } 1 \leq j \leq 2\},\\ \mathcal{I}_Y &= \{Y_{i,j} \mid 1 \leq i \leq m \text{ et } 1 \leq j \leq 2\}.\end{aligned}$$

Nous ordonnons les intervalles des ensembles $\mathcal{I}_X$ et $\mathcal{I}_Y$ par les relations :

$$\begin{aligned}&X_{n,0} < X_{n,1} < X_{n-1,0} < X_{n-1,1} < \cdots < X_{1,0} < X_{1,1},\\ &Y_{m,0} < Y_{m,1} < Y_{m-1,0} < Y_{m-1,1} < \cdots < Y_{1,0} < Y_{1,1}.\end{aligned}$$

Enfin, soient $\mathcal{I}_A$, $\mathcal{I}_{A'}$, $\mathcal{I}_B$ et $\mathcal{I}_{B'}$ quatre ensembles d'intervalles disjoints définis par :

$$\begin{aligned}\mathcal{I}_A &= \{A_j \mid 1 \leq j \leq nm + 2(m+n) + 1\},\\ \mathcal{I}_{A'} &= \{A'_j \mid 1 \leq j \leq nm + 2(m+n) + 1\},\\ \mathcal{I}_B &= \{B_j \mid 1 \leq j \leq nm + 2(m+n) + 1\},\\ \mathcal{I}_{B'} &= \{B'_j \mid 1 \leq j \leq nm + 2(m+n) + 1\}\end{aligned}$$

et ordonnés par les relations :

$$\begin{aligned}&A_1 < A_2 < \cdots < A_\ell && \text{pour } \ell = mn + 2(m+n) + 1,\\ &A'_1 < A'_2 < \cdots < A'_\ell && \text{pour } \ell = mn + 2(m+n) + 1,\\ &B_1 < B_2 < \cdots < B_\ell && \text{pour } \ell = mn + 2(m+n) + 1,\\ &B'_1 < B'_2 < \cdots < B'_\ell && \text{pour } \ell = mn + 2(m+n) + 1.\end{aligned}$$

Nous ordonnons tous les intervalles définis par les relations :

$$\mathcal{I}_V < \mathcal{I}_A < \mathcal{I}_B < \mathcal{I}_Y < \mathcal{I}_{A'} < \mathcal{I}_X < \mathcal{I}_{B'} < \mathcal{I}_E.$$

Considérons maintenant les familles de 2-intervalles $\mathcal{F}_{VX}$, $\mathcal{F}_{YE}$, $\mathcal{F}_{AA'}$, $\mathcal{F}_{BB'}$ et $\mathcal{F}_{VE}$ définies par :

$$\begin{aligned}\mathcal{F}_{VX} &= \{(I_{i,0},X_{i,0}),(I_{i,m+1},X_{i,1}) \mid 1 \leq i \leq n\},\\ \mathcal{F}_{YE} &= \{(Y_{i,0},J_{i,0}),(Y_{i,1},J_{i,n+1}) \mid 1 \leq i \leq m\},\\ \mathcal{F}_{AA'} &= \{(A_i,A'_i) \mid 1 \leq i \leq mn + 2(m+n) + 1\},\\ \mathcal{F}_{BB'} &= \{(B_i,B'_i) \mid 1 \leq i \leq mn + 2(m+n) + 1\},\\ \mathcal{F}_{VE} &= \{(I_{i,j},J_{j,i}) \mid u_i \in V,\, e_j \in E \text{ et } u_i \notin e_j\}.\end{aligned}$$

Ces familles ont des rôles différents dans notre construction. La famille $\mathcal{F}_{VX}$ (resp. $\mathcal{F}_{YE}$) permet de «*sélectionner*» un ensemble d'intervalles associés à un même sommet (resp. une même arête). Les familles $\mathcal{F}_{AA'}$ et $\mathcal{F}_{BB'}$ - utilisées conjointement

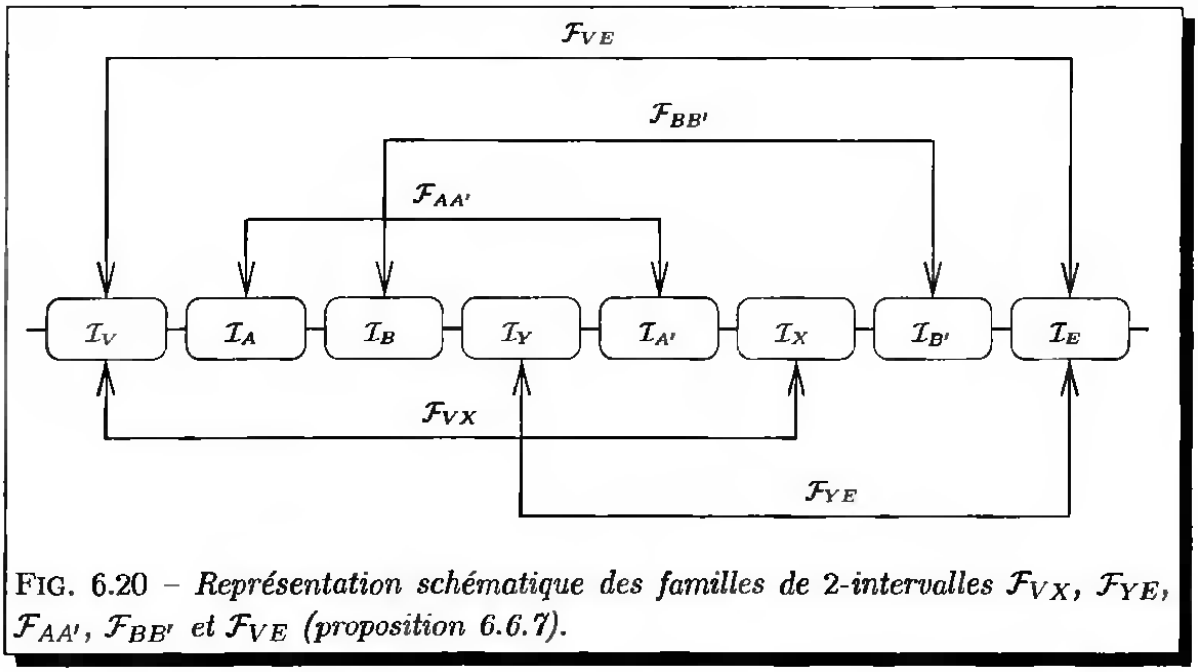


FIG. 6.20 – *Représentation schématique des familles de 2-intervalles $\mathcal{F}_{VX}$, $\mathcal{F}_{YE}$, $\mathcal{F}_{AA'}$, $\mathcal{F}_{BB'}$ et $\mathcal{F}_{VE}$ (proposition 6.6.7).*

- permettent quant à elles d'*isoler* un ensemble d'intervalles. Plus précisément, la famille $\mathcal{F}_{AA}$ permet d'isoler l'ensemble $\mathcal{I}_Y$ et la famille $\mathcal{F}_{BB'}$ permet d'isoler l'ensemble $\mathcal{I}_X$. Enfin, la famille $\mathcal{F}_{VE}$ est le cœur de la réduction puisqu'elle «*code*» une représentation du graphe G.

L'instance correspondante du problème MOTIF 2-INTERVALLES $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES est donnée par une famille $\mathcal{F}$ de 2-intervalles et par un mot p. La famille $\mathcal{F}$ est l'union des familles précédemment définies :

$$\mathcal{F} = \mathcal{F}_{VX} \cup \mathcal{F}_{YE} \cup \mathcal{F}_{AA'} \cup \mathcal{F}_{BB'} \cup \mathcal{F}_{VE}.$$

Remarquons que tous les 2-intervalles de la famill e$\mathcal{F}$ sont $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-comparables. Il suffit de remarquer qu'il n'existe pas deux 2-intervalles D_1,$D_2 \in \mathcal{F}$ tels que $D_1 < D_2$ dans notre construction (c.f. figure 6.20).

Détaillons maintenant la construction du mot p de l'instance. Nous procédons une nouvelle fois en plusieurs étapes. Tout d'abord, soient p_X et p_Y les mots définis par :

$$\begin{aligned} p_X &= x_1\, x'_1\, x_2\, x'_2 \ldots x_{n-K}\, x'_{n-K} \\ p_Y &= y_1\, y'_1\, y_2\, y'_2 \cdots y_\ell\, y'_\ell \qquad\qquad \text{pour } l = \frac{1}{2}K(K-1) \end{aligned}$$

où les x_i, x'_i, y_i et y'_i sont des lettres distinctes. Intuitivement, le mot p_X représente $n-K$ intervalles de l'ensemble $\mathcal{I}_X$, tandis que le mot p_Y représente $\frac{1}{2}K(K-1)$ intervalles de l'ensemble $\mathcal{I}_Y$. De plus, soient p_A, $p_{A'}$, p_B et $p_{B'}$ les mots définis par :

$$\begin{aligned} p_A &= a_1\, a_2 \cdots a_\ell && \text{pour } \ell = mn + 2(m+n) + 1, \\ p_{A'} &= a_1\, a_2 \cdots a_\ell && \text{pour } \ell = mn + 2(m+n) + 1, \\ p_B &= b_1\, b_2 \cdots b_\ell && \text{pour } \ell = mn + 2(m+n) + 1, \\ p_{B'} &= b_1\, b_2 \cdots b_\ell && \text{pour } \ell = mn + 2(m+n) + 1. \end{aligned}$$

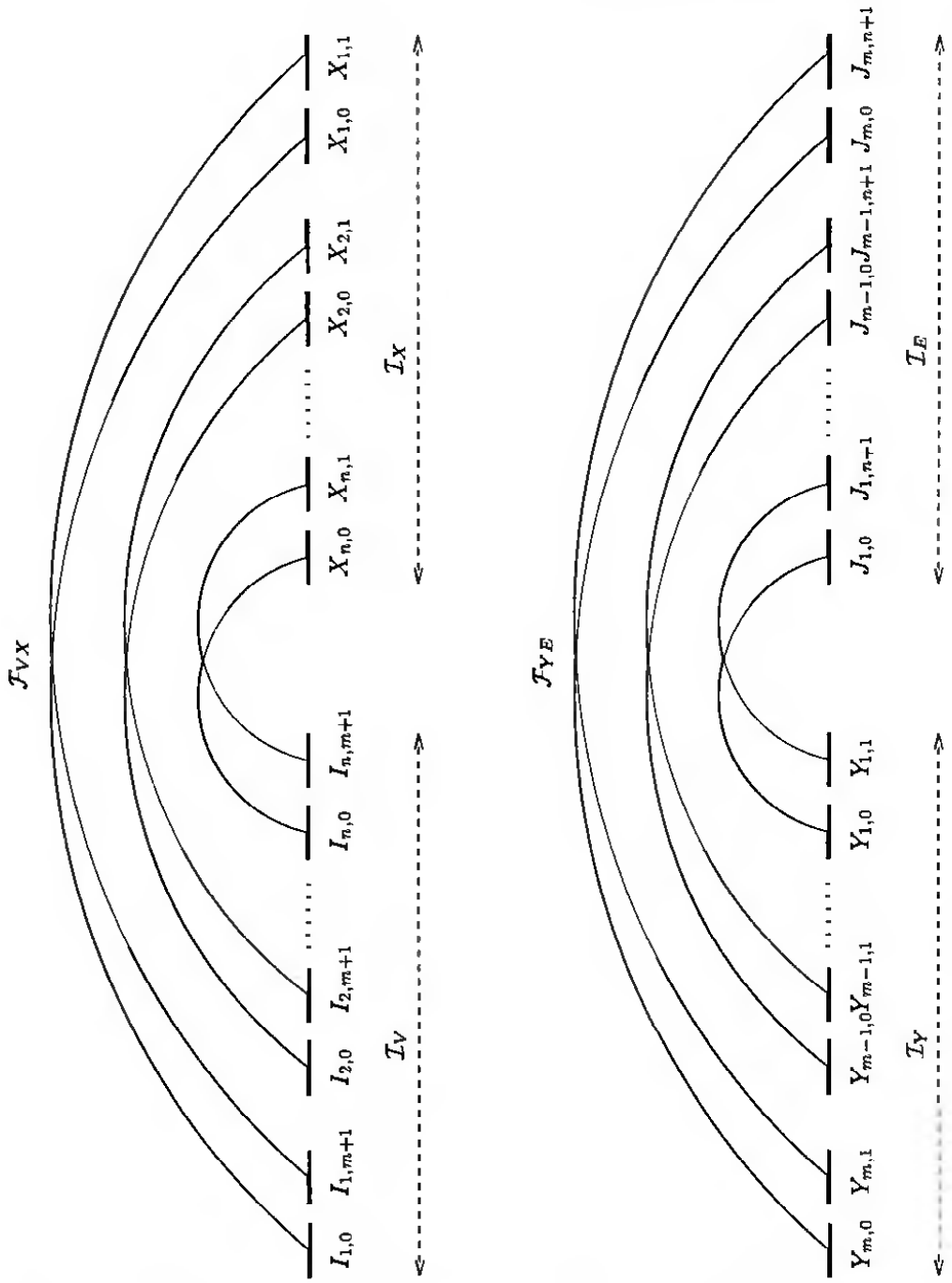


FIG. 6.21 – *Représentation schématique des familles de 2-intervalles $\mathcal{F}_{VX}$ et $\mathcal{F}_{YE}$ (proposition 6.6.7). Par souci de clarté, les intervalles $I_{i,j}$ pour $1 \leq i \leq n$ et $1 \leq j \leq m$, et $J_{i,j}$ pour $1 \leq i \leq m$ et $1 \leq j \leq n$ ne sont pas représentés.*

Encore une fois, les a_i et b_i sont des lettres distinctes. Intuitivement, les mots p_A, $p_{A'}$, p_B et $p_{B'}$ sont associés respectivement aux ensembles $\mathcal{I}_A$, $\mathcal{I}_{A'}$, $\mathcal{I}_B$ et $\mathcal{I}_{B'}$. Nous définissons maintenant $n-K$ mots p_i (qui correspondent dans notre construction à $n-K$ sommets distincts du graphe) et $\frac{1}{2}K(K-1)$ mots p'_i (qui correspondent dans notre construction à $\frac{1}{2}K(K-1)$ arêtes distinctes du graphe) :

$$p_i = x_i\, \alpha_i\, x'_i \qquad \text{pour } 1 \leq i \leq n-K,$$

$$p'_i = y_i\, \beta_i\, y'_i \qquad \text{pour } 1 \leq i \leq \frac{1}{2}K(K-1)$$

où les α_i, $1 \leq i \leq n-K$, sont des mots de longueur $\frac{1}{2}K(K-1)$ et les β_i, $1 \leq i \leq \frac{1}{2}K(K-1)$, sont des mots de longueur $n-K$ (les x_i et y_i sont les lettres des mots p_X et p_Y). Nous imposons l'égalité $\alpha_i[j] = \beta_j[i]$ pour $1 \leq i \leq n-K$ et $1 \leq j \leq \frac{1}{2}K(K-1)$. Enfin, nous notons p_V et p_E les mots définis par :

$$p_V = p_1\, p_2 \cdots p_{n-K},$$

$$p_E = p'_1\, p'_2 \cdots p'_\ell \qquad \text{pour } \ell = \frac{1}{2}K(K-1).$$

Nous pouvons maintenant définir complètement le motif p de l'instance de notre problème MOTIF 2-INTERVALLES $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-COMPARABLES :

$$p = p_V\, p_A\, p_B\, p_Y\, p_{A'}\, p_X\, p_{B'}\, p_E.$$

Nous vérifions que le mot p est *bien formé*; *i.e.* chaque lettre de p apparaît exactement deux fois. De plus, il est compatible avec le modèle de comparaison $\mathcal{R} = \{\sqsubset, \sqsubset\!\!\sqsubset\}$. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une clique $V' \subseteq V$ de taille K dans G. Notons $E' \subseteq E$ l'ensemble des arêtes de la clique. Soit $\mathcal{F}' \subseteq \mathcal{F}$ la sous-famille de 2-intervalles définie de la façon suivante :

1. pour chaque sommet $u_i \in V \setminus V'$ et chaque arête $e_j \in E'$, la sous-famille $\mathcal{F}'$ contient le 2-intervalle $(I_{i,j}, J_{j,i})$;
2. pour chaque sommet $u_i \in V \setminus V'$, la sous-famille $\mathcal{F}'$ contient les 2-intervalles $(I_{i,0}, X_{i,0})$ et $(I_{i,m+1}, X_{i,1})$;
3. pour chaque arête $e_i \in E'$, la sous-famille $\mathcal{F}'$ contient les 2-intervalles $(Y_{i,0}, J_{i,0})$ et $(Y_{i,1}, J_{i,n+1})$;
4. $\mathcal{F}_{AA'} \subset \mathcal{F}'$;
5. $\mathcal{F}_{BB'} \subset \mathcal{F}'$.

Nous vérifions que le mot p est un représentant de la sous-famille $\mathcal{F}'$.

Inversement, supposons qu'il existe une sous-famille $\mathcal{F}' \subseteq \mathcal{F}$ de 2-intervalles $\{\sqsubset, \sqsubset\!\!\sqsubset\}$-comparables de représentant p. Nous remarquons dans un premier temps la présence dans p des sous-mots $p_A\, p_{A'}$ et $p_B\, p_{B'}$ avec $p_A = p_{A'}$, $p_B = p_{B'}$ et $|p_A| = |p_B| = mn + 2(m+n) + 1$. Mais, nous avons :

$$\text{card}(\mathcal{F}_{VE} \cup \mathcal{F}_{VX} \cup \mathcal{F}_{YE}) \leq mn + 2(m+n).$$

Par conséquent, les sous-mots $p_A\, p_{A'}$ et $p_B\, p_{B'}$ sont associés respectivement aux familles de 2-intervalles $\mathcal{F}_{AA'}$ et $\mathcal{F}_{BB'}$. De plus, en rappelant que nous avons :

$$\mathcal{I}_V < \mathcal{I}_A < \mathcal{I}_B < \mathcal{I}_Y < \mathcal{I}_{A'} < \mathcal{I}_X < \mathcal{I}_{B'} < \mathcal{I}_E,$$

nous en tirons immédiatement les conséquence suivantes :

1. le facteur p_V est associé à des intervalles de l'ensemble $\mathcal{I}_V$;
2. le facteur p_Y est associé à des intervalles de l'ensemble $\mathcal{I}_Y$;
3. le facteur p_X est associé à des intervalles de l'ensemble $\mathcal{I}_X$;
4. le facteur p_E est associé à des intervalles de l'ensemble $\mathcal{I}_E$.

Rappelons maintenant que $p_V = p_1\, p_2 \cdots p_{n-K}$ avec $p_i = x_i\ \alpha_i\ x'_i$ pour $1 \leq i \leq n-K$. Remarquons alors que le mot p contient le sous-mot $x_i\ \alpha_i\ x'_i\ x_i\ x'_i$ pour $1 \leq i \leq n-K$ où les deux premières occurrences des lettre x_i, x'_i sont associées à des intervalles de $\mathcal{I}_V$ et les deux dernières occurrences des lettres x_i, x'_i sont associées à des intervalles de $\mathcal{I}_X$ (c.f. remarque précédente). Par conséquent, il existe deux 2-intervalles $D = (I_{i,j}, X_{k,\ell})$ et $D' = (I_{i',j'}, X_{k',\ell'})$ de la sous-famille $\mathcal{F}'$ (et *a fortiori* de la famille $\mathcal{F}$) avec $I_{i,j}, I_{i',j'} \in \mathcal{I}_V$ et $X_{k,\ell}, X_{k',\ell'} \in \mathcal{I}_X$ tels que $I_{i,j} < I_{i',j'} < X_{k,\ell} < X_{k',\ell'}$. Mais, par construction (c.f. figure 6.21), nous avons nécessairement les égalités suivantes :

$$\begin{aligned} I_{i,j} &= I_{i,0}, & X_{k,\ell} &= X_{i,0}, \\ I_{i',j'} &= I_{i,n+1}, & X_{k',\ell'} &= X_{i,1}. \end{aligned}$$

En effet, nous avons $D \sqsupset\!\!\sqsubset D'$ si et seulement si $i \neq i'$. Aussi, le facteur α_i, $1 \leq i \leq n-K$, correspond à $\frac{1}{2}K(K-1)$ intervalles d'un même ensemble $\mathcal{I}_i$. En appliquant le même raisonnement, nous en concluons que le facteur β_i, $1 \leq i \leq \frac{1}{2}K(K-1)$, correspond à $n-K$ intervalles d'un même ensemble $\mathcal{I}'_i$. Enfin, en remarquant que $\alpha_i[j] = \beta_j[i]$ pour $1 \leq i \leq n-K$ et $1 \leq j \leq \frac{1}{2}K(K-1)$, nous en concluons qu'il existe $n-K$ sommets de G que ne sont pas incidents à $\frac{1}{2}K(K-1)$ arêtes. Par conséquent, il existe une clique de taille K dans G. $\boxed{6.6.7}$

REMARQUE 6.6.8 MOTIF 2-INTERVALLES $\{\sqsubset, \sqsupset\sqsubset\}$-COMPARABLES est un problème NP-complet même si la famille $\mathcal{F}$ est à support disjoint. ◇

6.6.6 Modèle de comparaison $\mathcal{R} = \{<, \sqsubset\}$

Le problème MOTIF 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES est sans aucun doute le problème le plus intéressant (tout au moins du point de vue biologique car il correspond à la recherche d'une structure secondaire sans pseudo-nœud dans un ensemble d'hélices). Cependant, nous n'avons pas été en mesure de déterminer la complexité de ce problème.

