LIF

Laboratoire d'Informatique Fondamentale
de Marseille

Unité Mixte de Recherche 6166
CNRS – Université de Provence – Université de la Méditerranée

Ordonnancement avec exclusion mutuelle par un graphe d'intervalles ou d'une classe apparentée : complexité et algorithmes

Frédéric Gardi

Rapport/Report 25-2005

14 Juin 2005, révisé 13 Déc 2005

Ordonnancement avec exclusion mutuelle par un graphe d'intervalles ou d'une classe apparentée : complexité et algorithmes

Frédéric Gardi

LIF – Laboratoire d'Informatique Fondamentale de Marseille
UMR 6166
CNRS – Université de Provence – Université de la Méditerranée

Parc Scientifique et Technologique de Luminy,
case 901, 163 avenue de Luminy
13288 Marseille cedex 9, France.

Frederic.Gardi@lif.univ-mrs.fr

Abstract/Résumé

The *mutual exclusion scheduling problem* can be formulated as follows in graph-theoretic terms : given a graph G and an integer k, determine a minimum coloring of G such that each color is used at most k times. When the graph G is an *interval graph*, this problem has some applications in *workforce planning*. Then, the subject of this thesis is to detail the complexity of the mutual exclusion scheduling problem for interval graphs and related classes.

Keywords : mutual exclusion scheduling, graph coloring, workforce planning, interval graphs, classes of graphs.

Le *problème d'ordonnancement avec exclusion mutuelle* peut être formulé comme suit en les termes de la théorie des graphes : étant donné un graphe G et un entier k, déterminer une coloration minimum de G tel que chaque couleur apparaisse au plus k fois. Lorsque le graphe G est un *graphe d'intervalles*, ce problème a des applications dans le domaine de la *planification de personnel*. Ainsi, cette thèse a pour objet l'étude détaillée de la complexité du problème d'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles ou de classe apparentée.

Mots-clefs : ordonnancement avec exclusion mutuelle, coloration de graphes, planification de personnel, graphes d'intervalles, classes de graphes.

Relecteurs/Reviewers :

Alain Colmerauer (LIF, Marseille)
Gérard Cornuéjols (LIF, Marseille)
Dominique De Werra (EPFL, Lausanne)
Frédéric Maffray (IMAG, Grenoble)
Jean-François Maurras (LIF, Marseille)
Michel Van Caneghem (LIF, Marseille)

Ce rapport de recherche contient l'intégralité du texte de la thèse de Doctorat soutenu par l'auteur le 14 Juin 2005 devant le jury composé des personnes précédemment citées, à la Faculté des Sciences de Luminy, Université de la Méditerranée – Aix-Marseille II.

LABOR OMNIA VINCIT IMPROBUS[1]

À la mémoire de Kléber Gardi, mon grand-père.

Là où il y a une volonté, il y a un chemin.

Gaston Rébuffat (1921–1985),
alpiniste français, né à Marseille.

[1] *Un travail opiniâtre vient à bout de tout.* Virgile (*Géorgiques*, Livre I, vers 145–146).

Remerciements

En tout premier lieu, je tiens à remercier Monsieur Dominique De Werra, Professeur de Recherche Opérationnelle à l'École Polytechnique Fédérale de Lausanne, et Monsieur Frédéric Maffray, Chargé de Recherche au Laboratoire Leibniz de l'Institut d'Informatique et Mathématiques Appliquées de Grenoble, qui m'ont fait l'honneur d'être les rapporteurs de cette thèse. Les remarques dont tous deux m'ont fait part lors de la rédaction du mémoire ont permis d'en améliorer largement sa qualité. Ensuite, je remercie chaleureusement Messieurs Alain Colmerauer, Gérard Cornuéjols et Jean-François Maurras, Professeurs d'Informatique à l'Université de la Méditerranée – Aix-Marseille II, pour avoir accepté d'assister en qualité d'examinateurs à la soutenance de cette thèse.

J'ai rencontré Michel Van Caneghem, Professeur d'Informatique à l'Université de la Méditerranée – Aix-Marseille II, alors qu'il enseignait en Maîtrise d'Informatique. Depuis, nous n'avons cessé de collaborer, notamment à l'étude de problèmes de planification de personnel (mémoire de Maîtrise, mémoire de DEA, aujourd'hui thèse de Doctorat), mais aussi à la réalisation de logiciels traitant de ces différentes problématiques (logiciel Bamboo, édité par Prologia – Groupe Air Liquide). J'ai même eu le plaisir de le côtoyer en tant que chargé de travaux dirigés et pratiques dans le cadre du cours qu'il dispense avec Alain Colmerauer, joliment intitulé “Turing : des codes secrets aux machines universelles”. Michel Van Caneghem m'a toujours accordé sa confiance et m'a laissé une grande liberté dans la poursuite de mes recherches. Pour tout cela, je l'en remercie encore.

Pourtant, cette thèse aurait pu ne jamais voir le jour du fait de quelques “dysfonctionnements administratifs” sur lesquels je ne m'attarderai pas ici, tant ils sont caractéristiques de la faiblesse de nos institutions. En effet, alors que je m'apprêtais à débuter ma thèse, je me voyais priver d'un financement du Ministère de la Recherche que nous croyions pourtant acquis au vu de mes résultats de DEA. Malgré cela, Michel Van Caneghem a su me convaincre de poursuivre mes travaux de recherche au sein de l'Équipe Combinatoire et Recherche Opérationnelle du Laboratoire d'Informatique Fondamentale de Marseille, tout en étant salarié de la firme Prologia du Groupe Air Liquide. En dépit des nombreux sacrifices que cela a exigés, je ne le regrette pas.

J'adresse donc mes remerciements les plus sincères à la société Prologia – Groupe Air Liquide, qui m'a non seulement permis de participer à des projets d'envergure au sein desquels j'ai pu mettre en oeuvre mes compétences en optimisation combinatoire, mais m'a aussi laissé poursuivre librement mes travaux de recherche au sein du Laboratoire d'Informatique Fondamentale. Merci à Madame Anne-Françoise Douix, Directeur Général, Monsieur Alain David, Directeur du Département Systèmes à Base de Connaissance, Monsieur Stéphane

N'Dong, Directeur du Département Planification, Optimisation et Contraintes et Monsieur Fabrice Verdier, Directeur Commercial, pour la confiance qu'ils m'ont accordée et m'accordent encore. Je tiens à saluer l'ensemble du personnel de PROLOGIA pour son soutien et sa gentillesse, ainsi que l'équipe du Pôle Optimisation de la Division Services d'Air Liquide pour son accueil, lors de mon séjour de trois mois à Boulogne-Billancourt (c'est durant ce séjour que j'ai rédigé mes deux premiers articles !).

Je me dois aussi de remercier Monsieur David Trotman, Professeur de Mathématiques à l'Université de Provence – Aix-Marseille I et ancien Directeur de l'École Doctorale de Mathématiques et Informatique de Marseille, pour les financements qu'il m'a accordés dans le cadre de ma participation à différents colloques.