Les résultats des sous-sections 6.6.4 et 6.6.5 ne sont pas vraiment «*surprenants*» car nous avons montré précédemment que les problèmes SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \sqsupset\sqsubset\}$-COMPARABLES et SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset, \sqsupset\sqsubset\}$-COMPARABLES sont NP-complets. Le cas du problème MOTIF 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES est par contre plus intéressant car le problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES est lui résoluble en temps polynomial.

Nous présentons dans un premier temps un problème d'intervalles qui est de même complexité qu'un cas particulier du problème SOUS-FAMILLE 2-INTERVALLES $\{<$

, $\sqsubset$}-COMPARABLES. Nous appelons APPLICATION INJECTIVE POLYSÉMIQUE SPEC INTERVALLES [13] ce problème. Dans la suite, $<$ désigne l'ordre de précédence entre intervalles et $\subset$ désigne l'ordre d'inclusion stricte entre intervalles.

PROBLÈME 6.6.9 (APPLICATION INJECTIVE POLYSÉMIQUE SPEC INTERVALLES)
DONNÉES : Un ensemble d'intervalles $\mathcal{I}$ comparables deux à deux par $<$ ou *subseteq* et un ensemble d'intervalles $\mathcal{I}'$.
QUESTION : Existe-t-il une application injective ϕ de $\mathcal{I}$ dans $\mathcal{I}'$ telle que, pour tout $I_i, I_j \in \mathcal{I}$ et $R \in \{<, \subset\}$, $I_i \, R \, I_j$ si et seulement si $\phi(I_i) \, R \, \phi(I_j)$?

Sans perte de généralité, nous pouvons supposer dans le problème APPLICATION INJECTIVE POLYSÉMIQUE SPEC INTERVALLES que les intervalles n'ont aucun point extrémité en commun. Nous pouvons donc associer à chaque intervalle dont les extrémités sont x et x' le 2-intervalle constitué des intervalles x et x', *i.e.* nous associons à chaque point extrémité un intervalle. La proposition suivante utilise cette construction.

PROPOSITION 6.6.10 *Si* APPLICATION INJECTIVE POLYSÉMIQUE SPEC INTERVALLES *est un problème* NP*-complet, alors* MOTIF 2-INTERVALLES {$<$, $\sqsubset$}-COMPARABLES *est un problème* NP*-complet.*

PROPOSITION 6.6.11 *Le problème* APPLICATION INJECTIVE POLYSÉMIQUE SPEC INTERVALLES *est résoluble en temps polynomial si $\mathcal{I}'$ est également un ensemble d'intervalles comparables deux à deux par $<$ ou $\subset$.*

PREUVE
Si $\mathcal{I}$ et $\mathcal{I}'$ sont des ensembles d'intervalles comparables deux à deux par $<$ ou *subset*, nous pouvons représenter ces deux ensembles par des arbres ordonnés : Nous associons à la relation entre intervalles «*est un sous-intervalle de*» la relation entre les sommets de l'arbre «*est un descendant de*» et à la relation entre intervalles «*est strictement à gauche de*» l'ordre entre les fils d'un même sommet dans l'arbre [14]. Le problème APPLICATION INJECTIVE POLYSÉMIQUE SPEC INTERVALLES est - dans ce cas - équivalent au problème du *plongement d'un arbre ordonné dans un autre* (c.f. figure 6.22) qui est résoluble en temps polynomial. Nous renvoyons à [Kil92] pour une présentation très complète du problème - ainsi que de nombreuses autres variantes - et de l'algorithme proposé. 6.6.11

Si nous abandonnons le cadre des ensembles d'intervalles, le problème est NP-complet. Nous appelons APPLICATION INJECTIVE POLYSÉMIQUE cette généralisation du problème APPLICATION INJECTIVE POLYSÉMIQUE SPEC INTERVALLES.

13. Le terme «*polysémique*» est utilisé par P.J. Tanenbaum [Tan96].
14. Plus précisément, nous obtenons ainsi deux forêts dont les fils sont ordonnés, mais il est facile d'obtenir des arbres ordonnés en ajoutant dans chaque ensemble un intervalle qui contient tous les autres.

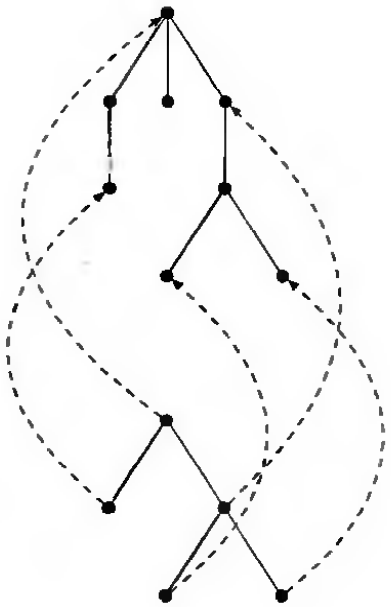


FIG. 6.22 – *Plongement d'un arbre ordonné dans un autre. En pointillés, une application injective ϕ solution du problème.*

PROBLÈME 6.6.12 (APPLICATION INJECTIVE POLYSÉMIQUE)
DONNÉES : Les ensembles partiellement ordonnés $P_1 = (X, \leq_1)$, $P_2 = (X, \leq_2)$, $P_1' = (Y, \leq_1')$ et $P_2' = (Y, \leq_2')$ tels que (1) les graphes de comparaison de P_1 et P_2 sont disjoints, (2) les graphes de comparaison de P_1' et P_2' sont disjoints et (3) les éléments de X sont comparables deux à deux par $\leq_1$ ou $\leq_2$.
QUESTION : Existe-t-il une application injective ϕ de X dans Y telle que, pour tout $x,x' \in X$ et $1 \leq i \leq 2$, $x \leq_i x'$ si et seulement si $\phi(x) \leq_i' \phi(x')$?

PROPOSITION 6.6.13 APPLICATION INJECTIVE POLYSÉMIQUE *est un problème* NP-*complet.*

PREUVE

Le problème APPLICATION INJECTIVE POLYSÉMIQUE appartient à NP. Montrons qu'il est NP-complet. Soit une instance arbitraire du problème CLIQUE donnée par un graphe $G = (V,E)$ d'ordre n et par un entier positif K. Notons $V = \{u_1,u_2,\ldots,u_n\}$ et $E = \{e_1,e_2,\ldots,e_m\}$. Soient X et Y les ensembles disjoints définis par :

$$X = X' \cup X'$$
$$Y = V \cup E$$

où $X' = \{x_i' \mid 1 \leq i \leq n-K\}$ et $X'' = \{x_i'' \mid 1 \leq i \leq \frac{1}{2}K(K-1)\}$ sont deux sous-ensembles disjoints. L'instance correspondante du problème APPLICATION INJECTIVE

POLYSÉMIQUE est donnée par les ensembles partiellement ordonnés $P_1 = (X, \leq_1)$, $P_2 = (X, \leq_2)$, $P_1' = (Y, \leq_1')$ et $P_2' = (Y, \leq_2')$ définis par:

P_1 : pour tout $x_i, x_j \in X$, $x_i \leq_1 x_j$ si et seulement si $x_i \in X'$ et $x_j \in X''$;

P_2 : pour tout $x_i, x_j \in X$, $x_i \leq_2 x_j$ si et seulement si (1) $x_i, x_j \in X'$ et $i < j$ ou (2) $x_i, x_j \in X''$ et $i < j$;

P_1' : pour tout $y_i, y_j \in Y$, $y_i \leq_1' y_j$ si et seulement si $y_i \in V$, $y_j \in E$ et $y_i \notin y_j$;

P_2' : pour tout $y_i, y_j \in Y$, $y_i \leq_2' y_j$ si et seulement si (1) $y_i, y_j \in V$ et $i < j$ ou (2) $y_i, y_j \in E$ et $i < j$.

Remarquons que (1) les graphes de comparaison de P_1 et P_2 sont disjoints, (2) les graphes de comparaison de P_1' et P_2' sont disjoints et (3) les éléments de X sont comparables deux à deux par $\leq_1$ ou $\leq_2$. La construction peut s'effectuer en temps polynomial.

Supposons qu'il existe une clique V' de taille K dans le graphe G. Soient $V \setminus V = \{u_{i_1}, u_{i_2}, \ldots, u_{i_{n-K}}\}$ avec $i_1 < i_2 < \ldots < i_{n-K}$ l'ensemble des sommets de G qui n'appartiennent pas à la clique et $E' = \{e_{i_1}, e_{i_2}, \ldots, e_{i_{\frac{1}{2}K(K-1)}}\}$ avec $i_1 < i_2 < \ldots < i_{\frac{1}{2}K(K-1)}$ l'ensemble des arêtes de la clique. Considérons l'application injective ϕ de X dans Y définie par:

$$\begin{cases} \phi(x_j') = u_{i_j} & \text{pour } 1 \leq j \leq n-K, \\ \phi(x_j'') = e_{i_j} & \text{pour } 1 \leq j \leq \frac{1}{2}K(K-1). \end{cases}$$

Nous vérifions facilement que l'application injective ϕ satisfait la propriété demandée.

Inversement, supposons qu'il existe une application injective ϕ de X dans Y qui satisfasse la propriété demandée. Rappelons que, pour tout $y_i, y_j \in Y$, nous avons $y_i \leq_1' y_j$ si et seulement si $y_i \in V$, $y_j \in E$ et $y_i \notin y_j$. Par conséquent, il existe dans G un ensemble V'' de $n-K$ sommets et un ensemble E' de $\frac{1}{2}K(K-1)$ arêtes tels que $u_i \notin e_j$ pour tout $u_i \in V''$ et $e_j \in E'$. Nous en concluons que le graphe induit par le sous-ensemble de sommets $V \setminus V''$ est une clique de taille K. $\boxed{6.6.13}$

6.6.7 Modèle de comparaison $\mathcal{R} = \{<\}$

Nous avons montré dans la sous-section 6.5.5 que la recherche d'une famille de cardinalité maximale de 2-intervalles $\{<\}$-comparables est un problème résoluble en temps quadratique. Remarquons maintenant qu'un motif u associé à une famille de 2-intervalles $(<)$-comparables est nécessairement de la forme:

$$u = a_1 a_1 a_2 a_2 \ldots a_k a_k.$$

Aussi, il existe une occurrence du motif $a_1 a_1 a_2 a_2 \ldots a_k a_k$ dans $\mathcal{F}$ si et seulement si il existe une sous-famille de $\mathcal{F}$ de cardinalité au moins k de 2-intervalles $\{<\}$-comparables.

6.6.8 Modèle de comparaison $\mathcal{R} = \{\sqsubset\}$

À l'instar de la sous-section précédente, nous avons montré (sous-section 6.5.6) que la recherche d'une famille de cardinalité maximale de 2-intervalles $\{\sqsubset\}$-comparables

est un problème résoluble en temps quadratique. Un motif u associé à une famille de 2-intervalles $\{\sqsubset\}$-comparables est nécessairement de la forme :

$$u = a_1 a_2 \ldots a_k a_k \ldots a_2 a_1.$$

Alors, il existe une occurrence du motif $a_1 a_2 \ldots a_k a_k \ldots a_2 a_1$ dans $\mathcal{F}$ si et seulement si il existe une sous-famille de $\mathcal{F}$ de cardinalité au moins k de 2-intervalles $\{\sqsubset\}$-comparables.

6.6.9 Modèle de comparaison $\mathcal{R} = \{\sqsupset\!\!\sqsubset\}$

Encore une fois, nous avons montré (sous-section 6.5.7) que la recherche d'une sous-famille de cardinalité maximale de 2-intervalles $\{\sqsupset\!\!\sqsubset\}$-comparables est un problème résoluble en temps quadratique. Un motif u associé à une famille de 2-intervalles $\{\sqsupset\!\!\sqsubset\}$-comparables est nécessairement de la forme :

$$a_1 a_2 \ldots a_k a_1 a_2 \ldots a_k.$$

En conséquence, il existe une occurrence du motif $a_1 a_2 \ldots a_k a_1 a_2 \ldots a_k$ dans $\mathcal{F}$ si et seulement si il existe une sous-famille de $\mathcal{F}$ de cardinalité au moins k de 2-intervalles $\{\sqsupset\!\!\sqsubset\}$-comparables.

6.7 Conclusion et problèmes ouverts

Dans un premier temps, nous avons montré que SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsubset, \sqsupset\!\!\sqsubset\}$-COMPARABLES est un problème NP-complet[15]. Nous en avons déduit que STABLE est un problème NP-complet même si le graphe G est un graphe de 2-intervalles (le graphe d'intersection d'une famille de 2-intervalles). Le même problème pour le modèle de comparaison $\mathcal{R} = \{\sqsubset, \sqsupset\!\!\sqsubset\}$ est également NP-complet[16]. Inversement, le problème est résoluble en temps polynomial pour les modèles de comparaison $\mathcal{R} = \{<\}$[17], $\mathcal{R} = \{\sqsubset\}$[18] et $\mathcal{R} = \{\sqsupset\!\!\sqsubset\}$[19]. Nous ignorons cependant la complexité du seul cas restant, c'est-à-dire le problème SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsupset\!\!\sqsubset\}$-COMPARABLES.

QUESTION 10 *Quelle est la complexité du problème* SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsupset\!\!\sqsubset\}$-COMPARABLES *?*

Remarquons que si SOUS-FAMILLE 2-INTERVALLES $\{<, \sqsupset\!\!\sqsubset\}$-COMPARABLES est un problème NP-complet, alors SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES est un problème NP-complet si et seulement si $|\mathcal{R}| > 1$ et $\sqsupset\!\!\sqsubset \in \mathcal{R}$. La question suivante reste également sans réponse.

QUESTION 11 *Quelle est la complexité du problème* SOUS-FAMILLE 2-INTERVALLES $\{\sqsubset, \sqsupset\!\!\sqsubset\}$-COMPARABLES *si $\mathcal{F}$ est une famille de 2-intervalles à support disjoint?*

15. proposition 6.5.2 page 176.
16. proposition 6.5.10 page 183.
17. sous-section 6.5.5.
18. sous-section 6.5.6.
19. corollaire 6.5.22 page 189.

Nous avons ensuite introduit le problème de la recherche d'un motif dans une famille de 2-intervalles relativement à un modèle de comparaison $\mathcal{R}$ donné (problème MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES). Nous avons justifié l'intérêt de ce problème en relation avec la recherche des strustures secondaires d'ARN particulières. Nous n'avons obtenu sur ce problème que des résultats partiels. Néanmoins, nous avons montré que MOTIF 2-INTERVALLES $\{<, \sqsubset, \boxed{\sqsubset}\}$-COMPARABLES et MOTIF 2-INTERVALLES $\{\sqsubset, \boxed{\sqsubset}\}$-COMPARABLES sont des problèmes NP-complets[20]. Cependant, nous n'avons pas été en mesure de donner la complexité du problème le plus intéressant (tout au moins du point de vue biologique), à savoir le problème MOTIF 2-INTERVALLES $\{<, \sqsubset\}$-COMPARABLES. Le problème reste donc posé.

QUESTION 12 *Quelle est la complexité du problème* MOTIF 2-INTERVALLES $\{<, \sqsubset\}$--COMPARABLES *?*

Le cas du modèle d'intersection $\mathcal{R} = \{<, \boxed{\sqsubset}\}$ reste également sans répose.

QUESTION 13 *Quelle est la complexité du problème* MOTIF 2-INTERVALLES $\{<, \boxed{\sqsubset}\}$--COMPARABLES *?*

20. proposition 6.6.5 page 195 et proposition 6.6.7 page 198.

Chapitre 7

Reconnaissance des introns auto-catalytiques de groupe I

Nous présentons dans ce dernier chapitre un logiciel que nous avons développé pour la recherche des introns auto-catalytiques de groupe I *dans les séquences génomiques. Ces introns se distinguent en particulier par le fait qu'ils sont capables de s'*exciser *sans l'aide d'aucune enzyme. La partie centrale des introns auto-catalytiques de groupe I est composée de six hélices notées* ($P3$,$P3'$), ($P4$,$P4'$), ($P5$,$P5'$), ($P6$,$P6'$) ($P7$,$P7'$) *et* ($P8$,$P8'$). *La recherche de ces introns est une tâche difficile en raison d'un grande variabilité sous forme de bases non appariées à l'intérieur des hélices, de variations de longueur, d'erreurs et d'éventuelles insertions. De plus, les longueurs des boucles terminales* $L5$, $L6$ *et* $L8$, *et les distances entre le cœur et les points d'épissage sont imprévisibles.*

Nous décrivons un algorithme dédié à la recherche des introns auto-catalytiques de groupe I et discutons de sa mise en œuvre dans notre logiciel `orange`. *Le logiciel* `orange` *est librement téléchargeable et est également accessible sur Internet.*

7.1 (Très) Brefs rappels de biologie moléculaire

Nous commençons ce chapitre par quelques brefs rappels de biologie moléculaire.

L'ADN est formé de deux hélices enlacées, le long desquelles se succèdent les bases (*adénosine* A, *cytosine* C, *guanine* G et *thymine* T) liées aux groupements ribose que joignent des liaisons phosphodiester faisant intervenir un atome de phosphore. Le terme *nucléotide* désigne l'ensemble base-ribose-phosphate et est souvent utilisé pour indiquer la longueur d'un fragment d'ADN de manière interchangeable avec le mot *base*. Les deux chaînes de la double hélice sont rendues solidaires par des «*liaisons hydrogènes*» (liaisons chimiques assez faibles) contractées entre les bases. Une adénosine ne peut s'associer qu'avec une thymine, une guanime seulement avec une cytosine. L'information génétique est constituée par la succession des bases le long d'une chaîne. En raison des règles d'appariements (A avec T et G avec C), la séquence d'une chaîne se déduit directement de celle de l'autre (nous disons qu'elle lui est *complémentaire*).

ADN

nucléotide

Un *gène* donné détermine la synthèse d'une protéine particulière[1] parce que l'enchaînement de ses éléments dicte l'enchaînement des éléments de cette dernière. Il existe existe en effet une règle de correspondance appelée *code génétique* qui associe à chaque groupe de trois nucléotides successifs (*codon*) dans le gène un acide aminé dans la protéine. Par exemple, le codon ATG spécifie un acide aminé appelé *méthionine*. Le code génétique est un code «*dégénéré*» : les vingt acides aminés étant codés $4^3 = 64$ codons différents, un même acide aminé a généralement plusieurs codons (figure 7.1) : il existe donc des codons *synonymes*. La mise en œuvre du gène, en vue de la synthèse d'une protéine, passe par différentes phases. Tout d'abord, la *transcription* au cours de laquelle est fabriquée dans le noyau une copie de travail du gène reprenant l'information sous une forme légèrement différente : une molécule d'ARN messager (abrégé ARNm), chimiquement très similaire à l'ADN, mais ne comportant qu'une seule chaîne. L'ARNm va alors passer dans le cytoplasme et s'associer aux ribosomes qui assurent la *traduction* : la synthèse de la protéine spécifiée par la séquence. Plus précisément, les deux sous-unités du ribosome s'accrochent à l'extrémité de l'ARNm, puis progressent le long de la chaîne. Chaque codon, une fois mis en position dans le ribosome, est reconnu par un ARN de transfert. Ces petits ARN (environs 80 nucléotides) possèdent une structure tridimensionnelle caractéristique. À l'une de leurs extrémités se trouve une boucle qui porte une suite de trois nucléotides, l'*anti-codon*, qui est le complémentaire de l'un de ses codons. À l'autre extrémité de l'ARN de transfert se trouve fixé l'acide aminé correspondant au codon que reconnaît l'anticodon. Le ribosome effectue la liaison entre les acides aminés successivement apportés, de sorte que la chaîne de la protéine s'allonge progressivement. Finalement, le ribosome arrive au codon «STOP» et se dissocie en relâchant l'ARNm, tandis que la protéine se replie sur elle-même pour prendre la forme qui la rend active.

code génétique
codon
transcription
traduction
anti-codon

Cependant, les choses ne sont pas toujours aussi simples. En 1977, un phénomène surprenant fut découvert : certains gènes possèdent des portions excédentaires de nucléotides qui ne figurent pas dans leur ARNm. Autrement dit, ces gènes apparaissent morcelés en parties codantes[2] – les *exons* –, séparés par des portions non codantes – les *introns*. Ce phénomène est d'autant plus surprenant que cette fragmentation peut être considérable. Par exemple, dans les gènes codant pour chacune des chaînes de l'*hémoglobine*[3], il y a trois exons séparés par deux introns.

exon
intron

Le plus souvent, les gènes des organismes supérieurs sont fragmentés. De plus, cette fragmentation peut prendre presque tous les degrés possibles : d'un unique intron – cas du gène codant pour l'*actine* – à plusieurs dizaines d'introns – cas du gène codant pour le collagène. Encore plus surprenant, les introns sont bien souvent plus longs que les exons : dans un gène «*moyen*», la somme de tous les exons représentent environ 1 000 nucléotides, tandis que la somme de tous les introns est de l'ordre de 5 000 à 10 000 nucléotides (et parfois beaucoup plus) [DS85].

Il est clair que, pour diriger la synthèse d'une protéine, un «*gène morcelé*» requiert une étape supplémentaire par rapport au processus que nous avons brièvement présenté précédemment. Le gène morcelé est tout d'abord recopié (*transcrit*) intégra-

1. ou plus exactement «*code*» pour cette protéine.
2. Une partie du gène est dite *codante* si elle est recopiée dans l'ARN messager.
3. Le pigment rouge du sang.

Phe	*UUU*	Ser	*UCU*	Tyr	*UAU*	Cys	*UGA*
	UUC		*UCC*		*UAC*		*UGC*
Leu	*UUA*		*UCA*	stop	*UAA*	stop	*UGA*
	UUG		*UCG*		*UAG*	Trp	*UGG*
Leu	*CUU*	Pro	*CCU*	His	*CAU*	Arg	*CGU*
	CUC		*CCC*		*CAC*		*CGC*
	CUA		*CCA*	Gln	*CAA*		*CGA*
	CUG		*CCG*		*CAG*		*CGG*
Ile	*AUU*	Thr	*ACU*	Asn	*AAU*	Ser	*AGU*
	AUC		*ACC*		*AAC*		*AGC*
	AUA		*ACA*	Lys	*AAA*	Arg	*AGA*
Met	*AUG*		*ACG*		*AAG*		*AGG*
Val	*GUU*	Ala	*GCU*	Asp	*GAU*	Gly	*GGU*
	GUC		*GCC*		*GAC*		*GGC*
	GUA		*GCA*	Glu	*GAA*		*GGA*
	GUG		*GCG*		*GAG*		*GGG*

FIG. 7.1 – *Ce tableau montre comment chacun des vingt acides aminés possibles est représenté par un ou plusieurs groupes de trois bases, les codons, dans l'ARNm. Celui-ci est une copie fidèle de la séquence codante du gène (à cela près que les bases T,* thymine, *sont remplacées par des U,* uracile. *Trois codons, UAA, UAG et UGA codent « fin de message ». Le codon AUG signifie à la fois « méthionine » et « début de message ». Toutes les protéines, lors de leur synthèse, débutent par une méthionine.*

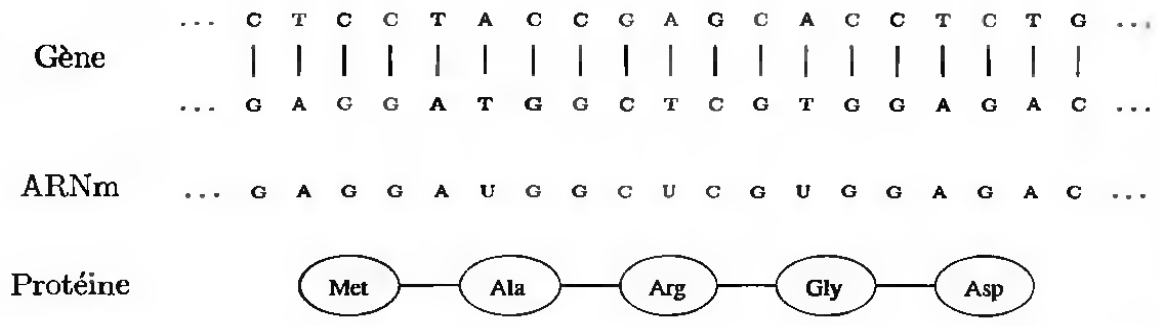


FIG. 7.2 – *Une petite partie d'un gène, l'ARN messager correspondant et le début de la protéine dont il dirige la synthèse.*

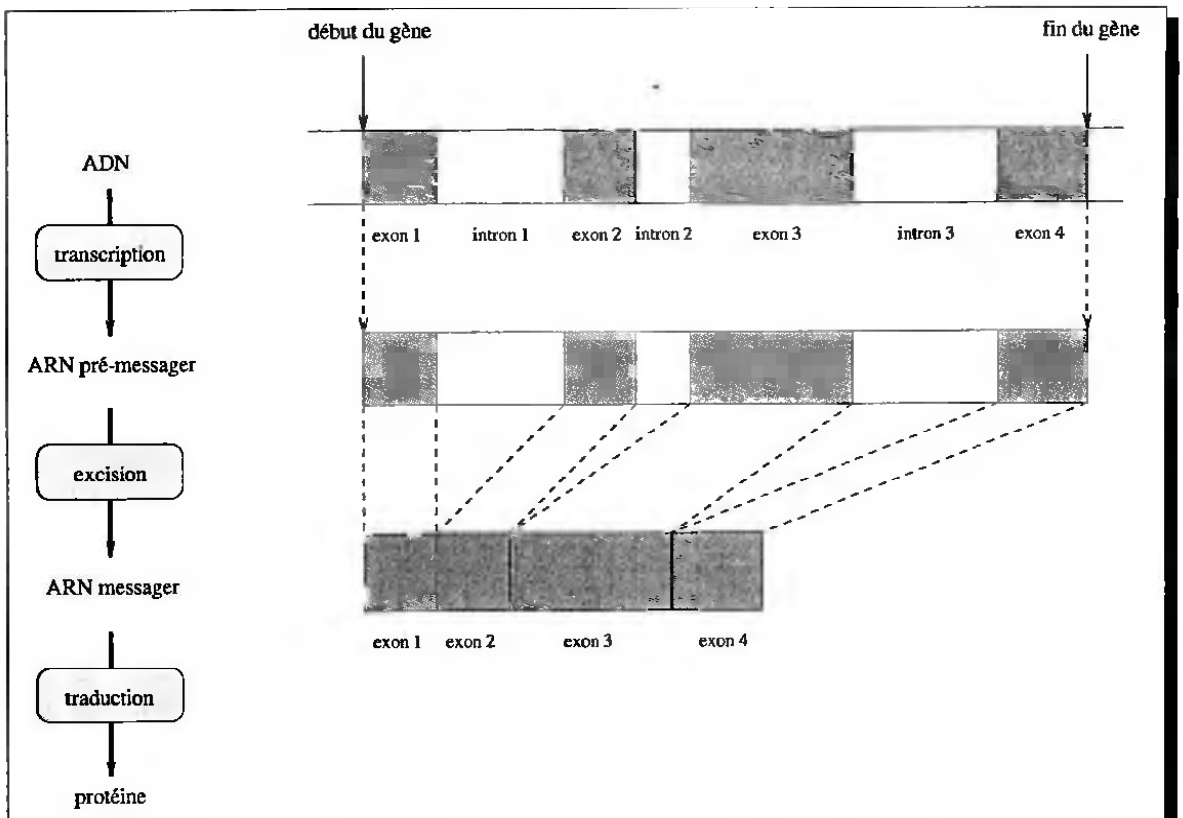


FIG. 7.3 – *Pour diriger la synthèse d'une protéine, un gène « morcelé » requiert une étape supplémentaire par rapport aux processus connus chez les bactéries. Le gène est d'abord intégralement recopié (*transcrit*) en une molécule d'ARN pré-messager. Ensuite, intervient un processus d'excision des introns et de raccordement des exons. La molécule obtenue (ARNm) ne contient qu'une suite* non interrompue *de nucléotides utilisés pour le codage de la protéine correspondant au gène.*

lement[4] en une molécule d'ARN pré-messager. Celle-ci est ensuite débarrassée de ses introns et les exons réassociés bout à bout de façon à donner un ARN messager comprenant uniquement les enchaînements de nucléotides capables de coder pour la protéine finale (figure 7.3).

7.2 Un intron qui s'excise tout seul

Dans le cas des ARN ribosomiques, une découverte surprenante a été faite en 1982. T.R. Cesh a observé qu'un intron d'ARN ribosomique (chez un protozaire) pouvait s'exciser *seul*, c'est-à-dire sans le secours d'aucune enzyme. En d'autres termes, les introns de certains ARN «*s'auto-découpent*» [Cec87]. Il s'agit là d'une découverte importante car elle montre qu'un acide nucléique peut effectuer une *catalyse*[5], alors qu'il était communément admis que cette propriété était l'apanage des protéines. Toutefois, cette réaction observée *in vitro* est beaucoup plus lente que ce qui est observé *in vivo* dans la cellule, ce qui pourrait impliquer la participation dans

4. C'est à dire avec tous ses introns et tous ses exons.
5. Une auto-catalyse en l'occurrence.

la cellule de protéines qui accélèrent la réaction [DS85].

Les introns auto-catalytiques connus sont si variables que seules sept bases (cinq dans le cœur de l'intron et deux aux extrémités) sont strictement conservées. La structure secondaire des introns auto-catalytiques est un peu mieux conservée. En effet, même si la cristallisation qui permet d'étudier la structure des macromolécules n'a pu être réalisée jusqu'à présent pour les introns auto-catalytiques, des hypothèses confirmées dans d'autres contextes expérimentaux ont permis de modéliser la structure bi- et tri-dimensionnelle des introns auto-catalytiques [MW90]. La partie centrale des introns auto-catalytiques est composée de six hélices notées ($P3$,$P3'$), ($P4$,$P4'$), ($P5$,$P5'$), ($P6$,$P6'$), ($P7$,$P7'$) et ($P8$,$P8'$) (un exemple d'intron auto-catalytique de groupe I est donné en figure 7.4). La structure complète d'un intron auto-catalytique est composée de dix hélices (de ($P1$,$P1'$) à ($P10$,$P10'$)). Cependant, il demeure toutefois une grande variabilité sous forme de bases non appariées à l'intérieur des hélices, de variations de longueur, d'erreurs et d'éventuelles insertion. Bien sur, les longueurs des boucles terminales et les distances entre le cœur et les points d'épissage sont imprévisibles.

La taille des hélices qui composent la partie centrale d'un intron auto-catalytique est donnée en figure 7.5. Les distances qui les séparent sont données en figure 7.6. La longueur moyenne de la partie centrale est de l'ordre de quelques centaines de nucléotides.

7.3 Le programme *citron*

7.3.1 Présentation générale

Le programme *citron* [LDM94, Lis95] est le premier logiciel capable de rechercher les introns auto-catalytiques de groupe I dans les séquence génomiques. Il est composé d'un programme écrit en langage C et de cinq fichiers de configuration : *citron*

- le fichier P7FREQ contient la matrice consensus du signal $P7$,
- le fichier J8_7FREQ contient la matrice consensus du signal $J8/7$,
- le fichier P7CFREQ contient la matrice consensus du signal $P7'$,
- le fichier plancher contient treize paramètres seuils,
- le fichier penabonu contient cinq paramètres permettant d'affiner les scores.

Une exécution du programme *citron* produit trois fichiers résultats :

- le fichier sequence.DRW contient la structure de tous les introns localisés dans la séquence.
- le fichier sequence.POS contient la liste de toutes les hélices (Pi,Pi'), $3 \leq i \leq 8$, dont le score est supérieur au seuil correspondant.
- le fichier sequence.TRK contient tous les ensembles $\{(Pi,Pi') \mid 4 \leq i \leq 7\}$ qui ont obtenu un score important mais qui ne figurent dans aucune solution finale.

Nous décrivons maintenant les grandes lignes de l'algorithme utilisé par le programme *citron* pour localiser des introns auto-catalytiques de groupe I :

1. Recherche de tous les signaux $J67 - P7$: tous les segments de dix nucléotides sont systématiquement évalués et retenus si leur score est supérieur à un seuil paramétrable.

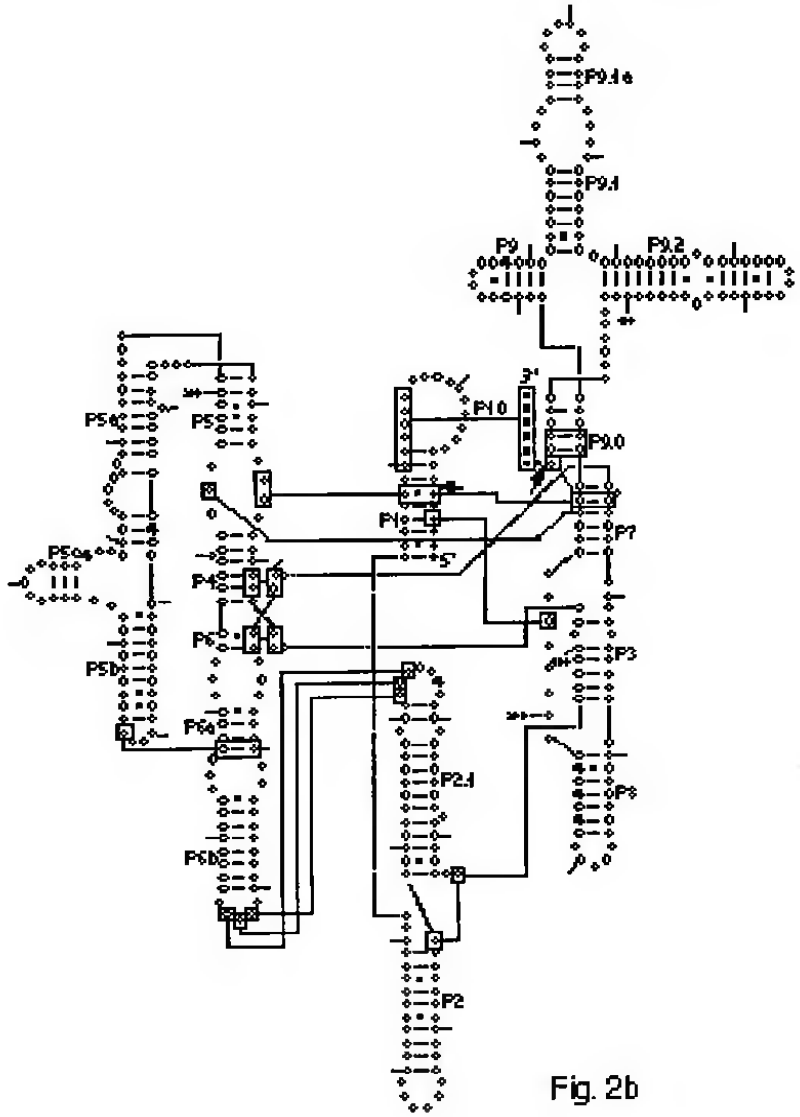


FIG. 7.4 – *Exemple d'une structure secondaire d'un intron-autocatalytique de groupe I.*

Hélices	Longueur	régulier/variable
($P3$,$P3'$)	de 5 à 9	variable
($P4$,$P4'$)	6	régulier
($P5$,$P5'$)	de 2 à 8	variable
($P6$,$P6'$)	de 2 à 6	variable
($P7$,$P7'$)	6	régulier
($P8$,$P8'$)	de 3 à 12	variable

FIG. 7.5 – *La longueur d'une hélice correspond au nombre de paires de nucléotides qui la compose. Les hélices* ($P4$,$P4'$) *et* ($P7$,$P7'$) *ont des longueurs presque toujours constantes. De plus, dans les hélices* ($P4$,$P4'$) *et* ($P7$,$P7'$), *certains nucléotides apparaissent régulièrement ou n'apparaissent presque jamais à des positions particulières identifiées.*

inter-segment	longueur	régulier/variable
$P3 - P4$	de 2 à 120	variable
$P4 - P5$	de 2 à 5	régulier
$P5 - P5'$	de 3 à 1400	variable
$P5' - P4'$	de 2 à 5	régulier
$P4' - P6$	0	
$P6 - P6'$	de 4 à 900	variable
$P6' - P7$	3	régulier
$P7 - P3'$	de 2 à 140	variable
$P3' - P8$	de 0 à 2	variable
$P8 - P8'$	de 3 à 2000	variable
$P8' - P7'$	de 6 à 7	régulier

FIG. 7.6 – *Le nombre de bases séparant deux hélices consécutives correspond à la distance entre deux segments* Pi *et* Pi'. *Trois des cinq bases universelles se trouvent entre* $P4$ *et* $P5$, $P5'$ *et* $P4'$, *et* $P6'$ *et* $P7$; *les deux autres sont à l'intérieur même des segments* $P7$ *et* $P7'$.

2. Recherche de tous les signaux $J87' - P7'$: tous les segments de douze nucléotides sont systématiquement évalués et retenus si leur score est supérieur à un seuil paramétrable.
3. Construction des hélices potentielles ($P7$,$P7'$) à partir des signaux calculés aux étapes 1 et 2.
4. Pour chaque hélice potentielle ($P7$,$P7'$) :
   (a) Recherche des hélices potentielles ($P4$,$P4'$).
   (b) Recherche des combinaisons $\{P4,P5,P5',P4',P6,P6',P7,P7'\}$.
5. Pour chaque combinaison $\{P4,P5,P5',P4',P6,P6',P7,P7'\}$, recherche des hélices ($P3$,$P3'$) et ($P8$,$P8'$) potentielles.

7.3.2 Détails de mise en œuvre

La recherche exhaustive des signaux $J67 - P7$ et $J87' - P7'$ constitue les deux premières étapes du programme ***citron***. Le programme utilise pour cela une matrice consensus, c'est-à-dire une matrice qui contient les fréquences d'apparition de chaque base à chaque position du signal [Sta84]. L'utilisation de deux scores seuils permet de sélectionner les segments $P7$ et $P7'$ candidats.

À la suite de ces recherches, ***citron*** construit toutes les hélices ($P7$,$P7'$) potentielles. Encore une fois, une hélices est retenue si elle satisfait un certain nombre de contraintes (principalement sur le type de ses appariements) et si son score est supérieur à un seuil paramétrable.

La recherche des hélices ($P4$,$P4'$) est - de très loin - l'étape la plus coûteuse en temps du programme ***citron***. En effet, les hélices ($P4$,$P4'$) peuvent contenir un *renflement* et sont plus ou moins proches d'une hélice ($P7$,$P7'$) associée. Aussi, ***citron*** utilise-t-il une recherche exhaustive sur la portion de la séquence qui précède un segment $P7$ donné. La sélection des hélices ($P4$,$P4'$) potentielles dépend de divers critères :

1. la qualité désirée des appariements;
2. la position de l'hélice relativement à une hélice ($P7$,$P7'$) donnée;
3. la présence éventuelle d'un renflement.

Une fois une hélice ($P4$,$P4'$) sélectionnée, ***citron*** tente de localiser une hélice ($P5$,$P5'$) associée. Ces hélices peuvent être très courtes et sont de fait relativement peu contraintes. Les hélices ($P4$,$P4'$) potentielles pour lesquelles aucune hélice ($P5$,$P5'$) n'a pu être localisée sont supprimées de l'ensemble des candidats.

L'étape suivante consiste en la recherche des hélices ($P6$,$P6'$) associées. La recherche est cette fois-ci parfaitement localisée puisque ces hélices sont positionnées entre deux hélices précédemment calculées, à savoir les hélices ($P4$,$P4'$) et ($P7$,$P7'$). Une hélice ($P6$,$P6'$) est sélectionnée suivant la qualité de ses appariements et la longueur de sa boucle.

Le programme ***citron*** recherche ensuite les hélices ($P8$,$P8'$) dans la séquence. Ces hélices sont positionnées entre les segments $P7$ et $P7'$. Une insertion importante peut séparer le segment $P7$ du segment $P8$. Enfin, ***citron*** tente de compléter les solutions en localisant les hélices ($P3$,$P3'$). Le segment $P3'$ est positionné entre les segments $P7$ et $P8$, tandis que le segment $P3$ précède le segment $P4$ (une grande insertion sépare éventuellement $P3$ de $P4$).