Bien entendu, je salue tous les membres du Laboratoire d'Informatique Fondamentale et du Département d'Informatique de la Faculté des Sciences de Luminy que j'ai côtoyés ces dernières années. Il me tient à cœur de remercier quelques unes de ces personnes en particulier. Un grand merci à Mesdames les Secrétaires pour leur gentillesse et leur dévouement : Sylvie Calabrèse, Isabelle Raye, Brigitte Garabédian et Muriel Quaglia. Merci à Victor Chepoi, Gérard Cornuéjols, Jean-François Maurras et Yann Vaxès pour toutes les réponses qu'ils ont apportées à mes questions ; c'est aussi grâce à eux, si aujourd'hui je comprends un peu mieux ce qu'est l'optimisation combinatoire. Merci encore à Victor Chepoi pour avoir pris le temps de relire certains de mes articles et pour ses conseils avisés en matière de rédaction. Merci à Alain Colmerauer pour ses encouragements constants. Merci à Paul Sabatier pour les multiples articles scientifiques qu'il a commandés à ma demande et dont certains furent décisifs dans l'avancée de mes recherches. Merci à Édouard Thiel pour m'avoir invité à un des séminaires du laboratoire, ainsi que pour sa gestion irréprochable des rapports de recherche du laboratoire. Merci à Henri Garetta pour répondre avec toujours autant de sympathie à nos inlassables questions concernant ce bon vieux langage C. Merci à Jean-Luc Massat et Claude Sabatier pour leur aide précieuse lorsque surgirent les petits ennuis "système" ou "réseau". Merci à Jacques Guizol pour sa bonne humeur et sa gentillesse (Jacques, vous pouvez toujours compter sur nous pour accueillir vos stagiaires bulgares). Enfin, un grand merci à tous les étudiants avec qui j'ai partagé ce fameux bureau du 6ème étage et avec qui j'ai eu de nombreuses et riches discussions (bien que pas toujours scientifiques) : Olivier Anglada (qui m'a accompagné dans la bonne humeur tout au long de ces trois années), Régis Barbanchon (qui est Maître de Conférence, mais que j'adore taquiner ; Régis, encore merci pour tes conseils en matière de typographie), Khalil Djelloul (bien qu'il se soit longtemps caché dans le bureau d'Alain Colmerauer), Bertrand Estellon (dernier arrivant, mais non des moindres ; avec qui les débats sur le métier sont toujours animés), Mahdi Malik (d'une gentillesse et d'une attention rare), Roumen Nedev (pour qui la combinatoire est une seconde peau ; Roumen, dessine moi un polyèdre !), Karim Nouioua (sans doute "le meilleur d'entre nous" et je ne plaisante pas).

Je me dois d'ailleurs de m'attarder quelque peu sur ce dernier : Karim Nouioua. Je l'ai rencontré alors que je terminais ma première année de thèse. Plus qu'un collègue de travail, c'est devenu un ami. Un ami avec qui j'ai passé tant de soirées à discuter "optimisation" dans notre bureau du 6ème (« $\mathcal{P} = \mathcal{NP}$? »). Un ami qui m'a poussé il y a maintenant plus d'un an à participer avec lui au Challenge ROADEF'2005, en sus de nos recherches respectives. « Pour voir si nous sommes capables ! », me disait-il. Ce Challenge, nous l'avons gagné avec Karim, mais aussi Bertrand, dont l'arrivée en cours de route fut sans doute décisive. Un article est maintenant en cours d'écriture. Cela restera un de mes plus beaux souvenirs de ces trois années de Doctorat. De grosses rigolades. De rudes soirées (voire nuits) passées à programmer. En définitive, de sacrés bons moments. Merci Bertrand, merci Karim.

Un grand merci à Bruno et Marianne Amenta partis depuis peu "faire fortune" aux Amériques pour leur indéfectible amitié. Que de fous rires inoubliables. Vous me manquez déjà. Un salut à mes fidèles amis goultois, ainsi qu'à mes amis sportifs du BCI Athlétisme, que j'ai dû quelque peu délaisser ces dernières années pour mener à bien cette entreprise.

Enfin, je remercie bien sûr ceux qui ont fait de moi ce que je suis, ceux grâce à qui tant d'années d'études ont été possibles, ceux envers qui j'ai une dette imprescriptible : mes parents Max et Michèle. Un grand merci à ma grand-mère Odette qui me supporte depuis bien des années et à ma tante Christiane qui m'a toujours encouragé. Pépé et Mémé Bonnet, je ne vous oublie pas. Bravo à Romain, mon jeune frère, pour cette première année réussie en Histoire, loin de toute Mathématique. Une grosse bise à Sabine – "mon Panda" – qui m'a accompagné tout au long de cette thèse et chez qui une grande partie a été écrite (notamment la preuve de l'assertion (7) du lemme 4.7 que je conjecturais depuis près de deux ans, merci génial Panda !). Merci à Madame Agnès De Broche, sa maman, pour sa relecture patiente et son aide précieuse en matière d'orthographe et de grammaire.

Toutes les pages que j'ai pu noircir ces dernières années ne m'ont sûrement pas apporté autant que ceux auprès de qui je les ai noircies. Merci à tous ceux qui étaient là. Une pensée pour ceux qui ne sont plus. Merci à tous ceux qui ont cru en moi.

À Marseille, le 25 Mai 2005

F. G.

Résumé

Le *problème d'ordonnancement avec exclusion mutuelle* peut être formulé comme suit en les termes de la théorie des graphes : étant donné un graphe G non orienté et un entier k, déterminer une coloration minimum de G tel que chaque couleur apparaisse au plus k fois. Lorsque le graphe G est un *graphe d'intervalles*, ce problème a des applications dans le domaine de la *planification de personnel*. Ainsi, cette thèse a pour objet l'étude détaillée de la complexité du problème d'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles ou de classe apparentée.

Les deux premiers chapitres de la thèse traitent de la complexité du problème pour les graphes d'intervalles, ainsi que deux extensions, les graphes d'arcs circulaires et les graphes de tolérances. Lorsque le problème s'avère être polynomial, nous proposons des algorithmes à la fois simples et efficaces pour le résoudre (notamment des algorithmes en temps et espace linéaire).

Motivé par des aspects pratiques, nous analysons ensuite l'impact de la propriété suivante sur la complexité du problème : le graphe G admet une coloration où chaque couleur apparaît au moins k fois. Nous montrons que si un graphe d'intervalles satisfait cette condition, alors celui-ci admet une partition optimale en $\lceil n/k \rceil$ stables de taille au plus k qui peut être calculée en temps et espace linéaire. Cette assertion est ensuite étendue aux graphes sans $K_{1,3}$, aux graphes d'arcs circulaires, ainsi qu'aux graphes triangulés pour $k \leq 4$.

Le dernier chapitre est consacré à certains problèmes de partition relatifs aux graphes d'intervalles. Nous y étudions en particulier le problème de la partition d'un graphe d'intervalles ou d'arcs en sous-graphes d'intervalles propres, pour lequel nous établissons la proposition suivante : tout graphe d'intervalles admet une partition en moins de $\lceil \log_3 n \rceil$ sous-graphes d'intervalles propres et celle-ci peut être calculée en temps $O(n \log n + m)$ et espace linéaire. Ce résultat est mis à profit lors de la conception d'algorithmes polynomiaux d'approximation pour un problème de planification de personnel lié à l'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles ou d'arcs.

Mots-clefs : ordonnancement avec exclusion mutuelle, coloration de graphes, planification de personnel, graphes d'intervalles, classes de graphes.

Abstract

The *mutual exclusion scheduling problem* can be formulated as follows in graph-theoretic terms : given an undirected graph G and an integer k, determine a minimum coloring of G such that each color is used at most k times. When the graph G is an *interval graph*, this problem has some applications in *workforce planning*. Then, the subject of this thesis is to detail the complexity of the mutual exclusion scheduling problem for interval graphs and related classes.

The first two chapters of the thesis deal with the complexity of the problem for interval graphs, as well as two extensions, circular-arc graphs and tolerance graphs. When the problem is proved to be polynomial, we propose some simple and efficient algorithms for its resolution (in particular linear time and space algorithms).