L'étape finale consiste essentiellement à calculer le score total de la solution et à retenir celle-ci si le score est supérieur à un seuil paramétrable (d'autres règles de sélection concernent par exemple la longueur de la solution, la présence d'une sous-hélice ($P6a$,$P6a'$), *etc* ...

7.4 Le programme *orange*

7.4.1 Introduction

Nous présentons dans cette section un nouvel algorithme pour la recherche des introns auto-catalytiques de groupe I dans les séquences génomiques. Ce travail nous a conduit à développer un logiciel entièrement écrit en langage C : *orange*[6]. *orange*

Notre but était de satisfaire - autant que possible - les contraintes suivantes :

- éliminer ou diminuer les limitations du programme *citron*;
- améliorer le temps de calcul (par rapport à *citron*);
- permettre la recherche dans de grandes séquences génétiques.

De plus, nous tenions dans la mise en œuvre du programme à respecter les contraintes suivantes :

- développement modulaire;
- proposer un programme facilement paramétrable *via* un fichier de configuration;
- disponibilité sur Internet.

Le logiciel *orange* est de fait un logiciel assez complet et facilement extensible. Il comporte environ 17000 lignes de code (il est abondamment commenté) dans sa dernière version.

7.4.2 Présentation générale

La description précise d'un logiciel peut rapidement devenir quelque chose d'abscons et rébarbatif pour le lecteur. En effet, le logiciel *orange* est truffé de cas exceptionnels, d'exceptions, d'astuces de programmation, ... Aussi, avons-nous pris le parti de décrire essentiellement l'algorithme utilisé; nous ne consacrons qu'une courte partie aux structures de données du programme.

Nous commençons par une description générale de l'algorithme utilisé par *orange* (ces différentes étapes seront ensuite détaillées) :

1. Recherche de tous les signaux $J67 - P7$ dans la séquence.
2. Recherche de tous les signaux $J87' - P7'$ dans la séquence.
3. Calcul de l'ensemble des hélices ($P7$,$P7'$) potentielles à partir des signaux calculés aux étapes 1 et 2.
4. Pour chaque hélice ($P7$,$P7'$) potentielle calculée en 3, recherche des hélices potentielles ($P8$,$P8'$) associées. Les hélices ($P7$,$P7'$) pour lesquelles aucune hélice ($P8$,$P8'$) n'est retenue sont éliminées de l'ensemble des candidats.
5. Pour chaque segment $P7$ appartenant à au moins une hélice calculée en 3, recherche des hélices potentielles ($P6$,$P6'$) associées.

6. Ce logiciel est librement téléchargeable et également accessible sur Internet (c.f. section 7.6).

6. Pour chaque hélice ($P6$,$P6'$) calculée en 5, recherche des hélices potentielles ($P4$,$P4'$) et ($P5$,$P5'$) associées.
7. Pour chaque combinaison ($P4$,$P4'$), ($P5$,$P5'$), ($P6$,$P6'$) ($P7$,$P7'$) et ($P8$,$P8'$) calculée en 6, recherche des hélices potentielles ($P3$,$P3'$) associées.
8. Validation de l'ensemble des introns auto-catalytiques potentiels calculés en 7.
9. Éventuellement, calcul d'une décomposition en solutions non chevauchantes.

7.4.3 Recherche des hélices ($P7$,$P7'$)

Notre algorithme de recherche est complètement déterminé à partir de l'hélice la plus fortement contrainte, à savoir l'hélice ($P7$,$P7'$). Aussi, à l'instar du programme ***citron***, utilisons-nous également les signaux $J67 - P7$ et $J87' - P7'$ comme «*points d'ancrage*» dans la séquence. Cependant, la méthode utilisée ici diffère radicalement de ***citron***.

À partir de la matrice consensus d'un signal, nous pouvons calculer un certain nombre de *bases invariantes*, c'est-à-dire les bases ayant les fréquences les plus élevées à des positions données et un seuil d'acceptation pour la séquence toute entière. C'est cette approche qui a, par exemple, été retenue par l'algorithme ***tRNAscan*** [EM96]. Nous pouvons ensuite considérer le signal comme une classe de mots sur l'alphabet des ARN $\{A,C,G,U\}$. En d'autres termes, chaque position est définie par un sous-ensemble non vide de $\{A,C,G,U\}$. La recherche d'un signal dans une séquence se ramène donc à la recherche des occurrences de la classe de mots associée (à k erreurs près).

Nous avons choisi d'utiliser pour la recherche des segments $J67 - P7$ et $J87 - P7'$ l'algorithme ***shift-add*** [BYG92]. L'idée principale de l'algorithme ***shift-add*** consiste à représenter l'état courant de la recherche par un vecteur à m composante, où m est la longueur du motif de recherche. Pour chaque position courante dans le texte, nous considérons le vecteur état E_j qui contient, pour chaque préfixe du motif de recherche P, le nombre d'erreurs entre ce préfixe et le facteur correspondant du texte T finissant à la position j. En d'autres termes, pour $1 \leq i \leq m$, $E_j[i]$ est le nombre d'erreurs entre $P[1 : i]$ et $T[j - i + 1 : j]$. Une occurrence de P se termine à la position j dans T avec au plus k erreurs si $E_j[m] < k + 1$.

Le calcul du vecteur E_{j+1} ne dépend que de E_j et du caractère $T[j + 1]$. Il suffit en effet de compléter le nombre d'erreurs pour chaque préfixe de P. Aussi, pouvons-nous calculer en prétraitement, pour toute lettre a de l'alphabet, un vecteur, noté S_a, à m colonnes :

$$\text{pour tout } i,\ 1 \leq i \leq m, \qquad S_a[i] = \begin{cases} 0 & \text{si } a = p_i \\ 1 & \text{sinon} \end{cases}$$

Le calcul du vecteur E_{j+1} est donné par :

$$E_{j+1}[i] = E_j[i - 1] + S_{T[j+1]}[i].$$

Le principal intérêt de l'algorithme ***shift-add*** est de pouvoir représenter les vecteurs par des nombres en base 2^b, où b est le nombre de bits nécessaires pour représenter chaque composante d'un vecteur état (la valeur du paramètre b dépend

bien sûr du nombre d'erreurs k). Pour toute position j dans le texte T et toute lettre a de l'alphabet, nous pouvons représenter les vecteurs E_j et S_a par :

$$E_j = \sum_{1 \leq i \leq m} E_j[i] 2^{(i-1)b}$$
$$S_a = \sum_{1 \leq i \leq m} S_a[i] 2^{(i-1)b}$$

Lorsque m n'est pas trop grand, nous pouvons coder chacun de ces nombres sur un unique mot machine. Le calcul de E_j se réécrit alors en :

$$E_{j+1} = (E << b) + S_{T[j+1]}$$

où $<<$ est l'opérateur de décalage à gauche d'un mot machine. Il existe une occurrence de P qui se termine à la position j dans T avec au plus k erreurs si :

$$E_{j+1} < (k+1)2^{(m-1)b}.$$

Le calcul de la table S s'effectue en temps $O(\lceil \frac{mb}{\omega} \rceil (m+c))$ et l'espace utilisé est $O(\lceil \frac{mb}{\omega} \rceil c)$. En pire cas comme en moyenne, l'étape de recherche est en temps $O(\lceil \frac{mb}{\omega} \rceil n)$, où $O(\lceil \frac{mb}{\omega} \rceil$ est le temps nécessaire pour effectuer un nombre constant d'opérations sur un état de longueur mb représenté sur des mots machine de longueur ω. Lorsque qu'un unique mot machine est suffisant pour la représentation de l'état de la recherche ($mb \leq \omega$), la complexité totale de l'algorithme ***shift-add*** est en temps $O(n+m+c)$ et en espace $O(c)$. Si $mb > \omega$, plusieurs mots machines sont nécessaires pour la représentation de l'état de recherche. Cependant, R. Baeza-Yates et G.H. Gonnet montrent que si le nombre de mots machine nécessaires n'est pas trop grand, alors l'algorithme ***shift-add*** reste efficace en pratique.

L'algorithme ***shift-add*** est particulièrement bien adapté à la recherche d'une classe de mots. En effet, il demeure dans ce cas tout aussi efficace que pour le cas de la recherche d'un mot simple. Il suffit simplement de modifier la définition de la table S :

$$\text{pour tout } i,\ 1 \leq i \leq m, \qquad S_a[i] = \begin{cases} 0 & \text{si } a \in p_i \\ 1 & \text{sinon} \end{cases}$$

La table S s'effectue en temps $O(\lceil \frac{mb}{\omega} \rceil (m'+c))$, où m' est la taille de la description de la classe de mots. L'étape de recherche est identique au cas de la recherche d'un unique mot. Nous voyons que dans le cas des alphabets de petite taille, cette extension ne grève pas les bonnes performances de l'algorithme.

Nous recherchons donc dans un premier temps à l'aide de l'algorithme ***shift-add*** les segments $J6/7 - P7$ et $J8/7' - P7$. L'étape suivante consiste à construire toutes les hélices potentielles ($P7$,$P7'$). Nous utilisons pour cela les règles d'appariements de [LDM94] et rappelées en figure 7.7 (remarquons le renflement de la base 5 du segment $P7$). Une hélice ($P7$,$P7'$) est retenue si elle satisfait à un certain nombre de contraintes; en particulier, son score doit être supérieur à un seuil paramétrable et le type de chacun de ses appariements est vérifié.

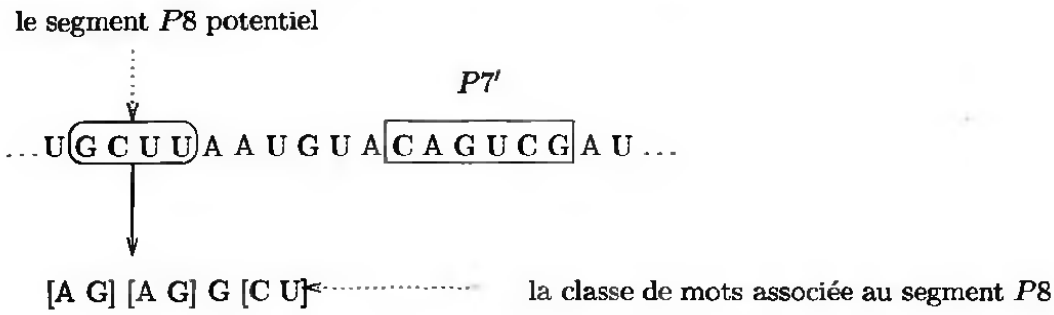
région $P7$:	(1 2 3)	4	5	6	7	8	9	10	
		|		|	|	|	|	|	
région $P7'$:		12		11	10	9	8	7	(6 5 4 3 2 1)

FIG. 7.7 – *Règle d'appariements des segments P7 et P7'.*

7.4.4 Recherche des combinaisons $\{(P7,P7'),(P8,P8')\}$

Une fois toutes les hélices $(P7,P7')$ potentielles localisées, l'étape suivante consiste à associer à chacune d'entre elles un ensemble d'hélices $(P8,P8')$ potentielles. Nous décrivons l'algorithme utilisé lorsque les erreurs d'appariements ne sont pas autorisées[7] dans les hélices $(P8,P8')$.

Nous ramenons cette phase de recherche à la recherche d'une classe de mots sans erreur; notre choix pour l'algorithme de recherche s'est donc naturellement porté sur ***shift-or***[8]. En effet, le segment $P8'$ est 6 ou 7 bases *avant* le segment $P7'$. Aussi, pour chaque segment $P8'$ potentiel, construisons-nous une classe de mots qui correspond à son complémentaire inverse. Par exemple,



La recherche du segment $P8$ (associé au segment $P8'$ courant) est parfaitement localisée puisqu'il se situe après le segment $P7$ associé (une grande insertion peut éventuellement être présente) et avant le segment $P8'$ considéré. Nous introduisons également dans le programme des limitations sur la construction de la classe de mots suivant la qualité des appariements désirée. Le programme permet par exemple d'interdire - *via* un fichier de configuration - la présence des appariements $G-U$; la classe de mot associé au segment $P8'$ dans l'exemple précédent serait dans ce cas $AAGC$.

Clairement, la longueur de la classe de mots correspond à la longueur minimale d'une hélice $(P8,P8')$. Cependant, l'hélice $(P8,P8')$ peut être plus grande dans la séquence. Aussi, nous considérons en fait dans notre algorithmes deux paramètres : la longueur minimale et maximale (notées respectivement $\ell_{\min}$ et $\ell_{\max}$) d'une hélice $(P8,P8')$. Nous construisons tout d'abord la classe de mots de longueur $\ell_{\max}$ et, en ajoutant un nouveau vecteur à l'algorithme ***shift-or***, nous recherchons ensuite les occurrences des préfixes de longueur au moins $\ell_{\min}$ de cette classe de mots dans la portion de séquence entre les segments $P7$ et $P7'$.

7. configuration par défaut.

8. La version sans erreur de l'algorithme ***shift-add***.

Nous vérifions également que le score de toute combinaison $\{(P7,P7'),(P8,P8')\}$ est supérieur à un seuil paramétrable. Toute hélice $(P7,P7')$ pour laquelle nous n'avons pas trouvé au moins une hélice $(P8,P8')$ potentielle associée est supprimée de la liste des candidats.

7.4.5 Recherche des combinaisons $\{(P6,P6'),(P7,P7')\}$

Une fois les ensembles $\{(P7,P7'),(P8,P8')\}$ calculés, l'étape suivante consiste à localiser toutes les hélices $(P6,P6')$ potentielles. Cette hélice se situe avant l'hélice $(P7,P7')$ à une distance fixe. Nous utilisons encore une fois pour cette étape une recherche de classe de mots sans erreur. De même que pour la recherche des hélices $(P8,P8')$, nous considérons à nouveau deux paramètres $\ell_{\min}$ et $\ell_{\max}$ qui correspondent respectivement à la longueur minimale et maximale d'une hélice $(P6,P6')$. À noter que cette fois-ci, en raison d'une longueur minimale trop petite, les erreurs d'appariement ne sont jamais autorisées.

Il est important de noter qu'en pratique une valeur trop petite du paramètre $\ell_{\min}$ dégrade considérablement les performances du programme. En particulier, si $\ell_{\min} = 2$, il convient alors d'imposer - *via* le fichier de configuration - deux appariements de type Watson-Crick.

Remarquons que les hélices potentielles $(P6,P6')$ et $(P8,P8')$ associées à une même hélice $(P7,P7')$ sont «*indépendantes*». Rien ne nous empêche en fait de rechercher les hélices potentielles $(P6,P6')$ *avant* les hélices potentielles $(P8,P8')$. Nous justifions notre choix. Les hélices $(P8,P8')$ sont - en général - sensiblement plus contraintes que les hélices $(P6,P6')$: `citron` propose par exemple qu'une hélice $(P8,P8')$ est constituée au minimum de trois paires de bases dont deux au moins sont des paires de Watson-Crick, tandis qu'une hélice $(P6,P6')$ est constituée au minimum de deux appariements de type Watson-Crick. Nous avons vérifié en pratique que pour des paramètres «*raisonnables*», la recherche des hélices potentielles $(P8,P8')$ avant les hélices $(P6,P6')$ accélère le temps de calcul en éliminant d'avantage de candidats $(P7,P7')$[9].

7.4.6 Recherche des combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$

Après la recherche des hélices $(P6,P6')$, nous pouvons rechercher toutes les combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$. En effet, le segment $P4'$ est situé à distance fixe du segment $P6$ (ils sont en général adjacents mais `orange` autorise toute distance fixe) et l'hélice $(P5,P5')$ est située entre les segments $P4$ et $P4'$.

La recherche des hélices $(P4,P4')$ est plus délicate puisque chaque hélice $(P4,P4')$ peut contenir un renflement[10]. La figure 7.8 donne les règles des appariements entre les segments $(P4,P4')$.

9. Il serait intéressant en pratique de doter `orange` d'un module lui permettant de déterminer *dynamiquement* l'ordre de recherche des hélices $(P6,P6')$ et $(P8,P8')$ suivant les paramètres de configuration (les contraintes structurelles des deux hélices).

10. Une option de configuration du programme (`p4_p4c_enable_bulge_p`) permet d'autoriser ou non la présence d'un renflement dans une hélice $(P4,P4')$. Lorsque les renflements ne sont pas autorisés, le temps de calcul des hélices $(P4,P4')$ est au moins divisé par 5; cette option peut donc être particulièrement utile pour un «*premier passage*» du programme, en particulier pour des séquences longues.

$P4$	1	2	3	4	5	6	
	|	|	|	|	|	|	
$P4'$	6	5	4	3	2	1	

$P4$	1	2	3	4	5	6	7
	|	|		|	|	|	|
$P4'$	6	5		4	3	2	1

$P4$	1	2	3	4	5	6	7
	|	|	|	|		|	|
$P4'$	6	5	4	3		2	1

$P4$	1	2	3	4		5	6
	|	|	|	|		|	|
$P4'$	7	6	5	4	3	2	1

$P4$	1	2		3	4	5	6
	|	|		|	|	|	|
$P4'$	7	6	5	4	3	2	1

FIG. 7.8 – *Règles des appariements entre les segments $P4$ et $P4'$.*

À partir de chaque hélice potentielle ($P6$,$P6'$), nous construisons la classe de mots associée au segment inverse $P4'$. Ceci n'est bien sûr possible que parce que le segment $P4'$ est situé à distance fixe du segment $P6$. De plus, nous prenons en compte les deux observation suivantes [LDM94] :

1. un segment $P4$ est suivi de AA ou GA;
2. un segment $P4'$ est précédé de A.

Autrement dit, nous n'effectuons dans un premier temps la recherche des hélices ($P4$,$P4'$) que si le segment $P4'$ est précédé de A[11]. De plus, nous insérons $[AG]A$ en suffixe de la classe de mots associée à la recherche du segment $P4$.

Il est spécifié dans [LDM94] qu'une hélice ($P4$,$P4'$) peut contenir au plus un appariement non conventionnel (*mismatch*). Ce cas est cependant relativement rare en pratique. Aussi, une option de configuration permet de régler le nombre de ce type d'appariement et, si celui-ci est nul, ***orange*** utilise ***shift-or*** à la place de ***shift-add***.

L'hélice ($P5$,$P5'$) est localisée entre les segments $P4$ et $P4'$ à distance «*fixe*». Pour chaque hélice ($P4$,$P4'$), nous recherchons une hélice ($P5$,$P5'$) associée. Si aucune ne peut être trouvée, l'hélice ($P4$,$P4'$) est supprimée de la liste des candidats. Nous vérifions également que le score de toute combinaison $\{(P4,P4'),(P5,P5')\}$ est supérieure à un seuil paramétrable; intuitivement, ce filtre permet de compenser une hélice ($P4$,$P4'$) «*moyenne*» par une «*bonne*» hélice ($P5$,$P5'$).

Enfin, toute hélice ($P6$,$P6'$) pour laquelle nous n'avons pas trouvé au moins une combinaison $\{(P4,P4'),(P5,P5')\}$ potentielle est supprimée de la liste des candidats. Nous vérifions également dans un dernier temps que le score de chaque combinaison $\{(P4,P4'),(P5,P5'),(P6,P6'),(P7,P7'),(P8,P8')\}$ est supérieur à un seuil paramétrable.

11. Pour être tout à fait exact, cette contrainte est utilisée dès la recherche des hélices ($P6$,$P6'$) afin de limiter leur nombre.

7.4.7 Recherche des hélices ($P3$,$P3'$)

Il nous reste à rechercher les hélices ($P3$,$P3'$) pour former des solutions contenant les hélices (Pi,Pi'), $3 \leq i \leq 8$. Le segment $P3$ est localisé avant le segment $P4$, tandis que le segment $P3'$ est localisé entre les segments $P7$ et $P8$.[12]. Le placement relatif des segments $P3$ et $P3'$ dépend de la présence ou non de longues insertions entre les segments $P3$ et $P4$ d'une part et $P7$ et $P3'$ d'autre part. Cependant, le segment $P3'$ est situé à distance fixe du segment $P8$. C'est donc à partir du segment $P3'$ que nous effectuons la recherche des hélices ($P3$,$P3'$).