Motivated by practical aspects, we also analyse the impact of the following property on the complexity of the problem : the graph G admits a coloring such that each color appears at least k times. We show that if an interval graph satisfies this property, then it admits an optimal partition into $\lceil n/k \rceil$ stable sets of size at most k which can be computed in linear time and space. Then, this assertion is extended to $K_{1,3}$-free graphs, circular-arc graphs, as well as chordal graphs for $k \leq 4$.

The last chapter is devoted to partition problems related to interval graphs. We investigate particularly the problem of partitioning interval ou circular-arc graphs into proper interval subgraphs, for which the following proposition is established : any interval graph admits a partition into less than $\lceil \log_3 n \rceil$ proper interval graphs and this one can be computed in $O(n \log n + m)$ time and linear space. This result is used in the design of polynomial approximation algorithms for a workforce planning problem related to mutual exclusion scheduling for interval or circular-arc graphs.

Title : Mutual exclusion scheduling with interval graphs or related classes : complexity and algorithms.

Keywords : mutual exclusion scheduling, graph coloring, workforce planning, interval graphs, classes of graphs.

Table des matières

Table des figures

« In an address delivered on 9 September 1949 at a meeting of the American Psychological Association at Denver, Colorado, Thorndike [1950] studied the problem of the 'classification' of personnel :

[...] There are, as has been indicated, a finite number of permutations in the assignment of men to jobs. When the classification problem as formulated above was presented to a mathematician, he pointed to this fact and said that from the point of view of the mathematician there was no problem. Since the number of permutations was finite, one had only to try them all and choose the best. He dismissed the problem at that point. This is rather cold comfort to the psychologist, however, when one considers that only ten men and ten jobs mean over three and a half million permutations. Trying out all the permutations may be a mathematical solution to the problem, it is not a practical solution. » [2]

Alexander Schrijver,
Combinatorial Optimization : Polyhedra and Efficiency,
Volume A, pages 295–296. (Historical notes)

[2] *« Dans une communication livrée le 9 Septembre 1949 à un colloque de l'Association Américaine de Psychologie à Denver, Colorado, Thorndike [1950] étudiait le problème de la 'classification' du personnel :*

[...] Il y a, comme il l'a été indiqué, un nombre fini de permutations dans l'affectation des hommes aux tâches. Quand le problème de la classification comme formulé ci-dessus fut présenté à un mathématicien, il souligna ce fait et dit que du point de vue du mathématicien il n'y avait pas de problème. Puisque le nombre de permutations est fini, il n'y a qu'à toutes les essayer et choisir la meilleure. Il laissa le problème en cet état. Cela n'est guère réconfortant pour le psychologue, cependant, quand on sait que seulement dix hommes et dix tâches entraînent plus de trois millions et demi de permutations. Essayer toutes les permutations peut être une solution mathématique au problème, ce n'est pas une solution pratique. »

Chapitre 1

Introduction

Cette thèse est dédiée à l'étude d'un *problème d'ordonnancement où la relation d'exclusion mutuelle entre les tâches est représentée par un graphe d'intervalles ou d'une classe apparentée*. Après avoir introduit le problème et sa formulation en les termes de la théorie des graphes, nous donnerons, concernant les différentes classes de graphes étudiées, toutes les définitions nécessaires à la bonne compréhension du mémoire. Enfin, nous ferons un état de l'art du problème à l'issue duquel nous annoncerons le plan ainsi que les principaux résultats de la thèse.

Pour des raisons évidentes, nous considérerons le lecteur comme étant initié aux notions les plus basiques de l'algorithmique et de la théorie des graphes. Si toutefois ce n'était pas le cas, nous invitons celui-ci à consulter les ouvrages de Cormen *et al.* [28] et de Berge [9] (en français), ou encore l'excellent livre de Golumbic [75] dont le sujet se situe précisément à la croisée de ces deux domaines.

1.1 Présentation du problème

Voici un problème fondamental en théorie de l'ordonnancement : n tâches doivent être exécutées sur k processeurs en le minimum de temps, avec la contrainte que certaines ne peuvent pas être exécutées en même temps parce qu'elles partagent les mêmes ressources. En raison de leurs applications industrielles importantes, de nombreuses variantes de ce problème ont déjà fait l'objet d'études approfondies ; la très vaste littérature consacrée aux domaines de l'optimisation combinatoire et de la recherche opérationnelle abonde de références à leurs sujets (le lecteur intéressé est renvoyé à l'article de Krarup et De Werra [106] ainsi qu'au récent recueil de Blazewicz *et al.* [12]).

Lorsque toutes les tâches ont le même temps d'exécution, le problème en question peut être élégamment formulé en les termes de la théorie des graphes. En effet, un moyen des plus naturels pour représenter l'exclusion mutuelle entre les tâches est un graphe non orienté, où chaque sommet correspond à une tâche et deux sommets sont reliés par une arête si les tâches correspondantes sont en conflit. Un ordonnancement optimal des n tâches sur les k processeurs correspond alors exactement à une *coloration des sommets du graphe à l'aide du plus petit nombre de couleurs possible tel que chaque couleur n'apparaisse pas plus de k*

fois (voir la figure 1.1). Nous rappelons que colorier un graphe non orienté consiste en le marquage de chacun de ses sommets par une couleur de façon à ce que deux sommets reliés par une arête aient des couleurs différentes ; lorsque le nombre de couleurs utilisées est le plus petit possible, nous obtenons alors une *coloration minimum* du graphe.

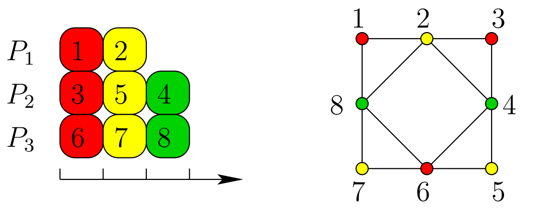


Fig. 1.1 – ordonnancement *vs* coloration.

De ce fait, Baker et Coffman [6] ont baptisé *problème d'ordonnancement avec exclusion mutuelle* (de l'anglais *mutual exclusion scheduling problem*) le problème d'optimisation combinatoire suivant :

Ordonnancement avec Exclusion Mutuelle
Entrée : un graphe $G = (V, E)$, un entier positif k ;
Sortie : une coloration minimum de G où chaque couleur apparaît au plus k fois.

Par commodité, nous utiliserons l'abréviation Ordo pour nommer ce problème. Lorsque k est un paramètre fixé (*i.e.* une constante du problème), nous parlerons du problème k-Ordo.

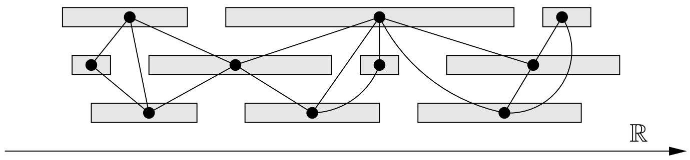


Fig. 1.2 – Un graphe d'intervalles et sa représentation.

Colorier un graphe avec le minimum de couleurs est un problème $\mathcal{NP}$-difficile des plus célèbres [104]. Par conséquent, le problème d'ordonnancement avec exclusion mutuelle est lui aussi $\mathcal{NP}$-difficile et la recherche d'un algorithme polynomial résolvant le problème dans le cas le plus général semble compromise. Toutefois, si l'on se restreint à des graphes pour lesquels une coloration minimum peut être obtenue en temps et espace polynomial (comme par exemple les graphes parfaits), alors le problème Ordo n'est plus nécessairement $\mathcal{NP}$-difficile. Parmi ces graphes, les *graphes d'intervalles* se distinguent par leur champ d'application aussi large que varié : génétique, ordonnancement, psychologie, archéologie, etc (pour une revue complète de ces applications, nous renvoyons le lecteur aux ouvrages de Roberts [132, 133], Golumbic [75] et Fishburn [54]). Un graphe d'intervalles est le graphe des intersections d'un ensemble d'intervalles de l'axe des réels, c'est-à-dire un graphe dont chaque sommet peut être mis en correspondance avec un intervalle de façon à ce que deux sommets reliés par une arête soient représentés par des intervalles qui se chevauchent (voir la figure 1.2).

Voici une application se rapportant aux graphes d'intervalles que nous avons rencontrée lors de notre séjour au sein de la firme PROLOGIA du Groupe Air Liquide, spécialisée dans la résolution de problèmes de planification de personnel. Soit $\{T_i\}_{i=1,\ldots,n}$ un ensemble de tâches journalières à affecter à des employés, chacune possédant une date de début d_i et une date de fin f_i. Un employé ne peut effectuer correctement un ensemble de tâches que si celles-ci ne se chevauchent pas dans le temps. Pour diverses raisons (code du travail, sécurité, maintenance des machines), un employé ne doit pas non plus effectuer plus de k tâches dans une journée de travail (avec généralement $k \leq 5$). La question qui se pose alors est : combien d'employés doit-on mobiliser pour qu'à la fin de la journée toutes les tâches aient été réalisées ? Bien entendu, un planning décrivant quelles sont les tâches à allouer à chaque employé est demandé en sus. La figure 1.3 montre un planning produit par le logiciel BAMBOO [7] et son moteur de planification automatique, auquel nous avons contribué.

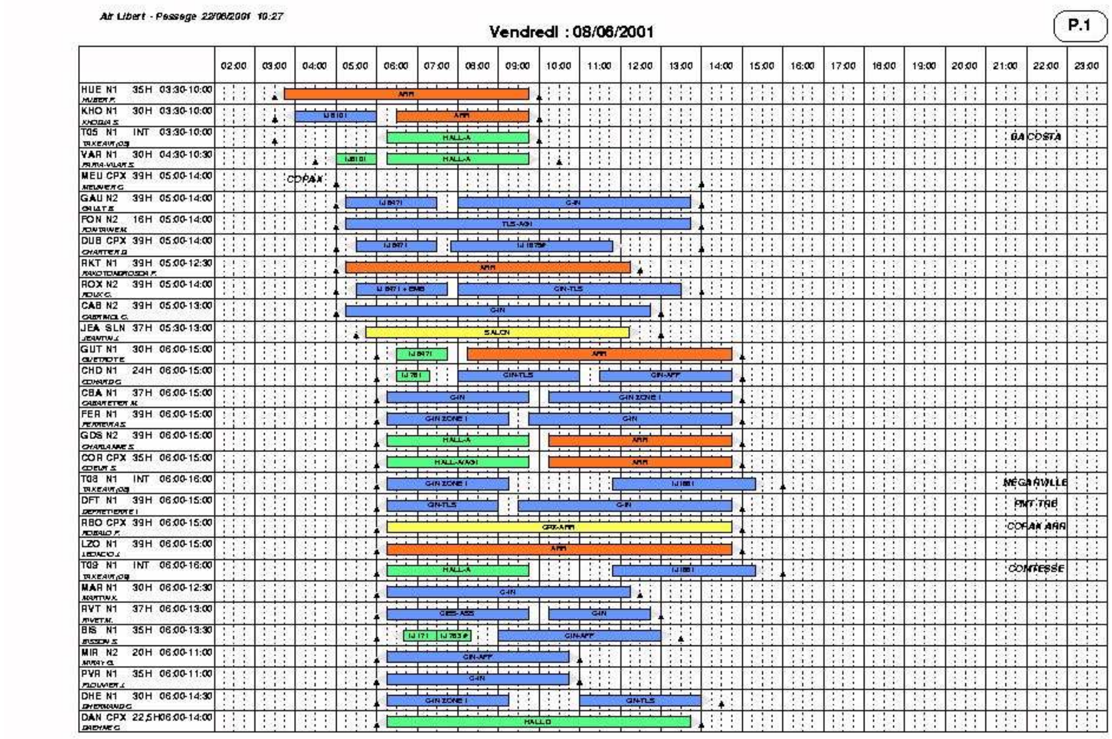


FIG. 1.3 – Un planning de personnel en aérogare.

Chaque tâche T_i n'étant ici qu'un intervalle $[d_i, f_i]$ de l'axe du temps, le problème revient à colorier le graphe d'intervalles sous-jacent tel que chaque couleur n'apparaisse pas plus de k fois, soit *le problème* ORDO *pour les graphes d'intervalles*. Lorsque le planning est cyclique (les mêmes tâches se répètent chaque jour et certaines d'entre elles sont à cheval sur deux jours consécutifs), le problème se ramène alors au *problème* ORDO *pour les graphes d'arcs circulaires*. Lorsqu'il manque des employés pour effectuer la totalité des tâches prévues (absences, sous-effectifs, mauvaises prévisions), il peut être intéressant d'autoriser le chevauchement entre certaines tâches lors de l'affectation. Dans ce cas, le problème se réduit au *problème* ORDO *pour les graphes de tolérances*. Les définitions et propriétés de ces différentes classes de graphes seront détaillées une prochaine section.

Malheureusement, un résultat de Bodlaender et Jansen [14] nous indique que même en se restreignant à ces classes de graphes, le problème d'ordonnancement avec exclusion mutuelle reste un problème *difficile*.

Théorème 1.1 (Bodlaender et Jansen, 1995) *Le problème* Ordo *est $\mathcal{NP}$-difficile pour les graphes d'intervalles, les graphes d'arcs circulaires et les graphes de tolérances, même lorsque k est un paramètre fixé supérieur ou égal à quatre.*

Malgré ce résultat négatif, nous ne pouvons pas nous résoudre à abandonner le problème. Nous citerons à ce propos Fred S. Roberts [132, p. 158] :

« Let us remark that in a real-world situation, it is not sufficient to say a problem is unsolvable or hard. Imagine the $500,000 consultant walking into the mayor's office and reporting that after carefully study, he has concluded that the problem of routing garbage trucks is hard ! Garbage trucks must be routed. So what can you do in such a situation ? The answer is, you develop partial solutions, you develop solutions which are applicable only to certain special situations, you modify the problem, or, in some cases, you even "lie". You lie by using results which are not necessarily true, but which seem to work. » [1]

Cette thèse a donc pour objet l'étude détaillée de la complexité du problème d'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles, de même que pour certaines de leurs extensions comme les graphes d'arcs circulaires et les graphes de tolérances. Nous exhibons notamment plusieurs cas polynomiaux qui s'avèrent significatifs en pratique et pour lesquels nous nous sommes attaché à concevoir des algorithmes à la fois simples et efficaces, c'est-à-dire pouvant être implantés sans grande difficulté par le praticien et dont la complexité en temps et espace est la plus faible possible. Une cartographie de la complexité du problème pour l'ensemble des classes de graphes étudiées sera présentée en conclusion. Mais avant d'entrer dans le détail du plan et des principaux résultats de la thèse, nous proposons au lecteur de se familiariser avec ces classes de graphes ainsi que les différents résultats parus sur le sujet, afin que celui-ci situe mieux la problématique et ses enjeux.