Encore une fois, nous utilisons une recherche par classe de mots (sans erreur par défaut). Rappelons qu'à chaque hélice potentielle ($P7$,$P7'$) est associée un ensemble d'hélices ($P8$,$P8'$) et un ensemble de combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$. Pour chaque couple ($P7$,$P7'$) et ($P8$,$P8'$), nous déterminons un segment $P3'$ et construisons la classe de mots associée au segment $P3$. Nous localisons la recherche suivant les combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$ et vérifions la présence ou non d'un segment $P3$ associé. Des options de configuration permettent d'autoriser ou non les grandes insertions entre les segments $P3$ et $P4$ d'une part et $P7$ et $P3'$ d'autre part.

7.4.8 Score d'une solution

Le score d'une solution est la somme des scores des hélices qui la composent. Néanmoins, nous voulons privilégier les introns compacts. Aussi, calculons-nous un *bonus* d'autant plus élevé que la longueur de l'intron est petite. Nous utilisons la fonction suivante :

$$\text{bonus} = c\ \exp(-\ell/k).$$

où ℓ est la longueur de la solution (c'est-à-dire la distance entre le premier nucléotide du segment $P3$ et le dernier nucléotide du segment $P7'$) et c et k sont deux paramètres de configuration[13]. Quelques courbes sont données en figure 7.9.

7.4.9 Décomposition en solutions non chevauchantes

Le programme `orange` localise un ensemble d'introns auto-catalytiques dans une séquence donnée. De plus, il associe à chaque intron auto-catalytique un score qui reflète la qualité de ses appariements et le respect des contraintes imposées. Cependant, une séquence peut contenir plusieurs introns. Par exemple, dans le génome mitochondrial du champignon *Podospora anserina*, dont la taille est de l'ordre de 120 000 nucléotides, 34 introns de groupe I sont répartis dans une dizaine de gènes (un gène peut contenir plusieurs introns). Mais les introns localisés par `orange` ne sont pas tous *indépendants*, au sens où certains d'entre eux partagent une même partie de la séquence. Aussi, afin de fournir une décomposition en introns auto-catalytiques de la séquence, est-il nécessaire de déterminer[14] un sous-ensemble d'introns non chevauchants deux à deux. De plus, la décomposition proposée doit être en un certain

12. Remarquons à partir de l'enchaînement des hélices que les introns auto-catalytiques de groupe I contiennent un pseudo-nœud.

13. Par défaut, $c = 200$ et $k = 1000$. Les valeurs de c et k sont respectivement paramétrables *via* les options `mod_len_score_mult` et `mod_len_score_param`.

14. L'option est désactivée par défaut.

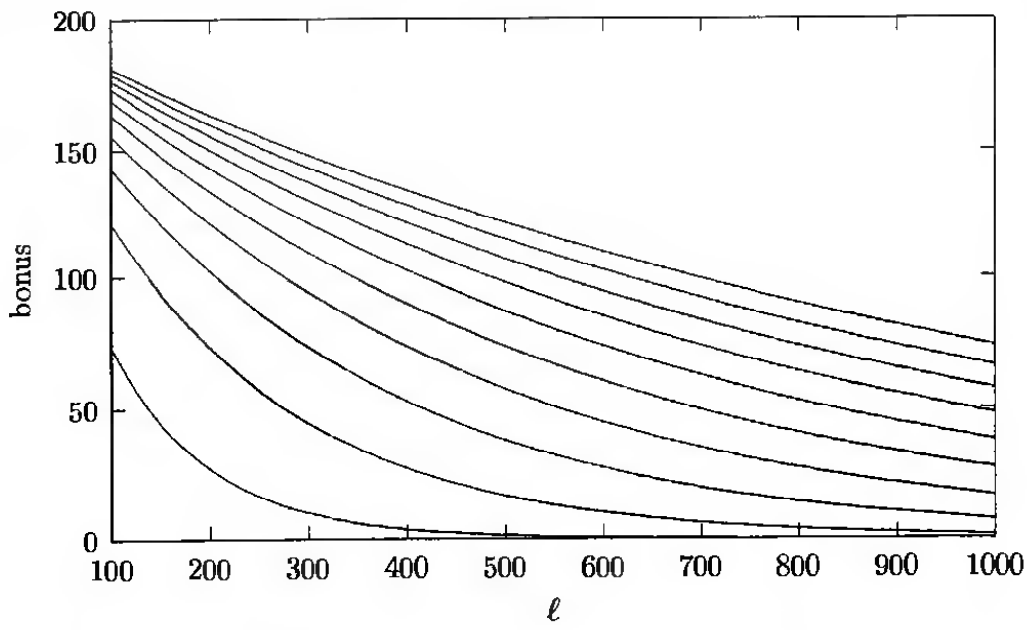


FIG. 7.9 – *Courbes de modification du score d'un intron auto-catalytique de groupe I bonus* $= c\ \exp(-\ell/k)$ *pour* $c = 200$ *et* k *allant de 100 à 1000 par pas de 100 en fonction de la longueur* ℓ *d'une solution.*

sens la plus «*vraisemblable*» parmi toutes les décompositions possibles. Nous faisons l'hypothèse dans la suite que la structure la plus «*vraisemblable*» est celle qui maximise la somme des scores des introns qui la composent.

Plus formellement, nous notons $\mathcal{I}$ l'ensemble des introns auto-catalytiques de groupe I identifiés par le programme `orange`:

$$\mathcal{I} = \{I_1, I_2, \ldots, I_k\}.$$

Pour tout $I_j \in \mathcal{I}$, nous notons $premier(I_j)$ et $dernier(I_j)$ respectivement la position du premier et du dernier nucléotide de l'intron I_j dans la séquence. Le score de l'intron I_j est noté $\omega(I_j)$. Par extension, pour tout sous-ensemble $\mathcal{I}' \subseteq \mathcal{I}$, nous notons $\omega(\mathcal{I}')$ la somme des scores des introns de I' :

$$\omega(\mathcal{I}') = \sum_{I_j \in \mathcal{I}'} \omega(I_j).$$

Nous disons que deux introns I_{j_1} et I_{j_2} sont *non chevauchants*, noté $I_{j_1} \cap I_{j_2} = \emptyset$, si $dernier(I_{j_1}) < premier(I_{j_2})$ ou $dernier(I_{j_2}) < premier(I_{j_1})$.

Étant donné un ensemble d'introns $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$, nous pouvons calculer un sous-ensemble d'introns non chevauchants deux à deux $\mathcal{I}' \subseteq \mathcal{I}$ de score maximal en calculant un ensemble stable pondéré de poids maximum dans un graphe d'intervalle. Plus précisément, nous associons à l'ensemble $\mathcal{I}$ le graphe $G_{\mathcal{I}} = (V, E)$ défini par :

$$V = \mathcal{I}$$
$$E = \{\{I_j, I_\ell\} \mid I_j \cap I_\ell \neq \emptyset\}$$

Nous vérifions facilement que $G_{\mathcal{I}}$ est un graphe d'intervalle, *i.e.* le graphe d'intersection des introns auto-catalytiques de l'ensemble $\mathcal{I}$. Aussi, un sous-ensemble $\mathcal{I}' \subseteq \mathcal{I}$ d'introns non chevauchants deux à deux de score maximal correspond-il à un ensemble stable pondéré de poids maximum dans le graphe $G_{\mathcal{I}}$.

Le calcul d'un stable de cardinalité maximale est un problème NP-complet en général [Kar72]. Cependant, la complexité est différente dans le cas des graphes d'intervalle. Le premier algorithme en temps polynomial pour trouver un ensemble stable de cardinalité maximale dans un graphe d'intervalles a été proposé par Gavril [Gav74a]. Ensuite, Gupta, Lee et Leung ont proposé un algorithme en temps $O(n \log n)$ [GLL82]. Plus récemment, Asano, Asano et Imai [AAI86] ont présenté un algorithme en temps $O(n \log n)$ pour trouver un ensemble stable pondéré maximum. Leug [Leu84] a proposé un algorithme pour générer tous les ensembles stables d'un graphe d'intervalles. Basé sur l'algorithme de Gupta, Lee et Leung [GLL82], ils ont obtenu un algorithme en temps $O(n^2 + \beta_I)$ où β_I est la somme des cardinaux des ensembles stables maximaux. Masuda, Nakajima, Kashiwabara et Fujisawa [KMNF92] donnent un algorithme en temps $O(n \log n + \beta_I)$ pour trouver tous les ensembles stables maximaux et un algorithme en temps $O(n \log n + \delta_I)$ pour trouver tous les ensembles stables de cardinalité maximale où δ_I est la somme des cardinalités des ensembles stables de cardinalité maximale.

7.4.10 Configuration

Le programme *orange* est entièrement paramétrable *via* un fichier de configuration. Citons en particulier :

- configuration globale : score minimum, longueur maximale, ... ;
- configuration des points d'ancrage : définition des classes de mots et nombre d'erreurs autorisées;
- configuration des pénalités : il est possible d'affecter des pénalités lorsqu'une solution contient de longues insertions entre les segments $P3$ et $P4$ d'une part et $P7$ et $P3'$ d'autre part;
- configuration fine de toutes les hélices : longueur, types d'appariements autorisés, ... ;
- configuration fine des segments inter-hélices : distance minimale et maximale.

Tous les paramètres ont des valeurs par défaut et sont donc facultatifs. Nous renvoyons le lecteur au fichier `intron-cfg-token.h` pour la liste complète et commentée des options et au fichier `intron-cfg-dflt.h` pour toutes les valeurs par défaut des options. La syntaxe est toujours la même :

$$\text{option} = \text{valeur}$$

Le symbole # indique le début d'un commentaire qui se poursuit jusqu'à la fin de la ligne. Une option spécifiée au lancement du programme remplace la valeur par défaut de cette dernière.

7.4.11 Filtrage

Les sous-sections précédentes ne décrivent que grossièrement l'algorithme que nous utilisons; nous avons en effet passé sous silence bon nombre de tests nous

permettant d'éliminer des solutions potentielles non valables ou «*trop proche*» d'une autre solution de score supérieur.

Par exemple , un filtre est chargé d'éliminer les «*doublons*». Plus précisément, nous associons à chaque solution un intervalle (du premier nucléotide du segment $P3$ au dernier nucléotide du segment $P7'$) et si plusieurs solutions correspondent à un même intervalle, nous ne conservons que celle de score maximal. Le but de ce filtre est double :

1. En pratique, il existe souvent des ensembles de solutions qui ne diffèrent les unes des autres que par de faibles modifications. Ce filtre permet de ne conserver qu'un unique représentant (le meilleur) parmi ces solutions.
2. Il permet d'accélérer significativement la recherche en solution non chevauchantes qui est l'étape immédiatement après.

Suivant le même principe, ***orange*** utilise des filtres pour :

- éliminer les «*doublons*» des combinaisons $\{(P4,P4'),(P5,P5')\}$;
- éliminer les «*doublons*» des combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$.

7.5 Quelques détails de mise en œuvre

7.5.1 Gestion des solutions partielles

Nous décrivons sommairement la structure de données que nous utilisons pour construire les solutions partielles. Notons tout de suite que la structure de données n'est pas optimale en espace; cependant, elle est assez efficace en pratique.

Rappelons brièvement la description du problème. Nous associons à chaque hélice $(P7,P7')$ un ensemble d'hélices potentielles $(P8,P8')$[15] et un ensemble de combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$. Ces deux ensembles sont indépendants. De même, toute combinaison $\{(P4,P4'),(P5,P5'),(P6,P6')\}$ se décompose elle-même en une hélice $\{P6,P6')\}$ et un ensemble de combinaisons $\{(P4,P4'),(P5,P5')\}$. Cette description suggère la structure de données décrite en figure 7.10. Outre sa simplicité, cette structure nous permet de plus de mettre en œuvre très facilement les remarques de la sous-section suivante.

7.5.2 Recherche des combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$

Nous commençons par deux remarques triviales mais importantes pour notre propos :

1. si deux hélices $(P7,P7')$ ne diffèrent que par leur segment $P7'$, alors nous pouvons toujours associer à chacune d'entre elles le même ensemble de combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$;
2. si deux hélices $(P6,P6')$ ne diffèrent que par leur segment $P6'$, alors nous pouvons toujours associer à chacune d'entre elles le même ensemble de combinaisons $\{(P4,P4'),(P5,P5')\}$.

En complément de la structure de données décrite en figure 7.10, nous utilisons deux dictionnaires[16] : le premier associe à chaque segment $P7$ l'ensemble des com-

15. Cet ensemble se limite souvent à un unique élément en pratique; ce n'est cependant pas toujours le cas et nous devons donc maintenir une structure d'ensemble.

16. Ces deux dictionnaires sont mis en œuvre par des tables de hachage (module **cht**).

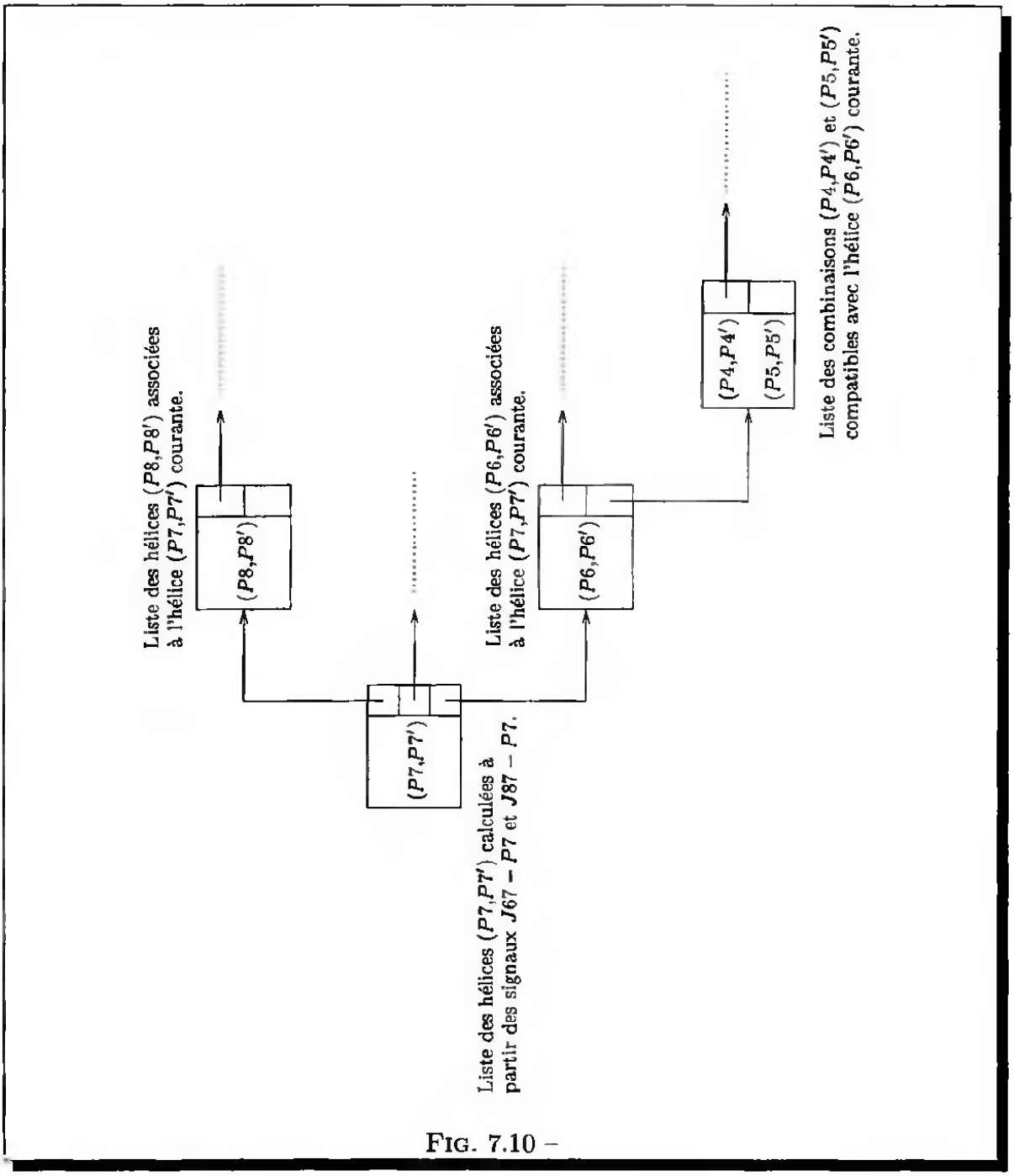

Liste des hélices (P8,P8') associées à l'hélice (P7,P7') courante.
(P8,P8')
(P7,P7')
Liste des hélices (P7,P7') calculées à partir des signaux J67 − P7 et J87 − P7.
Liste des hélices (P6,P6') associées à l'hélice (P7,P7') courante.
(P6,P6')
(P4,P4')
(P5,P5')
Liste des combinaisons (P4,P4') et (P5,P5') compatibles avec l'hélice (P6,P6') courante.

FIG. 7.10 –

binaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$ associées (si elles ont déjà été calculées), tandis que le second associe à chaque segment $P6'$ l'ensemble des combinaisons $\{(P4,P4'),(P5,P5')\}$ associées (si elles ont déjà été calculées). Nous remarquons tout de suite que ces associations peuvent facilement être mises en œuvre par des pointeurs sur les listes de la structure de données principale.

Nous utilisons ces dictionnaires de la façon suivante :

- une fois une hélice potentielle $(P7,P7')$ déterminée (et l'ensemble des hélices potentielles $(P8,P8')$ associées), nous consultons le dictionnaire pour vérifier si la recherche des combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$ n'a pas déjà été effectuée pour le segment $P7$ courant;
- une fois une hélice potentielle $(P6,P6')$ déterminée, nous consultons le dictionnaire pour vérifier si la recherche des combinaisons $\{(P4,P4'),(P5,P5')\}$ n'a pas déjà été effectuée pour le segment $P6$ courant.

Remarquons alors que si une entrée du dictionnaire correspond, la recherche se limite à une simple manipulation de pointeurs.

Nous avons constaté en pratique que ces dictionnaires permettent d'augmenter de façon significative la vitesse de l'algorithme en «*évitant*» bon nombre de recherches superflues. Cependant, cette accélération à un coût : nous devons maintenir les deux dictionnaires en mémoire (ce coût n'est malgré tout pas prohibitif). Le programme ***orange*** tente de limiter cet inconvénient en supprimant - *si nécessaire* - à intervalles réguliers des entrées du dictionnaire qui ne seront plus utilisées dans la suite[17].

7.6 Internet

`orange` est librement téléchargeable. Il sera de plus très bientôt accessible sur Internet[18].

17. Précisons nos propos. Les hélices $(P7,P7')$ sont examinées par ordre croissant de la position du premier nucléotide du segment $P7$. Aussi, pouvons-nous si nécessaire supprimer toutes les entrées dans le dictionnaire des segments $P6'$ dont la distance entre son dernier nucléotide et la position courante dans la séquence est supérieure à la distance maximale (calculée en début de programme).

18. Adresse temporaire `http://genetique.liafa.jussieu.fr`.

Annexe A

Formes triangulaires

Nous regroupons ici la plupart des résultats concernant les problèmes sur les matrices et sous-matrices triangulaires. Consulter les sections 2.1, 2.2 et 2.3 pour les notations et définitions.

A.1 Matrice triangulaire

- **Données** : Une (0,1)-matrice A d'ordre n.
- **Solution** : Deux matrices de permutation P et Q d'ordre n telles que $PAQ^T \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_n, \triangledown_n, \triangleleft_n, \triangleright_n\}$.
- **Complexité** : inconnue.
- **Commentaires** : Le problème est résoluble est temps $O(n^2)$ si $\mathrm{per}(A) \geq 1$ (théorème 4.3.4 et proposition 4.3.7). Il est également résoluble en temps polynomial si la matrice A contient au plus $n+1$ éléments non nuls (proposition 3.5.7).

- **Données** : Une (0,1)-matrice A d'ordre n.
- **Solution** : Une matrice de permutation P d'ordre n telle que $PAP^T \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_n, \triangledown_n\}$
- **Complexité** : NP-complet.
- **Commentaires** : Équivalent au problème ARRANGEMENT LINÉAIRE TRIANGULAIRE (proposition 4.4.8). Le problème est NP-complet même si A est une matrice symétrique avec au plus trois éléments non nuls par ligne et par colonne (proposition 4.4.13). Il est également NP-complet si A est une matrice irréductible. La complexité du problème reste inconnue si A est la matrice d'adjacence d'un graphe biparti ou d'un graphe de séparation. Le problème est résoluble en temps polynomial si A est la matrice d'adjacence d'un graphe acyclique (corollaire 4.4.16).

- **Données** : Une (0,1)-matrice A d'ordre n.
- **Solution** : Une matrice de permutation P d'ordre n telle que $PAP^T \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangleleft_n, \triangleright_n\}$.