1.2 Définitions et notations basiques

Nous rappelons dans cette section quelques notions et définitions de base de la théorie des graphes et de la théorie de la complexité. La terminologie que nous employons est essentiellement empruntée à Berge [9] et Cormen *et al.* [28], et peut être retrouvée dans sa version anglaise chez Golumbic [75].

[1] *« Remarquons qu'en situation réelle, il ne suffit pas de dire que le problème est insoluble ou difficile. Imaginez le consultant à 500 000 $ entrant dans le bureau du maire et rapportant qu'après une étude minutieuse, il a conclu que le problème de l'acheminement des bennes à ordures est difficile ! Les bennes à ordures doivent être acheminées. Alors que pouvez-vous faire dans une telle situation ? La réponse est, vous développez des solutions partielles, vous développez des solutions qui sont applicables seulement à certaines situations particulières, vous modifiez le problème, ou même, dans certains cas, vous "mentez". Vous mentez en utilisant des résultats qui ne sont pas nécessairement vrais, mais qui semblent fonctionner. »*

Sur les graphes

Nous ne considérons que des graphes simples et finis. Soit $G = (V, E)$ un graphe. Sauf mention contraire, G sera non orienté et n et m dénoteront respectivement son nombre de sommets et son nombre d'arêtes. Une arête reliant le sommet $x \in V$ au sommet $y \in Y$ peut être indifféremment notée $xy \in E$ ou $(x, y) \in E$. Un graphe $G = (V, E)$ est *isomorphe* à un graphe $G' = (V', E')$ s'il existe une bijection $f : V \rightarrow V'$ tel que $(x, y) \in E$ si et seulement si $(f(x), f(y)) \in E'$. Le *complément* de G est le graphe $\overline{G} = (V, \overline{E})$ où $\overline{E} = \{(x, y) \mid x \in V, y \in V \text{ et } xy \notin E\}$. Le *sous-graphe induit* de G par $V' \subseteq V$ est le graphe $G' = (V', E')$ où $E' = \{(x, y) \mid x \in V', y \in V' \text{ et } xy \in E\}$. On dit qu'un sommet $x \in V$ est *voisin* d'un sommet $y \in V$, ou encore *adjacent* à un sommet $y \in Y$, si x et y sont reliés par une arête. Le *voisinage* de x est l'ensemble de tous les voisins de x ; le *degré* $d(x)$ du sommet x dénote le cardinal du voisinage de x. Le *degré maximum* du graphe G, c'est-à-dire $\max\{d(x) \mid x \in V\}$, est noté $\Delta(G)$. Un sommet *isolé* est un sommet dont le voisinage est l'ensemble vide.

Une *chaîne* ou un *chemin* de longueur ℓ est une séquence de sommets $v_0, v_1, \ldots, v_\ell$ où $v_{i-1}v_i \in E$ pour tout $i = 1, \ldots, \ell$. On appelle v_0 et v_ℓ les *extrémités* de la chaîne. Un graphe est *connexe* s'il existe une chaîne reliant tout couple de sommets ; un sous-graphe connexe maximal (par inclusion) est appelé *composante connexe*. Un *cycle* de longueur ℓ est une séquence de sommets $v_0, v_1, \ldots, v_{\ell-1}, v_0$ où $v_{i-1}v_i \in E$ pour tout $i = 1, \ldots, \ell$ et $v_{\ell-1}v_0 \in E$. Une *corde* dans un cycle (resp. une chaîne) est une arête $v_iv_j \in E$ tel que $j \neq i - 1$ et $j \neq i + 1$. Un cycle (resp. une chaîne) sans corde est appelé *cycle induit* (resp. *chaîne induite*). Le cycle (resp. la chaîne) sans corde de longueur ℓ est noté C_ℓ (resp. P_ℓ). On dit qu'un cycle est *pair* (resp. *impair*) lorsqu'il est de longueur paire (resp. impaire), *idem* pour une chaîne. Un cycle induit est aussi appelé *trou*.

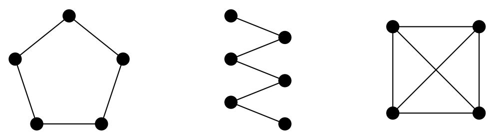


FIG. 1.4 – Le cycle sans corde C_5, la chaîne sans corde P_5 et la clique K_4.

Un *ensemble complet* ou *clique* est un sous-ensemble de sommets deux-à-deux adjacents. Une clique C est *maximale* (par inclusion) si aucune autre clique du graphe ne contient C comme sous-ensemble. La clique C est *maximum* si aucune autre clique du graphe n'est de taille strictement supérieure à celle de C ; $\omega(G)$ dénote la taille d'une clique maximum du graphe G. Par opposition, un *ensemble indépendant* ou *stable* est un sous-ensemble de sommets deux-à-deux non adjacents et la *stabilité* $\alpha(G)$ d'un graphe G dénote la taille d'un stable maximum dans G. Une *q-coloration* du graphe G correspond à une partition de G en q stables. $\chi(G)$ (resp. $\kappa(G)$) dénote le cardinal d'une coloration minimum (resp. d'une partition minimum en cliques) de G. Il est facile de vérifier que $\omega(G) \leq \chi(G)$ et $\alpha(G) \leq \kappa(G)$, de même que $\omega(\overline{G}) = \alpha(G)$ et $\chi(\overline{G}) = \kappa(G)$. Le nombre $\chi(G)$ est appelé *nombre chromatique* de G. Par analogie, le cardinal d'une coloration minimum de G tel que chaque couleur apparaisse au plus k fois sera noté $\chi(G, k)$; une borne inférieure du nombre $\chi(G, k)$ est donnée par $\max\{\chi(G), \lceil n/k \rceil\}$. Un *couplage* dans un graphe est un sous-ensemble d'arêtes tel que deux arêtes ne soient pas incidentes au même sommet. Un *couplage maximum* (resp. *couplage parfait*) est un couplage dont le cardinal est le plus grand possible (resp. est égal à $n/2$). Clairement, un couplage maximum correspond à une partition minimum en cliques de taille au plus deux ou encore à une partition en stables de taille au plus deux dans le complément. Enfin, une *coloration des arêtes* d'un graphe consiste en le marquage de chacune de ses arêtes par une couleur tel que deux arêtes incidentes à un même sommet aient des couleurs différentes.

Le *graphe complet* K_n est isomorphe à une clique sur n sommets ; le graphe K_3, isomorphe à C_3, est communément appelé *triangle*. On note tK_n le graphe composé de t copies disjointes de K_n. Un *graphe r-parti* est un graphe qui admet une partition en r stables (c'est-à-dire une r-coloration). Un *graphe biparti* admet une partition en deux stables ; on le note généralement $B = (X, Y, E)$ avec X et Y les deux ensembles indépendants de la partition. Un *graphe biparti complet* est un graphe biparti $B = (X, Y, E)$ où $E = \{(x, y) \mid x \in X \text{ et } y \in Y\}$; lorsque X et Y contiennent respectivement n et m sommets, celui-ci est noté $K_{n,m}$. Le graphe $K_{1,t}$ est appelé *étoile*, ou encore *griffe* lorsque $t = 3$ (de l'anglais *claw*). Un graphe sans $K_{1,3}$ est un graphe qui ne possède pas le graphe $K_{1,3}$ comme sous-graphe induit.

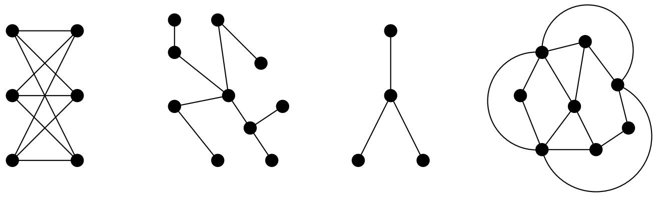


FIG. 1.5 – Le biparti complet $K_{3,3}$, un arbre, la griffe $K_{1,3}$ et un graphe planaire.

Un *arbre* est un graphe acyclique (*i.e.* sans cycle) et connexe ; une *forêt* est une union d'arbres deux-à-deux disjoints. Toute forêt à n sommets contient au plus $n - 1$ arêtes et peut être coloriée à l'aide de deux couleurs seulement. Un graphe est *planaire* s'il peut être dessiné dans le plan (ou sur la surface d'une sphère) de façon à ce que ses arêtes ne se croisent pas. Tout graphe planaire peut être colorié à l'aide de cinq couleurs (cf. [46, p. 96–97]) et même de quatre seulement, d'après le célèbre Théorème des Quatre Couleurs (cf. [46, p. 120–122] pour une note historique détaillée). Le *dual* d'un graphe G, noté $L(G)$, est le graphe d'incidence des arêtes de G : les arêtes de G sont les sommets de $L(G)$ et deux sommets de $L(G)$ sont adjacents si leurs arêtes correspondantes dans G sont incidentes à un même sommet. Les graphes duaux sont plus connus sous leur appellation anglaise de *line-graphs*. Toute coloration des arêtes d'un graphe G correspond à une coloration des sommets de son dual $L(G)$. Alors que toute coloration des arêtes demande un minimum de $\Delta(G)$ couleurs, un théorème de Vizing (cf. [46, p. 103–105]) établit que $\Delta(G) + 1$ couleurs suffisent. Par conséquent, nous avons $\Delta(G) \leq \chi(L(G)) \leq \Delta(G) + 1$ quel que soit le graphe G.

Sur les algorithmes et la complexité

Le modèle de machine que nous utilisons pour analyser la complexité de nos algorithmes est le modèle RAM de Cook et Reckhow (cf. [28, p. 6] et [75, p. 30]). Sauf mention contraire, un graphe $G = (V, E)$ en entrée d'un algorithme est représenté à l'aide de *listes d'adjacence*, soit pour chaque sommet $v \in V$ la liste des voisins de v ; une telle représentation requiert un espace de stockage de taille $O(n+m)$. Nous dirons qu'un problème de décision ou d'optimisation prenant un graphe en entrée peut être résolu en *temps et espace linéaire* s'il existe un algorithme permettant de résoudre ce problème en temps et espace $O(n+m)$ sur le modèle RAM ; un algorithme en temps et espace linéaire est généralement le mieux que l'on puisse espérer pour traiter un tel problème.

Le problème de la détermination de la clique maximum dans un graphe est appelé *problème de la clique maximum* ; de la même façon sont définis les *problèmes de la coloration minimum*, *du stable maximum*, *de la partition minimum en cliques* et *du couplage maximum*. Les quatre premiers problèmes sont connus pour être $\mathcal{NP}$-difficiles dans le cas général [104], tandis que le problème du couplage maximum peut être, lui, résolu en temps et espace polynomial [48]. Pour plus de détails sur la complexité de ces problèmes, consulter respectivement [28, p. 901–947] et [149, p. 113–123]. Le lecteur intéressé trouvera dans le livre de Garey et Johnson [60] une longue liste de problèmes $\mathcal{NP}$-complets et $\mathcal{NP}$-difficiles.

Enfin, une solution optimale $\mathcal{S}_{opt}$ d'un problème de minimisation (resp. de maximisation) peut être *β-approchée en temps et espace polynomial* s'il existe un algorithme polynomial en temps et espace retournant une solution $\mathcal{S}_{alg}$ à ce même problème tel que $\mathcal{S}_{alg} \leq \beta\mathcal{S}_{opt}$ (resp. $\mathcal{S}_{alg} \geq \mathcal{S}_{opt}/\beta$) ; l'algorithme en question est dit *d'approximation* et β est son *ratio*. De la même manière, une solution optimale $\mathcal{S}_{opt}$ d'un problème de minimisation peut être *approchée à β près* s'il existe un algorithme retournant une solution $\mathcal{S}_{alg}$ tel que $\mathcal{S}_{alg} \leq \mathcal{S}_{opt} + \beta$. Pour plus de détails sur la notion d'approximation en optimisation combinatoire, le lecteur pourra consulter l'introduction de Cormen *et al.* [28, p. 948–968] sur le sujet ainsi que la revue de résultats proposée par Hochbaum [88].

1.3 Graphes d'intervalles et classes apparentées

Dans cette section, nous définissons précisément les classes de graphes qui interviennent dans ce mémoire. Certaines d'entre elles n'apparaîtront que brièvement, d'autres seront omniprésentes comme par exemple les graphes d'intervalles, les graphes d'arcs ou encore les graphes de tolérances. Nous donnerons en fin de section une hiérarchie de l'ensemble de ces classes (voir la figure 1.17).

Les graphes parfaits

Un graphe G est *parfait* si et seulement si pour tout sous-graphe induit $G' \subseteq G$, les égalités $\omega(G') = \chi(G')$ et $\alpha(G') = \kappa(G')$ sont satisfaites. Lovász (cf. [75, p. 53–58]) a montré qu'en fait, une des deux égalités suffisait, c'est-à-dire qu'un graphe est parfait si et seulement si son complément l'est aussi. Cette propriété est connue sous le nom de Théorème des Graphes Parfaits.

Pendant plus de quarante ans, la Conjecture Forte des Graphes Parfaits lancée par Claude Berge – un graphe est parfait si et seulement si lui-même et son complément ne contiennent aucun trou impair de longueur supérieure ou égale à cinq – aura passionné nombre de mathématiciens (pour de plus amples détails sur la conjecture, consulter [75,

p. 71–75]). Cette conjecture a finalement été prouvée en 2002 par Chudnovsky *et al.* [24, 25] (voir aussi [34]). Dans la foulée, Cornuéjols *et al.* [35] présentaient un algorithme polynomial pour reconnaître les graphes parfaits.

Du point de vue de la complexité, un premier résultat fondamental de Grötschel *et al.* [79] établissait déjà que les problèmes de la clique maximum, de la coloration minimum, du stable maximum et de la partition minimum en cliques peuvent être résolus en temps et espace polynomial pour les graphes parfaits. Malheureusement, la méthode de l'ellipsoïde (cf. [119, p. 131–148] ou [136, p. 163–189]) utilisée par les trois auteurs s'avère inefficace en pratique ; la recherche d'algorithmes combinatoires simples et réellement efficaces pour ces problèmes restent donc d'actualité.

Le livre de Golumbic [75] constitue une excellente introduction au "petit monde des graphes parfaits" et contient de nombreuses références sur le sujet (voir aussi [22, 128]. La plupart des graphes que nous allons maintenant présenter sont parfaits. Nous allons d'ailleurs commencer par les deux premières classes de graphes à avoir été reconnus comme parfaits : les graphes triangulés (cf. [75, p. 94–95]) et les graphes de comparabilité (cf. [75, p. 132–133]).