- **Complexité** : $\Theta(n+k)$ où k est le nombre d'éléments non nuls dans A.
- **Commentaires** : Équivalent au tri topologique d'un graphe orienté [CLR92].

A.2 Sous-matrice principale

Nous rappelons dans cette sous-section les résultats de complexité sur les formes triangulaires des sous-matrices principale. Étant donnés une (0,1)-matrice symétrique A d'ordre n et un entier positif K, il s'agit de décider s'il existe une sous-matrice principale A' de A d'ordre K qui satisfait une propriété demandée. Autrement dit, il faut décider s'il existe une matrice de permutation P d'ordre n telle que :

$$PAP^T = \begin{bmatrix} A' & * \\ * & * \end{bmatrix}$$

où A' est une matrice d'ordre K qui satisfait une propriété demandée.

- **Solution** : $A' \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangledown_K\}$.
- **Complexité** : NP-complet..
- **Commentaires** : Équivalent au problème ARRANGEMENT LINÉAIRE TRIANGULAIRE pour $K = n$ (proposition 4.4.8).

- **Solution** : $A' \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangledown_K\}$.
- **Complexité** : NP-complet.
- **Commentaires** : Équivalent à la recherche du plus grand sous-graphe acyclique dans un graphe orienté (corollaire 4.6.4) [Kar72].

- **Solution** : $\mathbb{C}_K \leq A' \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangledown_K\}$.
- **Complexité** : NP-complet.
- **Commentaires** : Équivalent à la recherche d'un couplage unique restreint dans un graphe [GHL].

- **Solution** : $A' = \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangledown_K\}$.
- **Complexité** : NP-complet.
- **Commentaires** : Problème ISO SOUS-MATRICE EXACT-$\triangle$ (proposition 4.6.11).

- **Solution**: $A' = \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangledown_K\}$.
- **Complexité**: NP-complet.
- **Commentaires**: Problème ISO SOUS-MATRICE EXACT-$\triangle$ (proposition 4.6.16).

- **Solution**: $A' = \mathbb{C}_K$
- **Complexité**: NP-complet
- **Commentaires**: Équivalent à la recherche d'un couplage fort dans un graphe [Lew96].

A.3 Sous-matrice

Nous présentons maintenant les résultats de complexité sur les formes triangulaires des sous-matrices. Étant donnés une (0,1)-matrice A d'ordre n et un entier positif K, il s'agit de décider s'il existe une sous-matrice A' de A d'ordre K qui satisfait une propriété demandée. Autrement dit, il faut décider s'il existe des matrices de permutation P d'ordre m et Q d'ordre n telle que :

$$PAQ^T = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

où A_{11} est une matrice d'ordre K qui satisfait une propriété demandée.

- **Solution** : $\mathbb{I}_K \leq A_{11} \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangledown_K\}$.
- **Complexité** : NP-complet.
- **Commentaires** : Équivalent à la recherche d'un couplage unique restreint dans un graphe biparti [GHL]. Le problème $\mathbb{C}_K \leq A_{11} \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangledown_K\}$ est également NP-complet.

- **Solution** : $A_{11} \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangle_K, \triangledown_K, \triangledown_K\}$ et $A_{12} = 0_{K,n-K}$.
- **Complexité** : NP-complet.
- **Commentaires** : Équivalent au problème ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX (proposition 4.7.4). Le problème est NP-complet même si chaque ligne de la matrice A contient au plus trois éléments non nuls.

- **Solution** : $A_{11} \leq \mathbb{T}$ pour $\mathbb{T} \in \{\triangle_K, \triangle_K, \triangledown_K, \triangledown_K\}$.
- **Complexité** : inconnue.
- **Commentaires** : Le problème $A_{11} \geq B$, $B \in \{\triangle_K, \triangle_K, \triangledown_K, \triangledown_K\}$, est de même complexité (proposition 4.5.10).

- **Solution**: $A_{11} = \mathbb{T}$ pour $\mathbb{T}i \in \{\triangle_K, \triangle_K, \triangledown_K, \triangledown_K\}$.
- **Complexité**: NP-complet.
- **Commentaires**: Équivalent à la recherche d'un couplage exact-triangulaire dans un graphe biparti (poposition 4.5.6).
- **références**:

Annexe B

Compléments sur les graphes de 2-intervalles

Sommaire

B.1 Introduction **233**
B.2 Graphe d'arêtes **234**
B.3 Partition **235**
B.4 Coupe **237**
B.5 Clique **238**

B.1 Introduction

En complément du chapitre 6, nous présentons ici quelques problèmes classiques restreints à la classe des graphes de 2-intervalles[1]. Nous avons déjà montré[2] que STABLE est un problème NP-complet même si G est un graphe de 2-intervalles (à support unitaire). Étant donné un graphe $G = (V,E)$, un sous-ensemble $V' \subseteq V$ est un ensemble stable si et seulement si $V \setminus V'$ est une couverture des sommets de G. Aussi, COUVERTURE DES SOMMETS est également un problème NP-complet restreint à la classe des graphes de 2-intervalles.

Nous établissons dans un premier temps une équivalence entre la classe des graphes d'arêtes et la classe des graphes de 2-intervalles à support disjoint. Cette équivalence va nous permettre de montrer facilement que les problèmes PARTITION EN CLIQUES, INDEX CHROMATIQUE et K-COLORIAGE sont NP-complets même si G est un graphe de 2-intervalles. Nous étudierons ensuite la complexité des problèmes COUPE MAX SIMPLE et ENSEMBLE DE COUPE STABLE. Pour terminer, nous poserons la question concernant la complexité du problème CLIQUE restreint à la classe des graphes de 2-intervalles.

1. sous-section 6.4.3.
2. corollaire 6.5.3 page 179.

B.2 Graphe d'arêtes

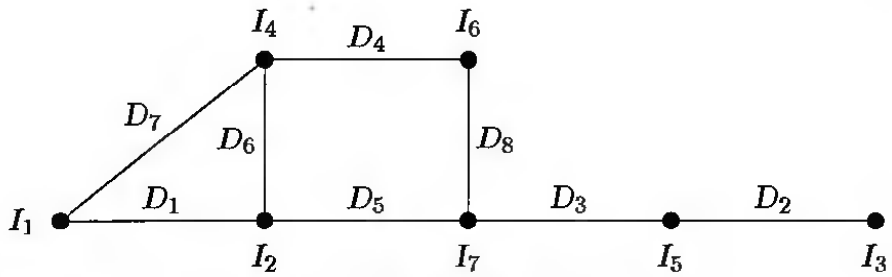
DÉFINITION B.2.1 *Soit $G = (V,E)$ un graphe d'ordre n. Le* graphe d'arêtes *de G, noté $L(G)$*[3]*, est le graphe dont les sommets sont les arêtes de G, et deux sommets sont adjacents dans $L(G)$ si les arêtes associées sont incidentes dans G.*

Autrement dit, le graphe d'arêtes de G est le graphe d'intersection de toutes les arêtes de G: $G \cong \Omega(E)$. Le graphe d'arêtes est une structure classique et très étudiée [Har71, Bol79, BR91, Jun99]. Par exemple, si G est eulérien, alors $L(G)$ est également eulérien; l'inverse n'est pas vrai en général. De même, si G est eulérien, alors $L(G)$ est hamiltonien; encore une fois, l'inverse n'est pas vrai en général [Jun99].

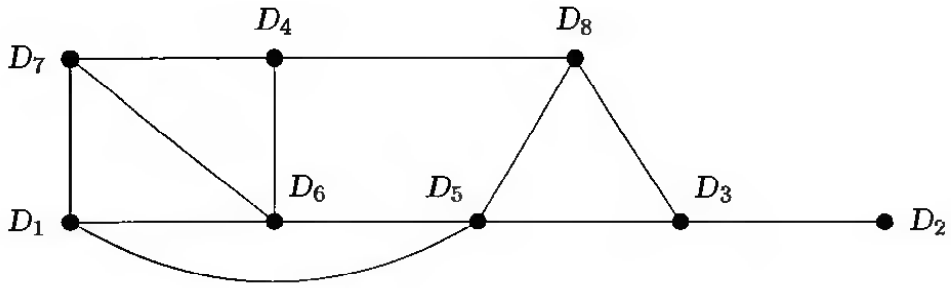
Il existe une relation étroite entre la classe des graphes d'arêtes et la classe des graphes de 2-intervalles. Soit $\mathcal{F}$ une famille de 2-intervalles à support disjoint. Le *codage* de la famille $\mathcal{F}$ est un graphe noté $G_{\mathcal{F}} = (V,E)$ défini par: $V = supp(\mathcal{F})$ et $E = \mathcal{F}$. En d'autres termes, chaque sommet de $G_{\mathcal{F}}$ représente un intervalle du support et chaque arête un 2-intervalle de $\mathcal{F}$. Deux sommets sont donc adjacents s'ils représentent un 2-intervalle de la famille $\mathcal{F}$.

codage

EXEMPLE 20 *Soit $\mathcal{F}$ la famille de 2-intervalles à support disjoint de la figure 6.6 page 172. Nous vérifions $supp(\mathcal{F}) = \{I_i \mid 1 \leq i \leq 7\}$. Le codage de $\mathcal{F}$ est donné par le graphe $G_{\mathcal{F}}$ défini par:*



Le graphe de 2-intervalles associé $G = \Omega(\mathcal{F})$ est donné par:



Le lemme suivant caractérise les graphes de 2-intervalles qui sont associés à une famille de 2-intervalles à support disjoint.

LEMME B.2.2 *Soient $\mathcal{F}$ une famille de 2-intervalles à support disjoint et $G = \Omega(\mathcal{F})$ le graphe de 2-intervalles associé. Le graphe d'arêtes du codage de $\mathcal{F}$ est isomorphe au graphe de 2-intervalles associé à $\mathcal{F}$,* i.e. $L(G_{\mathcal{F}}) \cong G$.

3. pour *line graph*.

PREUVE

Si D est une famille de 2-intervalles à support disjoint, alors chaque sommet du codage $G_{\mathcal{F}}$ représente un intervalle du support et chaque arête représente un 2-intervalle. Par conséquent, chaque sommet du graphe d'arêtes $L(G_{\mathcal{F}})$ représente un 2-intervalle, et deux sommets sont adjacents dans $L(G_{\mathcal{F}})$ si et seulement si les deux 2-intervalles associés s'intersectent. $\boxed{\text{B.2.2}}$

L'équivalence entre les deux représentations est triviale. Cependant, elle n'est pas stérile. En effet, nous pouvons tirer partie de nombreux résultats sur une structure classique (le graphe d'arêtes) pour étudier des problèmes restreints à la classe des graphes de 2-intervalles.

COROLLAIRE B.2.3 *Soit* Π *un problème de décision sur les graphes. Si* Π *est un problème* NP*-complet même si* G *est un graphe d'arêtes, alors* Π *est un problème* NP*-complet même si* G *est un graphe de* 2*-intervalles.*

REMARQUE B.2.4 Un problème Π peut être résoluble en temps polynomial si G est un graphe d'arêtes et NP-complet si G est un graphe de 2-intervalles. En effet, le corollaire B.2.3 implique uniquement que le problème Π est résoluble en temps polynomial si G est un graphe de 2-intervalles à support disjoint. C'est par exemple le cas pour le problème STABLE ◇

Le corollaire B.2.3 va nous permettre dans la suite de simplifier l'étude de la complexité de certains problèmes. En effet, il suffit maintenant de montrer qu'un problème est NP-complet même si G est un graphe d'arêtes pour montrer que ce même problème est NP-complet même si G est un graphe de 2-intervalles. Nous montrons ainsi par exemple que les problèmes PARTITION EN CLIQUES, K-COLORIAGE et ENSEMBLE DE COUPE STABLE sont NP-complets même si G est un graphe de 2-intervalles à support disjoint.

B.3 Partition

Nous considérons dans cette section les problèmes PARTITION EN CLIQUES et K-COLORIAGE dans le cas des graphes de 2-intervalles. Ces deux problèmes sont NP-complets dans le cas général [GJ79].

THÉORÈME B.3.1 ([GJ79]) PARTITION EN CLIQUES *est un problème* NP*-complet même si* G *est un graphe d'arêtes.*

La preuve est attribuée à E. Arjomandi dans une communication privée. Un résultat plus précis est donné par le théorème suivant.

THÉORÈME B.3.2 ([VR96]) PARTITION EN CLIQUES *est un problème* NP*-complet même si* G *est un graphe d'arêtes triparti planaire avec* $\Delta(G) = 3$.

Les auteurs utilisent une réduction depuis le problème COUVERTURE DES SOMMETS dans le cas des graphes cubiques planaires [GJ79]. Soient $G = (V,E)$ un graphe planaire cubique et G' le graphe obtenu à partir de G en remplaçant chaque arête

par un chemin de longueur 3. Alors, il existe une partition en $|E| + K$ cliques dans $L(G')$ si et seulement si il existe une couverture de sommets de taille K dans G.

Il suffit maintenant d'appliquer le corollaire B.2.3.

COROLLAIRE B.3.3 PARTITION EN CLIQUES *est un problème* NP-*complet même si G est un graphe de 2-intervalles à support disjoint.*

PREUVE

Théorème B.3.2 et lemme B.2.2. $\boxed{\text{B.3.3}}$

nombre chromatique

Le *nombre chromatique* d'un graphe G, noté $\chi(G)$, est le nombre minimum de couleurs nécessaires pour colorier les sommets d'un graphe de sorte que deux sommets adjacents n'aient pas la même couleur. Le problème de décision associé est en général appelé K-COLORIAGE: étant donné un graphe G et un entier positif K, avons-nous $\chi(G) \leq K$? R.M. Karp [Kar72] a montré que ce problème est NP-complet dans le cas général pour $K \geq 3$. Le problème reste NP-complet même si $K = 3$ et G est un graphe planaire avec $\Delta(G) \leq 4$ [GJS76]. Si P $\neq$ NP, il n'existe même pas un algorithme d'approximation en temps polynomial qui nécessite au plus $2\chi(G)$ couleurs [GJ76]. Si G est un graphe parfait, le problème K-COLORIAGE est résoluble en temps polynomial. Par exemple, un graphe est biparti si et seulement si il est 2-coloriable.

index chromatique

L'*index chromatique* d'un graphe, noté $\chi'(G)$, est le nombre minimum de couleurs nécessaires pour colorier les arêtes du graphe de sorte que deux arêtes adjacentes n'aient pas la même couleur. Remarquons que $\chi'(G)$ est le nombre chromatique $\chi(L(G))$ du graphe d'arêtes de G. L'index chromatique d'un graphe G vérifie toujours:

$$\chi'(G) \geq \Delta(G)$$

où $\Delta(G)$ est le degré maximum d'un sommet de G. En effet, toutes les arêtes incidentes à un même sommet doivent être coloriées avec des couleurs différentes. Le théorème de Vizing [Viz64][4] montre que l'index chromatique d'un graphe G est soit $\Delta(G)$ soit $\Delta(G)+1$. Cependant, $\chi'(G) = \Delta(G)$ si G est un graphe biparti. Bien que le théorème de Vizing donne $\Delta(G) \leq \chi'(G) \leq \Delta(G)+1$, I. Hoyler [Hoy81] a montré que décider si l'index chromatique d'un graphe G est $\Delta(G) + 1$ est un problème NP-complet. En fait, I. Hoyler a prouvé ce résultat sur les graphes cubiques.

THÉORÈME B.3.4 ([HOY81]) *Décider si l'index chromatique d'un graphe cubique est 3 est un problème* NP-*complet.*

Nous pouvons encore une fois en déduire immédiatement la complexité du calcul du nombre chromatique d'un graphe de 2-intervalles (à support disjoint).

COROLLAIRE B.3.5 K-COLORIAGE *est un problème* NP-*complet même si G est un graphe de 2-intervalles à support disjoint.*

PREUVE

Pour tout graphe G, il vient $\chi'(G) = \chi(L(G))$. Le théorème B.3.4 et le lemme B.2.2 permettent de conclure. $\boxed{\text{B.3.5}}$

4. L'article original est en Russe, mais une preuve en anglais est donnée, par exemple, dans [Bol79] et [Jun99].

B.4 Coupe

Soit $G = (V,E)$ un graphe d'ordre n et une fonction ω de E dans $\mathbb{Z}^+$. Une *coupe* est une partition $V = S \cup T$. La *capacité* d'une coupe, notée $c(S,T)$, est définie par :

$$c(S,T) = \sum_{\substack{\{u,v\} \in E \\ u \in S,\, v \in T}} \omega(\{u,v\})$$

Plusieurs problèmes sont associés à cette définition. Par exemple, le problème COUPE MAX recherche une coupe (S,T) telle que $c(S,T) \geq K$, où K est un paramètre de l'instance. Si $\omega(e) = 1$ pour tout $e \in E$, le problème est appelé COUPE MAX SIMPLE.

Le problème COUPE MAX a été prouvé NP-complet par R.M. Karp [Kar72] en utilisant une réduction depuis le problème PARTITION. La variante COUPE MAX SIMPLE est également NP-complète [GJS76] (réduction du problème MAX 2SAT). Remarquons néanmoins que ce problème est résoluble en temps polynomial si G est un graphe planaire [Had75].

Un *graphe de k-séparation* est un graphe de séparation où tous les sommets de l'ensemble stable sont adjacents à exactement k sommets de la clique. Nous nous intéressons en particulier aux graphes de 2-séparation en raison des résultats suivants. ***graphe de k-séparation***

LEMME B.4.1 ([BJ94]) *Tout graphe de 2-séparation est un graphe de 2-intervalles.*

THÉORÈME B.4.2 ([BJ94]) COUPE MAX SIMPLE *est un problème* NP*-complet même si G est un graphe de 2-séparation.*

Les auteurs utilisent une réduction depuis le problème COUPE MAX SIMPLE (non restreint) et construisent un graphe de 2-séparation $G' = (V',E')$ défini par:

$$V' = V \cup E^c$$
$$E' = \{\{u,v\} \mid u,v \in V \ \wedge \ u \neq v\} \cup \{\{u,e\} \mid u \in V \ \wedge \ e \in E^c \ \wedge \ u \in e\}$$

Le corollaire suivant est immédiat.

COROLLAIRE B.4.3 COUPE MAX SIMPLE *est un problème* NP*-complet même si G est un graphe de 2-intervalles.*

Un *ensemble de coupe* est un sous-ensemble de sommets dont la suppression déconnecte le graphe. Par extension, un *ensemble de coupe stable* est un ensemble de coupe qui est également un ensemble stable. ***ensemble de coupe***

Un graphe G est dit *décomposable* s'il est possible de colorier ses sommets avec deux couleurs de sorte que chaque couleur soit utilisée au moins une fois et le voisinage de chaque sommet u colorié c_1 contient au plus un sommet de couleur c_2. Autrement dit, un graphe est décomposable s'il existe une partition $V = V_1 \cup V_2$ des sommets telle que l'ensemble d'arêtes $\{u,v\} \in E$ avec $u \in v_1$ et $v \in v_2$ forment un couplage induit. La reconnaissance des graphes décomposables est un problème NP-complet dans le cas général [Chv84]. ***décomposable***

THÉORÈME B.4.4 ([MOS89]) *Soit G un graphe biparti de degré minimum 2. Décider si G est un graphe décomposable est un problème* NP*-complet.*

En utilisant le théorème précédent, il a été récemment montré le résultat suivant:

THÉORÈME B.4.5 ([BDLS00]) *Soit G un graphe biparti. Décider s'il existe un ensemble de coupe stable dans $L(G)$ est un problème* NP*-complet.*

Remarquons que ce théorème montre également que la recherche d'un ensemble de coupe stable dans un graphe parfait [Gol80] est un problème NP-complet car le graphe d'arêtes d'un graphe biparti est un graphe parfait.

COROLLAIRE B.4.6 *Soit G un graphe de 2-intervalles à support disjoint. Décider s'il existe un ensemble de coupe stable dans G est un problème* NP*-complet.*

PREUVE
Lemme B.2.2 et théorème B.4.5. $\boxed{\text{B.4.6}}$

B.5 Clique

Nous abordons pour terminer le problème CLIQUE restreint à la classe des graphes de 2-intervalles. Le complémentaire d'un graphe de 2-intervalles n'est pas nécessairement un graphe de 2-intervalles. Par conséquent, nous ne pouvons pas utiliser le fait que STABLE soit un problème NP-complet même si G est un graphe de 2-intervalles.