Les graphes triangulés

Un graphe est un *graphe triangulé* s'il ne contient aucun cycle induit de longueur supérieure ou égale à quatre (les graphes triangulés apparaissent sous le nom de *chordal graphs* dans la littérature anglophone). Les graphes triangulés correspondent exactement aux graphes d'intersection des sous-arbres dans un arbre, ce qui leur confère certaines applications dans le domaine de la classification. Ils sont reconnaissables en temps et espace linéaire (cf. [75, p. 84–91]) et les problèmes de la clique maximum, de la coloration minimum, du stable maximum et de la partition minimum en cliques peuvent être résolus en temps et espace linéaire lorsque l'on se restreint à ceux-ci [70] (voir aussi [75, p. 98–100]). Pour plus de détails sur ces graphes, leurs propriétés et leurs applications, nous renvoyons le lecteur à [75, p. 81–104].

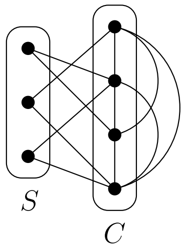


FIG. 1.6 – Un graphe scindé.

Une sous-classe intéressante des graphes triangulés est la classe des graphes scindés. Un *graphe scindé* est un graphe dont les sommets admettent une partition en deux sous-ensembles S et C où S un est stable et C une clique ; par analogie aux graphes bipartis, nous noterons $G = (S, C, E)$ le graphe en question (voir la figure 1.6). Clairement, le complément d'un graphe scindé est aussi un graphe scindé ; en fait, il s'avère qu'un graphe G est scindé si et seulement si G et $\overline{G}$ sont triangulés (cf. [75, p. 151–152]). Les graphes scindés peuvent être reconnus en temps et espace linéaire (cf. [75, p. 154]). Pour plus de détails sur ces graphes et leurs propriétés, le lecteur est invité à consulter [75, p. 149–156].

Voici deux extensions des graphes triangulés dont nous discuterons brièvement dans le mémoire : les graphes de Meyniel et les graphes faiblement triangulés. Un graphe est dit *de Meyniel* s'il possède la propriété suivante : tout cycle impair de longueur supérieure ou égale à cinq possède au moins deux cordes. Ces graphes sont parfaits (cf. [75, p. 95–98]) et peuvent être reconnus en temps $O(n^4)$ [134]. Lévêque et Maffray [109] ont récemment présenté un algorithme en temps et espace linéaire pour résoudre les problèmes de la clique maximum et de la coloration minimum sur les graphes de Meyniel. Un graphe est *faiblement triangulé* si lui-même ou son complément ne possèdent pas de cycle induit de longueur supérieure ou égale à cinq (le complément d'un graphe faiblement triangulé est donc faiblement triangulé). Ces graphes sont parfaits [84] et peuvent être reconnus en temps $O(n^4)$ [144]. Hayward *et al.* [85] ont donné des algorithmes en temps $O(n^4)$ pour résoudre les problèmes de la clique maximum et de la coloration minimum.

Les graphes de comparabilité

Un graphe est dit *de comparabilité* si ses arêtes peuvent être transitivement orientées. Une orientation transitive des arêtes satisfait la condition suivante : s'il existe un arc allant du sommet x vers le sommet y et un arc allant du sommet y vers le sommet z, alors il existe aussi un arc allant de x vers z. Autrement dit, un graphe de comparabilité est le graphe d'un ordre partiel.

Déterminer une orientation des arêtes de façon à ce que celle-ci soit transitive si et seulement si le graphe est de comparabilité peut être fait en temps et espace linéaire [121]. D'un autre côté, tester si une orientation est transitive ou non se réduit au problème de la multiplication de matrices, pour lequel le meilleur algorithme connu à ce jour prend un temps $O(n^{2.376})$ (cf. [28, p. 763]). Les problèmes de la clique maximum et de la coloration minimum peuvent être résolus en temps et espace linéaire, étant donnée en entrée une orientation transitive du graphe de comparabilité (cf. [75, p. 132–133] et [121]). Les problèmes du stable maximum et de la partition minimum en cliques se réduisent au couplage maximum dans un graphe biparti [89, 2, 51]. Pour plus de détails sur ces graphes et leurs propriétés, nous renvoyons le lecteur à [75, p. 105–148].

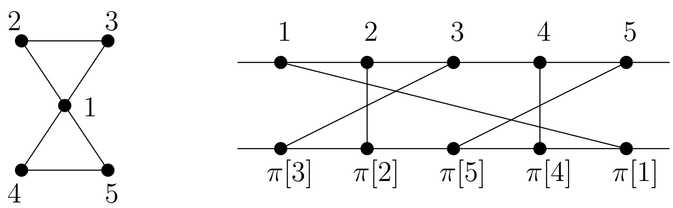


FIG. 1.7 – Un graphe de permutation et son diagramme.

Une sous-classe des graphes de comparabilité connue pour ses nombreuses applications est la classe des graphes de permutation. Soit $V = \{1, 2, \ldots, n\}$ et $\pi = [\pi[1], \pi[2], \ldots, \pi[n]]$

une permutation de V. Le graphe $G(\pi) = (V, E)$ est défini tel que $ij \in E$ si et seulement si $(i - j)(\pi^{-1}[i] - \pi^{-1}[j]) < 0$ où $\pi^{-1}[i]$ dénote la position du nombre i dans π. Un graphe G est dit *de permutation* s'il existe une permutation π tel que G soit isomorphe à $G(\pi)$. La figure 1.7 représente le diagramme de permutation correspondant à un certain graphe de permutation : les sommets i et j sont reliés par une arête dans le graphe si la ligne qui relie i à $\pi[i]$ dans le diagramme coupe la ligne qui relie j à $\pi[j]$ (cf. [75, p. 164–168]).

Pour obtenir le complément $\overline{G}(\pi)$ d'un graphe de permutation, il suffit de renverser la séquence π. Ainsi, le complément $\overline{G}(\pi)$ est aussi un graphe de permutation. En fait, un graphe est de permutation si et seulement si lui-même et son complément sont de comparabilité (cf. [75, p. 158–159]). Les graphes de permutation sont reconnaissables en temps $O(n^2)$ [142] ; lorsque le test s'avère positif, l'algorithme de reconnaissance fournit en plus la permutation π en sortie. Les problèmes de la clique maximum et de la coloration minimum (dont une application est le tri de permutations à l'aide du minimum de files) sont solubles en temps $O(n \log n)$, étant donnée en entrée une permutation π représentant le graphe (cf. [75, p. 164–168]). Pour plus de détails sur ces graphes et leurs applications, nous invitons le lecteur à consulter [75, p. 157–170].

Mentionnons enfin la classe des *graphes des ordres partiels séries-parallèles*. Communément appelés *cographes*, ceux-ci correspondent exactement aux graphes sans P_4 [140]. Le complément d'un cographe étant aussi un cographe, ceux-ci forment une sous-classe des graphes de permutation. Ils peuvent être reconnus et coloriés en temps et espace linéaire [33, 140].

Les graphes d'intervalles

Un graphe $G = (V, E)$ est un *graphe d'intervalles* si à chaque sommet $v \in V$ peut être associé un intervalle ouvert I_v de l'axe des réels tel que pour toute paire $u, v \in V$ de sommets, $uv \in E$ si et seulement si $I_u \cap I_v \neq \emptyset$ (voir la figure 1.2). La famille $\mathcal{I} = \{I_v\}_{v \in V}$ induit une *représentation par intervalles* du graphe G. Les *extrémités gauches et droites* d'un intervalle I_v sont respectivement notées $g(I_v)$ et $d(I_v)$, et l'ordre linéaire induit par les extrémités gauches (resp. droites) croissantes sur l'ensemble $\mathcal{I}$ est noté $<_g$ (resp. $<_d$). Voici une caractérisation de la classe des graphes d'intervalles, due à Gilmore et Hoffman [72] (voir aussi [75, p. 172–173]).