PROPRIÉTÉ B.5.1 *Soit $L(G)$ un graphe d'arêtes. Il existe une clique de taille $K > 3$ dans $L(G)$ si et seulement si il existe un sommet de G de degré K.*

Si $\mathcal{F}$ une famille de 2-intervalles à support disjoint, la clique de cardinalité maximale dans $\Omega(\mathcal{F})$ correspond donc au sommet de degré maximum dans le codage $G_{\mathcal{F}}$[5]. Autrement dit, nous pouvons calculer en temps polynomial la clique de cardinalité maximale si $\mathcal{F}$ est une famille de 2-intervalles à support disjoint.

Le problème reste posé dans le cas général, *i.e.* si $\mathcal{F}$ est une famille de 2-intervalles quelconque.

5. Les cliques de taille 3 dans $L(G_{\mathcal{F}})$ constituent une exception, elles se traduisent par la présence dans $G_{\mathcal{F}}$ d'un triangle ou d'une 3-étoile.

Annexe C

Description des modules du logiciel *orange*

Nous décrivons très brièvement dans cette annexe les pricipaux modules du logiciel `Orange`.

TAB. C.1 – Tableau des principaux fichiers utilisés dans le programme ***orange***.

`dlist.h`, `dlist.c`, `dlist-node.c`, `dlist-iterator.c`: Définitions et gestion de listes doublement chaînées. À noter que la routine de filtrage ***dlist_reduce*** est utilisée intensivement dans le programme. Le module `dlist-iterator` propose un itérateur unidirectionnel sur la structure de données. Pour accélérer la manipulation des listes, un grand nombre de routines sont également disponibles sous forme de macros.
`cht.h`, `cht-bucket.h`, `cht.c` Définition et manipulation des tables de hachage. La taille de la table doit être spécifiée lors de sa création. À noter qu'il est possible de spécifier - toujours lors de la création - un pointeur (éventuellement `NULL`) sur une fonction appelée automatiquement avant chaque suppression d'un élément.
`str.h`, `strtools.h`, `str.c`, `strtools.c`: Diverses manipulations de chaînes de caractères. Les modules `str` et `strtools` proposent en particulier différentes routines de remplissage, de nettoyage et de conversion dédiées à notre problème. La routine ***get_file*** permet de projeter le contenu d'un fichier ARN dans un tampon alloué dynamiquement.

`intron-display.h`, `intron-display-defs.h`, `intron-display.c`:

Gestion de l'affichage. Le module `intron-display` met en œuvre toutes les routines d'affichage des solutions. Le nombre de colonnes utilisées pour l'affichage est déterminé par la constante symbolique `INTRON_DISPLAY_COLUMNS` fichier `intron-display-defs.h` (80 par défaut).

`intron-cfg.h`, `intron-cfg-token.h`, `intron-cfg.c`:

Gestion de la configuration du logiciel. Le fichier `intron-cfg-token.h` spécifie la syntaxe de tous les mots-clés ainsi que des valeurs booléennes. Le module `intron-cfg` est en particulier chargé de la lecture des fichiers de configuration; il propose également une routine d'affichage de la configuration courante (***intron_display_config***).

`intron-cfg-dflt.h`, `intron-cfg-dflt.c`

Gestion de la configuration par défaut. Le module `intron-cfg-dflt` initialise chaque paramètre du programme; toutes ces valeurs sont définies dans le fichier `intron-cfg-dflt.h`. À noter que toute valeur d'un paramètre spécifiée à l'aide du module `intron-cfg` remplace la valeur par défaut.

`intron-cfg-check.c`:

Validation d'une configuration. Le module `intron-cfg-check` met en œuvre quelques tests - simples - de cohérence.

`orange.h`, `orange-infos.h`, `orange.c`, `orange-infos.c`:

Programme principal et gestion des options.

`ACGII.c`, `ACGII.h`

Définition et manipulation des introns auto-catalytiques de groupe I. Chaque solution est une instance de cet objet.

`shift-class.h`, `shift-class.c`

Création et manipulation des classes de mots. Les routines du module `shift-class` sont appelées avant chaque utilisation des algorithmes ***shift-add*** et ***shift-or***.

`bit-vector.h`, `bit-vector.c`

Gestion dynamique des vecteurs de bits. Les algorithmes ***shift-add*** et ***shift-or*** utilisent intensivement ce module.

`shift-add.h`, `shift-add.c`, `shift-or.h`, `shift-or.c`:

Mise en œuvre des algorithmes ***shift-or*** et ***shift-add*** [BYG92].

`intron-MWIS.c` Calcul d'une décomposition indépendante de score maximum en utilisant un algorithme de recherche d'un ensemble stable de poids maximal dans un graphe d'intervalle.
`factor.h`, `factor.c`, `stem.h`, `stem.c`: Définition et manipulation des segments et des hélices. À noter que les modules `factor` et `stem` proposent également des versions pondérées.
`memory.h`, `memory.c`: Gestion rapide de l'allocation mémoire. L'utilisation de cette librairie est globalement activée par l'option de compilation `USE_MEMORY_POOL`..
`error.h`, `error.c` Gestion des erreurs.
`intron-search.h`, `intron-search.c` Recherche des introns auto-catalytiques de groupe I. En particulier, le fichier `intron-search.c` contient tous les appels aux routines de recherche.
`intron-search-p7-p7c.c` Gestion de la recherche des hélices ($P7$,$P7'$). Ce module met en œuvre des routines de localisation des segments $J67 - P7$ et $J87' - P7'$ en utilisant le module `shift-add`; il permet également à partir de ces segments de construire une liste d'hélices potentielles ($P7$,$P7'$).
`intron-search-p8-p8c.c` Gestion de la recherche des hélices ($P8$,$P8'$) à partir de la liste des hélices potentielles ($P7$,$P7'$). Le module utilise ***shift-add*** ou ***shift-or*** suivant les valeurs des paramètres de configuration.
`intron-search-p4-p6c-region.c` `intron-search-core.c` Gestion de la recherche des combinaisons $\{(P4,P4'),(P5,P5'),(P6,P6')\}$ pour un segment $P7$ donné.
`intron-search-core.c` Gestion des solutions complètes potentielles. En particulier, ce module gère la recherche des hélices potentielles ($P3$,$P3'$).

Annexe D

Exemple d'exécution du programme orange

Nous présentons ici un exemple d'exécution - format texte - du logiciel orange. Nous avons volontairement choisi pour cette présentation une séquence très courte dont la structure secondaire est déjà connue et disponible dans les banques de séquences. Notre choix s'est porté sur *Pneumocystis carinii* (code GenBank L13614), un champignon eucaryote dont la structure secondaire est donnée en figure D.1.

Le résultat du logiciel orange sur cette même séquence est donnée ci-après. Nous avons demandé pour cet exemple uniquement l'affichage de la meilleure solution. En effet, le logiciel trouve une seconde solution qui est une variante mineure de la première. Pour chaque solution, l'affichage se décompose en quatre parties:

1. Affichage de la position et du score de la solution courante. La valeur de `mod.len.` reflète la compacité de l'intron; cette quantité peut être importante, nous voyons par exemple sur notre exemple que le tiers du score total provient de ce facteur. Le booléen `ins73` (resp. `ins34`) indique si la solution contient une «*longue*» insertion entre les segments $P3$ et $P4$ (resp. $P7$ et $P3'$).
2. Affichage de la portion de séquence associée à la solution courante. Des marqueurs ajustés en position et en longueur situent les hélices sur la portion de séquence.
3. Affichage des informations concernant chaque hélice: position, score, composition en paires de bases, ...Les informations «*exceptionnelles*» sont toujours précédées par le mot-clé `misc`.
4. Affichage de différent «*groupes d'hélices*». Ces information sont en particuliers utiles pour déterminer les valeurs des paramètres de filtrage.

Par souci de clarté, nous avons, sur cet exemple, produit une sortie minimale. Cependant, le logiciel orange permet en plus:

- d'afficher tous les paramètres de configuration;
- d'afficher les résulats partiels, y compris les combinaisons d'hélices potentielles dont aucune solution finale n'est issue;
- de calculer un sous-ensemble de solutions indépendantes (non chevauchantes) dont la somme des scores est maximale;
- ...

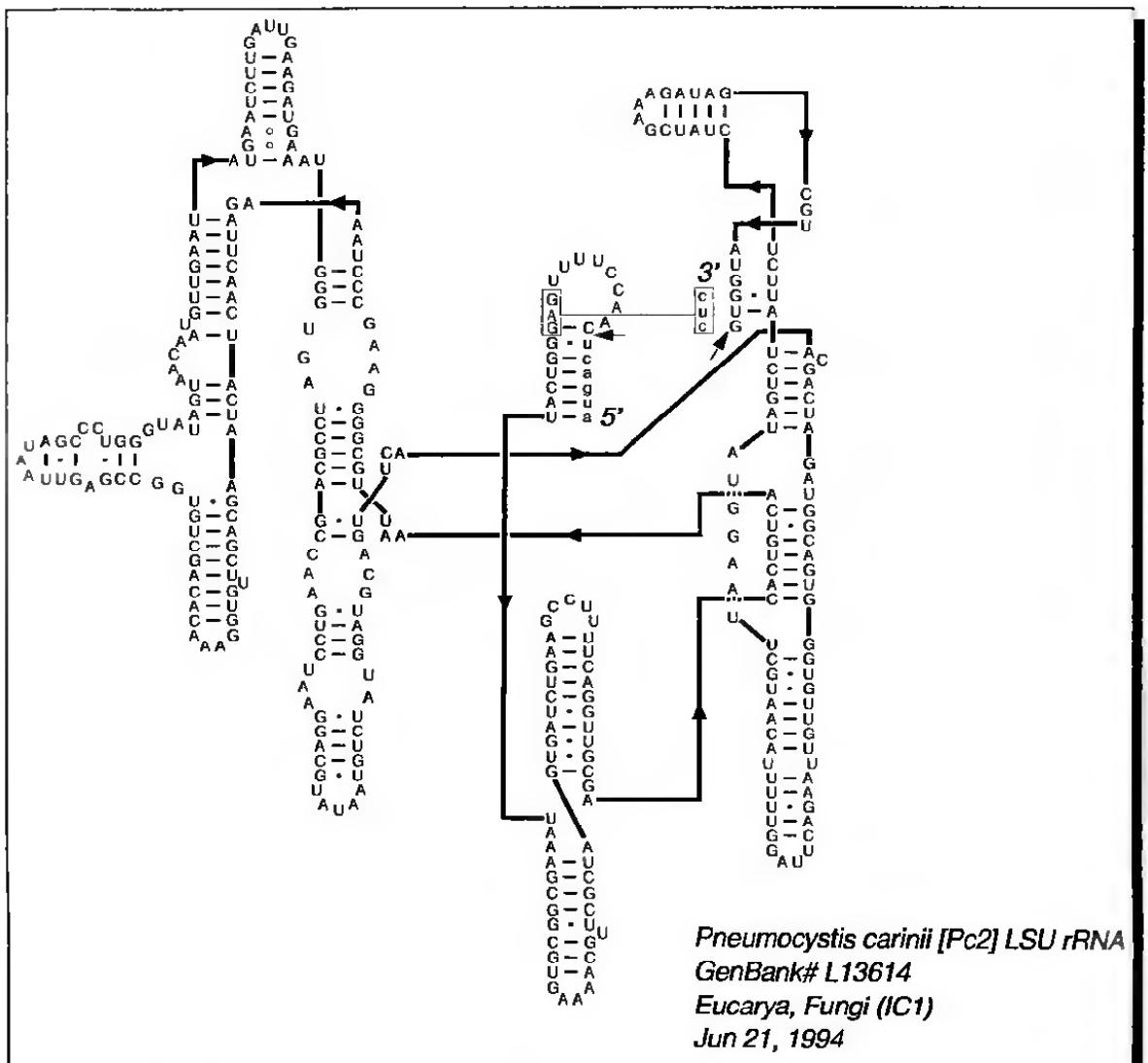


FIG. D.1 – *Structure secondaire de* Pneumocystis carinii *[LL93, DG94].*

```
[vialette@orange-0.1.1]$ ./bin/orange  --config=expl-config  --max=1  L13614.seq

** this is orange version 0.1.1
** Copyright 1999-2001 Stephane Vialette (vialette@liafa.jussieu.fr)

+-----------------------------------------------------+
|                                                     |
|    orange solutions sorted by decreasing score      |
|                                                     |
+-----------------------------------------------------+

display 1 / 2 orange solutions

#1

gen. infos
==========
Sequence: "L13614"
Score: 458 (301 + mod. len. 157)
Core Len. 244 (Pos. 1st nt. 70 Pos. last nt. 313)
ins34: false  ins73: false


display seq.
============
CAGGTTGCGACACTGTCAAATTGCGGGGAAGCCCTAAAGATTCAACTACTAAGCAGCTTGTGGAAACACAGCTGTGGCCG
          <P3-----   <P4----    <P5
AGTTAATAGCCCTGGGTATAGTAACAATGTTGAATATGAATCTTGATTGAAGATGAAATGGGTGATCCGCAGCCAAGTCC
                                                                P5c>    --P4c><P6
TAAGGACGTATAATGTCTATGGATGCAGTTCAACGACTAGATGGCAGTGGGTGTTGTTAAGACTTAGGTTTTTACAATGC
                          P6c>   <P7----  ----P3c><P8-----                ----P8c
TTAAGGTATAGTCTATTCTCTATC
>       --P7c>

details
=======
[P3/P3']: (70 , 77) - (261 , 268)
          score: 46 (46 + mod. len. 0)
          AA: 0 AC: 0 AG: 0 AT: 3 CC: 0 CG: 4 CT: 0 GG: 0 GT: 1 TT: 0
          WC: 7 mismatch: 0

[P4/P4']: (81 , 87) - (205 , 210)
          score: 36 (36 + mod. len. 0)
          AA: 0 AC: 0 AG: 1 AT: 1 CC: 0 CG: 4 CT: 0 GG: 0 GT: 1 TT: 0
          WC: 5 mismatch: 1
          misc: bulge P4 pos. 5

[P5/P5']: (92 , 93) - (199 , 200)
          score: 14 (14 + mod. len. 0)
          AA: 0 AC: 0 AG: 0 AT: 0 CC: 0 CG: 2 CT: 0 GG: 0 GT: 0 TT: 0
          WC: 2 mismatch: 0

[P6/P6']: (211 , 212) - (247 , 248)
          score: 29 (10 + mod. len. 19)
          AA: 0 AC: 0 AG: 0 AT: 0 CC: 0 CG: 1 CT: 0 GG: 0 GT: 1 TT: 0
          WC: 1 mismatch: 0
```

```
[P7/P7']: (252 , 258) - (308 , 313)
          score: 74 (74 + mod. len. 0)
          AA: 0 AC: 0 AG: 0 AT: 4 CC: 0 CG: 2 CT: 0 GG: 0 GT: 0 TT: 0
          WC: 6 mismatch: 0

[P8/P8']: (269 , 276) - (293 , 300)
          score: 102 (38 + mod. len. 64)
          AA: 0 AC: 0 AG: 0 AT: 3 CC: 0 CG: 2 CT: 0 GG: 0 GT: 3 TT: 0
          WC: 5 mismatch: 0


group details
=============
[P7/P7'] + [P8/P8']:
          total score: 176 (74 + 102)

[P4/P4'] + [P5/P5']
          total score: 50 (36 + 14)

[P4/P4'] + [P5/P5'] + [P6/P6']
          total score: 79 (36 + 14 + 29)

[P4/P4'] + [P5/P5'] + [P6/P6'] + [P7/P7'] + [P8/P8']
          total score: 255 (36 + 14 + 29 + 74 + 102)

-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Index

k-facteur, 15
2-intervalle, 169

adénosine, 209
adjacent, 11
ADN, 209
anti-codon, 210
APPLICATION INJECTIVE POLYSÉMIQUE, 205
APPLICATION INJECTIVE POLYSÉMIQUE SPEC INTERVALLES, 204
approximation
  algorithme d'_ absolue, 148
arête, 11
arbre, 15
  couvrant, 15
  enraciné, 15
arc, 12
ARRANGEMENT FAMILLE COUPLES SOUS-ENSEMBLES, 150
arrangement linéaire
  triangulaire, 74
ARRANGEMENT LINÉAIRE MAX-MAX, 135, 139
ARRANGEMENT LINÉAIRE MAX-MIN, 140
ARRANGEMENT LINÉAIRE MIN-MAX, 141
ARRANGEMENT LINÉAIRE À CUMUL CROISSANT, 108
ARRANGEMENT LINÉAIRE À CUMUL CROISSANT MAX, 109
ARRANGEMENT LINÉAIRE TRIANGULAIRE, 74, 75
automate
  à groupe, 39
  normalisé, 24
  standard, 27

base, 209
biclique, 95
boucle, 11

caterpillar, 174
chemin, 13
  élémentaire , 13
  hamiltonien, 13
chromatique
  index, 236
  nombre, 236
circuit, 13
  hamiltonien, 13
citron, 213
classe d'intersection, 172
clique, 15
CLIQUE, 79
cliques maximales d'un, 173
codage, 234
code génétique, 210
codon, 210
composante connexe, 14
couplage, 88
  fort, *voir* couplage induit
  induit, 88
  unique restreint, 89
COUPLAGE 3D, 158
$\{<, \Box\}$-COUPLAGE, 190
$\{\sqsubset, \Box\}$-COUPLAGE, 185
COUPLAGE UNIQUE RESTREINT ARÊTES, 95
COUVERTURE DES SOMMETS, 79, 134
couverture par cliques, 15
cytosine, 209

degré, 12
  sortant, 12
degré entrant, 12
diagramme de Hasse, 168

ensemble partiellement ordonné
  antichaîne, 168
  chaîne, 168
  hauteur, 168
ensemble stable, 15

$\{m-1,m\}$ EXACT 3SAT, 74
exon, 210
expression rationnelle, 23

famille de sous-ensembles, 60
forêt, 15

graphe, 11
  k-régulier, 15
  biparti, 14
    complet, 15
    exact-triangulaire, 90
    inf-triangulaire, 96
    sup-triangulaire, 96
  coloriage, 15
  complémentaire, 15
  complet, 15
  connexe, 14
  d'arêtes, 234
  d'inclusion, 189
  d'intersection, 171
  d'intervalles, 172
  d'intervalles propres, 172
  d'intervalles unitaires, 172
  décomposable, 237
  de t-intervalle, 174
  de comparaison, 168, 173
  de couverture, 168
  de croisement, 182
  de croisement de trapézoïdes circulaires, 186
  de séparation, 78
  de trapézoïdes circulaires, 182
  isomorphisme, 15
  ordre, 11
  orienté, 12
  orientation, 12
  parfait, 174
  séparation, 237
  simple, 11
  triangulé, 172
guanine, 209

hélice, 163

incident, 11
INF-SILHOUETTE, 121
INF-SILHOUETTE MONOTONE 121
intervalle, 168
  chevauchement, 169
  couple d'- emboîtés, 171
  extérieur, 171
  intérieur, 171
intervalles
  nombre d', 174
intron, 210
irréductible, 20
ISO INF-SILHOUETTE, 122
ISO MATRICE INF-Δ, 71
ISO SILHOUETTE, 123
ISO SOUS-MATRICE EXACT-$\mathbb{T}$, 100
ISO SOUS-MATRICE INF-$\mathbb{T}$, 99
isomorphisme
  famille de sous-ensembles, 60

matrice, 16
  complémentaire, 16
  d'adjacence, 19
  d'adjacence réduite, 20
  d'incidence, 60
  d'incidence sommets-arcs, 20
  d'intersection, 60
  de dépendance, 31
  de permutation, 17
  des cliques maximales, 173
  des occurrences, 61
  diagonale, 16
  enveloppe, 119
  forme standard, 129
  identité, 16
  inf-triangulaire, 18
  inverse, 18
  largeur de bande, 119
  silhouette, 120
    monotone, 121
  sup-triangulaire, 18
  surface, 129
  totalement unimodulaire, 21, 162
  transposée, 16
  triangulaire
    inférieure, 18
    supérieure, 18
  unimodulaire, 21

MATRICE INF-TRIANGULAIRE, 58
matrice:d'incidence sommets-arêtes, 21
matrice:symétrique, 16
PE d'incidence sommets-sommets, *voir* matrice d'adjacence
MIN-SURFACE, 131, 143
modèle de comparaison, 171
monotone, 147
MOTIF 2-INTERVALLES $\mathcal{R}$-COMPARABLES, 194
multigraphe, 11

nombre chromatique, 15
nucléotide, 209

orange, 217
orientation transitive, 173

PARTITION SOUS-MATRICES EXACT-$\mathbb{T}$, 157
PARTITION SOUS-MATRICES PRINCIPALES INF-$\mathbb{T}$, 155
PQ^T-ISO SILHOUETTE, 127
profil, 119
propriété de croissance, 62
  stricte, 62
pseudo-nœud, 166

RECOUVREMENT EXACT PAR 3-ENSEMBLES, *voir* x3c
repliement, 166
représentant, 193
  canonique, 193
rotation
  droite, 17
  gauche, 17

séquence
  complémentaire, 209
$\{m-1,m\}$ EXACT 3SAT, 74
semi-anneau, 28
  commutatif, 29
  idempotent, 29
sommet, 11
  feuille, 81
SOUS-MATRICE INF-TRIANGULAIRE, 86
SOUS-FAMILLE 2-INTERVALLES $\mathcal{R}$-COMPARABLES, 175
sous-graphe, 13
  couvrant, 13
  induit, 13
SOUS-GRAPHE INDUIT EXACT-TRIANGULAIRE, 90
SOUS-GRAPHE INDUIT INF-TRIANGULAIRE PONDÉRÉ, 97
sous-matrice
  principale, 99
SRD, *voir* système de représentants distincts
structure secondaire, 166
support, 171
  disjoint, 171
  unitaire, 171
système d'affectations, 30, 31
  disjoint, 31
système de représentants distincts, 61

t-repésentation, 174
thymine, 209
traduction, 210
transcription, 210
trapézoïde circulaire, 182, 186

x3c, 101

Bibliographie

[AAI86] T. Asano, T. Asano, and H. Imai, *Partitioning a polygonal region into trapezoids*, J. of the ACM **33** (1986), 290–312.

[ABMP91] H. Alt, N. Blum, K. Mehlhorn, and M. Paul, *Computing a maximum cardinality matching in a bipartite graph in time* $O(n^{1.5}\sqrt{m/\log n})$, Inform. Proc. Let. **37** (1991), 237–240.

[AC75] A.V. Aho and M.J. Corasick, *Efficient string matching: an aid to bibliographic search*, Comm. ACM **18** (1975), 333–340.

[ADH98] A.S. Asratian, T.M.J. Denley, and R. Häggkvist, *Bipartite graphs and their applications*, Cambridge University Press, Cambridge, 1998.

[Aho90] A.V. Aho, *Algorithms for finding patterns in strings*, Handbook of Theoret. Comp. Sc. (J. Van Leuwen, ed.), vol. A, Amsterdam, 1990, pp. 255–300.

[APSZ81] S.F. Assman, G.W. Peck, M.M. Syslo, and J. Zak, *The bandwidth of caterpillars with hair of length 1 and 2*, SIAM J. on Algebraic and Disc. Methods **2** (1981), 387–393.

[ASU86] A.V. Aho, R. Sethi, and J.D. Ullman, *Compilers, principles, techniques and tools*, Addison-Wesley, 1986.

[AYS84] H. Aoe, Y. Yamamoto, and R. Shimada, *A method for improving string pattern matching machines*, IEEE Trans. Softw. Engr. **10** (1984), no. 1, 116–120.

[BBC92] D. Beauquier, J. Berstel, and Ph. Chrétienne, *Éléments d'algorithmique*, Masson, 1992.

[BDLS00] A. Brandstädt, F.F. Dragan, V.B. Le, and T. Szymczak, *On stable cutsets in graphs*, Disc. Appl. Math. **105** (2000), 39–50.

[Ber75] C. Berge, *Perfect graphs*, Studies in graph theory (D.R. Fulkerson, ed.), Math. Assoc. Amer., Washington D.C., 1975, pp. 1–22.

[Ber76] ———, *Graphs and hypergraphs*, North-Holland Mathematical Library, 1976.

[BH92] R. Boppana and M.M. Halldórsson, *Approximating maximum independent sets by excluding subgraphs*, Bit **32** (1992), 180–196.

[BJ94] H.L. Bodlaender and K. Jansen, *On the complexity of the maximum cut problem*, Symposium on Theoretical Aspects of Computer Science, 1994, pp. 769–780.

[BKV96] B. Billoud, M. Kontic, and A. Viari, *Palingol: a declarative programming language to describe nucleic acids secondary structures and to scan sequence database*, Nucl. Acids Res. **24** (1996), 1395–403.

[BL76] K.S. Booth and G.S. Leuker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, Jourbal Comput. Syst. **13** (1976), 335–379.

[BM77] R.S. Boyer and J.S. Moore, *A fast string matching algorithm*, Comm. ACM **20** (1977), 762–772.

[Bol79] B. Bollobás, *Graph theory - an introductory course*, Springer-Verlag, New York, 1979.

[BR91] R.A. Brualdi and H.J. Ryser, *Combinatorial matrix theory*, Cambridge University Press, New York, 1991.

[Bru66] R.A. Brualdi, *Permanent of the direct product of matrices*, Pac. J. Math. **16** (1966), 471–482.

[BYG92] R. Baeza-Yates and G.H. Gonnet, *A new approch to text searching*, Comm. of the ACM **108** (1992), no. 2, 187–199.

[Cam89] K. Cameron, *Induced matchings*, Disc. Appl. Math. **24** (1989), 97–102.

[CCG+99] M. Crochemore, A. Czumaj, L. Gąsieniec, T. Lecroq, W. Plandowski, and W. Rytter, *Fast practical multi-pattern matching*, Information Processing Letters **71** (1999), 107–113.

[Cec87] T.R. Cech, *The chemistry of self-splicing RNA and RNA enzymes*, Science **236** (1987), 1532–1539.

[CH97] M. Crochemore and C. Hancart, *Automata for matching patterns*, Handbook of Formal Languages (G. Rosenberg and A. Salomaa, eds.), vol. 2, Springer-Verlag, 1997, pp. 399–462.

[Chv84] V. Chvátal, *Recoginzing decomposable graphs*, J. of Graph Theory **8** (1984), 51-53.

[CLR92] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press and McGraw Hill, Cambridge, 1992.

[CR94] M. Crochemore and W. Rytter, *Text algorithms*, Oxford University Press, 1994.

[CW79] B. Commentz-Walter, *A string matching algorithm fast on the average*, Proc. 6th Int. Coll. Automata, Lang. and Prog. (H.A. Maurer, ed.), 1979, pp. 118–132.

[DEKM98] R. Durbin, S.R. Eddy, A. Krogh, and G.J. Mitchison, *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, Cambridge University Press, Cambridge UK, 1998.

[DG94] S.H. Damberger and R.R. Gutell, *A comparative database of group I intron structures*, Nucl. Acids Res. **22** (1994), 3508–3510.

[Die00] R. Diestel, *Graph theory*, second ed., Graduate texts in Math., Springer-Verlag, 2000.

[DM41] B. Dushnik and W. Miller, *Partially ordered sets*, Amer. J. Math **63** (1941), 600–610.

[DS85] A. Danchin and P.P. Slonimski, *Sur la génétique et l'hérédité*, ch. Les gènes en morceaux, pp. 77–104, Société d'éditions scientifiques, 1985.

[ED94] S.R. Eddy and R. Durbin, *RNA sequence analysis using covariance models*, Nucl. Acids Res. **22** (1994), 2079–2088.

[Edm65] J. Edmonds, *Paths, trees and flowers*, Canad. J. Math. **17** (1965), 449–467.

[EM96] N. El-Mabrouk, *Recherche approchée de motifs Application à des structures biologiques structurées*, Ph.D. thesis, Université de Marne-la-Vallée, Dec 1996.

[EML96] N. El-Mabrouk and F. Lisacek, *Very fast identification of rna motifs in genomic dna. application to tRNA search in the yeast genome*, J. Mol. Biol. **264** (1996), 46–55.

[EPL72] S. Even, A. Pnueli, and A. Lempel, *Permutation graphs and transitive graphs*, J. Assoc. Comput. Mach. **19** (1972), 400–410.

[ES73] C.J. Everett and P.R. Stein, *The asymptotic number of* (0,1)*-matrices with zero permanent*, Disc. Math. **6** (1973), 29–34.

[FF62] L.R. Ford and D.R. Fulkerson, *Flows in network*, Princeton University Press, Princeton, 1962.

[FMW97] S. Felsner, R. Müller, and L. Wernisch, *Trapezoid graphs and generalizations, geometry and algorithms*, Disc. Appl. Math. **21** (1997), 13–32.

[FPR99] P. Festa, P.M. Pardalos, and M.G.C. Resende, *Feedback set problems*, Handbook of combinatorial optimization (DingZhu Du and Panos M. Pardalos, eds.), vol. A, Kluwer Academic Publishers, 1999, pp. 209–258.

[FS93] J.J. Fan and K.Y. Su, *An efficient algorithm for matching multiple patterns*, IEEE, Trans. Knowledge and Data Eng. **5** (1993), 339–351.

[Gav74a] F. Gavril, *Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent sets of a chordal graph*, SIAM, J. on Algebraic and Disc. Methods **1** (1974), 180–187.

[Gav74b] ———, *The intersection graphs of subtrees are exactly the chordal graphs*, J. Combin. Theory Ser. **B 16** (1974), 47–56.

[GGJK78] M.R. Garey, R.L. Graham, D.S. Johnson, and D.E. Knuth, *Complexity results for bandwidth minimization*, SIAM J. Appl. Math. **34** (1978), 477–495.

[GH64] P.C. Gilmore and A.J. Hoffman, *A characterisation of comparability graphs and of interval graphs*, Canad. J. Math. **16** (1964), 539–548.

[GHL] M.C. Golumbic, T. Hirst, and M. Lewenstein, *Uniquely restricted matchings*, Algorithmica, to appear.

[GJ76] M.R. Garey and D.S. Johnson, *The complexity of near-optimal graph coloring*, J. ACM **23** (1976), 43–49.

[GJ79] ———, *Computers and intractability: a guide to the theory of NP-completness*, Freeman, San Fransisco, 1979.

[GJS76] M.R. Garey, D.S. Johnson, and L. Stockmeyer, *Some simplified NP-complete graph problems*, Theoret. Comp. Sc. **1** (1976), 237–267.

[GJT76] M.R. Garey, D.S. Johnson, and R.E. Tarjan, *The planar hamiltonian circuit problem is np-complete*, SIAM J. Comput. **5** (1976), 704–714.

[GL93] M.C. Golumbic and R.C. Laskar, *Irredundancy in circular arc graphs*, Disc. Appl. Math. **44** (1993), 79–89.

[GLL82] U.I. Gupta, D.T. Lee, and J.Y-T. Leung, *Efficient algorithms for interval graph and circular-arc graphs*, Networks **12** (1982), 459–467.

[GM79] M. Gondran and M. Minoux, *Graphes et algorithmes*, Eyrolles, Paris, 1979.

[Gol80] M.C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press, New York, 1980.

[Gus97] D. Gusfield, *Algorithms on strings, tress ans sequences: Computer science and computatiobal biology*, Cambridge University Press, New York, 1997.

[Had75] F.O. Hadlock, *Finding a maximum cut of a planar graph in polynomial time*, SIAM J. Comput. **4** (1975), 221–225.

[Har71] F. Harary, *Graph theory*, Addison-Wesley, New York, 1971.

[HK73] J.E. Hopcroft and R.M. Karp, *An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs*, SIAM J. Comput. **2** (1973), no. 4, 225–231.

[HMPV00] M. Habib, R.M. McConnell, C. Paul, and L. Viennot, *Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing*, Theoret. Comp. Sc. **234** (2000), 59–84.

[Hoy81] I. Hoyler, *The NP-completeness of edge-coloring*, SIAM J. Comput. **10** (1981), no. 4, 718–720.

[HPV99] M. Habib, C. Paul, and L. Viennot, *Partition refinement techniques : an interesting algorithmic tool kit*, IJFCS: International J. of Foundations of Computer Science **10** (1999).

[HSS98] I.L. Hofacker, P. Schuster, and P.F. Stadler, *Combinatorics of RNA secondary structures*, Disc. Appl. Math. **88** (1998), 207–237.

[Hsu92] W.L. Hsu, *A simple test for the consecutive ones property*, Lectures Notes in Computer Science **650** (1992), 459–468.

[Jun99] D. Jungnickel, *Graphs, networks and algorithms*, Springer Verlag, Berlin. 1999.

[Kar72] R.M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (J.W. Thatcher and R.E Miller, eds.), Plenum Press, New York, 1972, pp. 85–103.

[KH78] D.G. Kirkpatrick and P. Hell, *On the complexity of a genelarized matching problem*, Proc. 10th Ann. ACM Cymp. on Theory of Computing (New York), ACM, 1978, pp. 240–245.

[Kil92] P. Kilpeläinen, *Tree matching problems with applications to structured text databases*, Ph.D. thesis, University of Helsinki, Finland, 1992.

[KMNF92] T. Kashiwabara, S. Masuda, K. Nakajima, and T. Fujisawa, *Generation of maximum independent sets of a bipartite graph and maximum cliques of a circular-arc graph*, J. of Algorithms **13** (1992), 161–174.

[KMP77] D.E. Knuth, J.H. Morris, and V. Pratt, *Fast pattern matching in strings*, SIAM J. Comput. **6** (1977), 323–350.

[KMR72] R.M. Karp, R.E. Miller, and A.L. Rosenberg, *Rapid identification of repeated patterns in string, trees and array*, Proc. 4th Annu. ACM Symp. Theory of Computing, 1972, pp. 125–136.

[LDM94] F. Lisacek, Y. Diaz, and F. Michel, *Automatic identification of group I intron cores in genomic DNA sequences*, J. Mol. Biol. **235** (1994), 1206–1217.

[Lef97] F. Lefevbre, *Grammaire S-attribuées muti-bandes et applications à l'analyse automatiques de séquences biologiques*, Juin 1997.

[Leu84] J.Y-T. Leung, *Fast algorithms for generating all maximal independent sets of interval, circular-arc and chordal graphs*, J. of the Algorithms **5** (1984), 22–34.

[Lew96] M. Lewenstein, *Matchings*, 1996.

[Lis95] F. Lisacek, *Un agent rationnel pour la reconnaissance de molécules d'ARN*, Technique et science informatiques **14** (1995), 161–178.

[LL93] Y. Liu and M.J. Leibowitz, *Variation and in vitro splicing of group I intron in rRNA genes of Pneumocystis carinii*, Nucl. Acids Res. **21** (1993), 2415–2421.

[LP00] R. Lyngs and C. Pedersen, *Pseudoknots in RNA secondary structures*, RECOMB00: Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, Tokyo, Japan, 2000.

[LS93] D.T. Lee and M. Sarrafzadeh, *Maximum indenpendent set of a permutation graph with k tracks*, Internat. J. Comput. Geom. Appl. **3** (1993), no. 3, 291–304.

[LY93] Y. Lin and J. Yuan, *Profile minimization problems for matrices and graphs*, Tech. report, Dept. of Math., Zhengzhou University, Zhengzhou, Henan 450052, P.R. China, 1993.

[Mar45] E. Marczewski, *Sur deux propriétés des classes d'ensembles*, Fund. Math. **33** (1945), 303–307.

[MM99] T.A. McKee and F.R. McMorris, *Topics in intersection graph theory*, SIAM monographs on discrete mathmatics and applications, 1999.

[Moh97] M. Mohri, *String-matching with automata*, Nordic J. of Computing **4** (1997), no. 2, 217–231.

[Mon86] B. Monien, *The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete*, SIAM J. on Algebraic and Disc. Methods **7** (1986), no. 4, 505–512.

[Mos89] A.M. Moshi, *Matching cutsets in graphs*, J. of Graph Theory **13** (1989), 527–536.

[MS83] F.R. McMorris and D.R. Shier, *Representing chordal graphs on $k_{1,n}$*, Comment. Math. Univ. Carolin. **24** (1983), 489–194.

[MV80] S. Micali and V.V. Vazirani, *An $o(\sqrt{|V|}\cdot|e|)$ algorithm for finding maximum matching in general graphs*, Proc. Twenty-first Annual Symposium on the Foundations of Computer Science, 1980, pp. 17–27.

[MW90] F. Michel and E. Westhof, *Modelling of the three-dimensional architecture of group I catalytic introns based on comparative sequence analysis*, J. Mol. Biol. **216** (1990), 585–610.

[NJ80] R. Nussinov and A.B. Jacobson, *Fast algorithm for predicting the secondary structure of singled-stranded RNA*, Proceedings of the National Academy of Science, vol. 77, 1980, pp. 6309–6313.

[NPGK78] R. Nussinov, G. Pieczenik, J.R. Griggs, and D.J. Kleitman, *Algorithms for loop matching*, SIAM J. of Appl. Math. **35** (1978), 35–68.

[Pap76] C.H. Papadimitriou, *The NP-completness of the bandwidth minimization problem*, Computing **16** (1976), 263–270.

[Pap94] ———, *Computational complexity*, Addison-Wesley, New York, 1994.

[Pee00] R. Peeters, *The maximum edge biclique problem is NP-complete*, Tech. report, Tilburg University. Faculty of Economics and Business Administration, 2000.

[Pol74] S. Poljak, *A note on stable sets and coloring of graphs*, Comment. Math. Univ. Carolin. **15** (1974), 307–309.

[PS82] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Dover Publications, Inc, New York, 1982.

[SBH+94] Y. Sakakibara, M. Brown, R. Hughey, I.S. Mian, and K. Sjölander, *Stochastic context-free grammars for tRNA*, Nucl. Acids Res. **22** (1994), 5112–5120.

[Sch78] T.J. Schaefer, *The complexity of satisfiability problem*, Proc. 10th Ann. ACM Symp. on Theory of Computing (New York), Association for computing Machinery, 1978, pp. 216–226.

[Sch85] E.R. Scheinerman, *Characterizing intersection classes of graphs*, Disc. Math. **55** (1985), 185–193.

[Sea92] D.B. Searls, *The linguistics of DNA*, American Scientist **20** (1992), 579–591.

[Sed77] R. Sedgewick, *Permutation generation methods*, Computing Surveys **9** (1977), 137–164.

[Sey95] P.D. Seymour, *Packing directed circuits fractionally*, Combinatorica **15** (1995), 281–288.

[Sta84] R. Staden, *Computer methods to locate signals in nucleic acid sequence*, Nucl. Acids Res. **12** (1984), 507–521.

[SV82] L. Stockmeyer and V.V. Vazirani, *NP-completeness of some generalizations of the maximum matching problem*, Information Processing Letters **15** (1982), no. 1, 14–19.

[Tan96] P.J. Tanenbaum, *Simultaneous representation of interval and interval-containment orders*, Order **13** (1996), 339–350.

[tDPD92] E. ten Dam, K. Pleij, and D. Draper, *Structural and functional aspects of rna pseudoknots*, Biochemistry **31** (1992), 11665–11676.

[TH79] W.T. Trotter and F. Harary, *On double and multiple interval graphs*. J. Graph Theory **3** (1979), 205–211.

[Tho68] K. Thompson, *Regular expression search algorithm*, CACM **11** (1968), 419–422.

[Ung88] W. Unger, *On the k-coloring of circle graphs*, STACS 88: Fifth annual symposium on the theoretical aspect of computer science, Bordeaux, France (R. Cori and M. Wirsing, eds.), Lectures Notes in Computer Science, vol. 294, Springer-Verlag, 1988, pp. 61–72.

[Val79] L.G. Valiant, *The complexity of computing the permanent*, Theoret. Comp. Sc. **8** (1979), 189–201.

[Viz64] V.G. Vizing, *Über eine Abschätzung der chromatischen Klasse eines p-Graphen (in Russian)*, Diskret. Analiz. **3** (1964), 25–30.

[VR96] G. Venkatesan and C. Pandu Rangan, *Approximate triclique coloring for register allocation*, Tech. report, Department of Computer Science, Indian Institute of Technology, Madras, 1996.

[Wat78] M.S. Waterman, *Secondary structure of single-stranded nucleic acids*, Adv. Math. Suppl. Studies **1** (1978), 167–212.

[WM94] S. Wu and U. Manber, *A fast algorithm for multi-pattern searching*, Tech. Report TR 94-17, University of Arizona, Tucson, 1994.

[WS78] M.S. Waterman and T.F. Smith, *RNA secondary structure: a complete mathmatical analysis*, Math. Biosc. **42** (1978), 257–266.

[WS84] D.B. West and D.B. Shmoys, *Recognizing graphs with fixed interval number is NP-complete*, Disc. Appl. Math. **8** (1984), 295–305.

[ZS81] M. Zuker and P. Stiegler, *Optimal folding of large RNA sequences using thermodynamics and auxiliary*, Nucl. Acids Res. **9** (1981), 133–148.

[ZS84] M. Zuker and D. Sankoff, *RNA secondary structure and their prédiction*, Bulletin of Mathematical Biology **46** (1984), 591–621.