Théorème 1.2 (Gilmore et Hoffman, 1964) *Pour tout graphe $G = (V, E)$, les assertions suivantes sont équivalentes :*

(1) G est un graphe d'intervalles ;

(2) G est un graphe triangulé et $\overline{G}$ un graphe de comparabilité ;

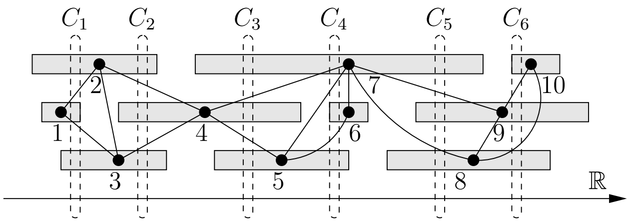
(3) les cliques maximales de G peuvent être ordonnées linéairement tel que pour tout sommet $v \in V$, les cliques maximales contenant v apparaissent consécutivement dans cette ordre.

Le complément d'un graphe d'intervalles est donc transitivement orientable. En fait, une orientation du complément s'obtient simplement en traçant un arc allant de $u \in V$

vers $v \in V$ si $d(I_u) \leq g(I_v)$. L'ordre partiel ainsi défini est appelé *ordre d'intervalles* (nous écrirons $I_u \prec I_v$ lorsque $d(I_u) \leq g(I_v)$).

Une matrice dont les coefficients sont dans $\{0, 1\}$ a la *propriété des 1s consécutifs* pour les colonnes (resp. lignes) si ses lignes (resp. colonnes) peuvent être permutées de façon à ce que les 1s de chaque colonne (resp. ligne) apparaissent consécutivement. L'assertion (3) du théorème peut alors être réécrite en les termes suivants : *la matrice d'incidence cliques maximales-vs-sommets de G a la propriété des 1s consécutifs pour les colonnes* [56] (voir aussi [75, p. 174]). De plus, le nombre de lignes de cette matrice (c'est-à-dire le nombre de cliques maximales) est inférieur à n, avec égalité si et seulement si le graphe ne comporte aucune arête.

Booth et Lueker [18] (voir aussi [75, p. 175–181]) ont été les premiers à donner un algorithme en temps et espace linéaire pour reconnaître les graphes d'intervalles ; leur algorithme, basé sur une structure de données complexe appelée PQ-arbre, permet en fait d'identifier la propriété des 1s consécutifs. Depuis, d'autres algorithmes plus simples, fondés sur des approches du type recherche en largeur comme Lex-BFS (cf. [75, p. 84–87]), ont vu le jour [32, 82].



	1	2	3	4	5	6	7	8	9	10
C_1	1	1	1							
C_2		1	1	1						
C_3				1	1		1			
C_4					1	1	1			
C_5							1	1	1	
C_6								1	1	1

FIG. 1.8 – Un graphe d'intervalles et sa matrice cliques maximales-*vs*-sommets.

La proposition suivante établit d'une part l'équivalence entre représentation par intervalles ouverts et représentation par intervalles fermés, et d'autre part, nous indique comment construire efficacement celles-ci.

Proposition 1.1 (Représentation) *Tout graphe d'intervalles $G = (V, E)$ admet une représentation par intervalles ouverts (resp. fermés) dont les extrémités sont des entiers compris entre 1 et n (resp. $2n$). De plus, une telle représentation s'obtient en temps et espace linéaire à partir du graphe.*

Preuve. Notons $A = (a_{ij})$ la matrice d'incidence cliques maximales-*vs*-sommets de G. En accord avec la discussion précédente, nous pouvons considérer que les 1s de chaque colonne de A apparaissent consécutivement. Cette matrice peut être construite en temps et espace linéaire à partir du graphe d'intervalles [82]. Pour chaque sommet $v \in V$, définissons l'intervalle ouvert I_v tel que $g(I_v) = \min\{i \mid a_{iv} = 1\}$ et $d(I_v) = \max\{i \mid a_{iv} = 1\}$. Cette représentation du graphe G est correcte car deux intervalles se chevauchent si et seulement si les sommets qui leur correspondent sont adjacents. De plus, la matrice A ayant au plus n lignes, les extrémités de I_v sont nécessairement dans $\{1, \ldots, n\}$.

Voyons maintenant comment obtenir à partir de cette représentation, une représentation du graphe G par intervalles fermés. Tout d'abord, trions dans l'ordre croissant l'ensemble des extrémités des intervalles, gauches et droites mélangées (en cas d'égalité, les extrémités droites sont placées avant les extrémités gauches). Ensuite, pour $i = 1, \ldots, 2n$, assignons à la $i^{ème}$ extrémité la valeur i. Les n intervalles sont alors redéfinis comme fermés et leurs nouvelles extrémités sont dans $\{1, \ldots, 2n\}$. Les ordres $<_g$ et $<_d$ n'ayant pas changé, le graphe d'intervalles sous-jacent reste le même. Les $2n$ entiers étant tous entre 1 et n, le tri peut se faire par paquets en temps et espace $O(n)$, ce qui complète la preuve. □

Remarque. Étant donné un graphe d'intervalles, une représentation par intervalles ouverts (ou par intervalles fermés) ordonnée selon $<_g$ ou $<_d$ peut donc être calculée en temps et espace linéaire.

Les graphes d'intervalles étant triangulés, les problèmes de la clique maximum, de la coloration minimum, du stable maximum et de la partition minimum en cliques peuvent être résolus en temps et espace linéaire. Toutefois, Gupta *et al.* [80, 81] proposent des algorithmes en temps $O(n \log n)$ pour résoudre ces problèmes, étant donnée en entrée une représentation par intervalles.

Pour plus de détails sur les graphes d'intervalles et leurs applications, nous renvoyons le lecteur à [132, p. 113–140], [133], [75, p. 171–202] et [54]. Nous allons maintenant présenter deux sous-classes des graphes d'intervalles qui illustrent parfaitement la variété de leur champ d'application : les graphes d'intervalles propres et les graphes à seuil.

Les graphes d'intervalles propres

Les graphes d'intervalles propres (et les graphes d'intervalles unitaires) ont été introduits par Roberts [130, 131] pour modéliser l'indifférence en théorie du choix social et en psychologie.

Un graphe est un *graphe d'intervalles propres* s'il admet une représentation par intervalles dans laquelle aucun intervalle n'en contient proprement un autre (voir la figure 1.9) ; une telle représentation est appelée *représentation par intervalles propres*. De la même manière, un graphe est un *graphe d'intervalles unitaires* s'il admet une représentation dans laquelle tous les intervalles sont de même taille ; une telle représentation est appelée *représentation par intervalles unitaires*.

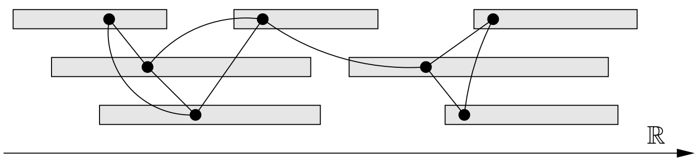


FIG. 1.9 – Un graphe d'intervalles propres et sa représentation.

En 1969, Roberts [131] (voir aussi [75, p. 187–188]) démontrait que la classe des graphes d'intervalles propres et la classe des graphes d'intervalles unitaires coïncident. Il établissait notamment que les graphes d'intervalles sans $K_{1,3}$ sont des graphes d'intervalles unitaires en utilisant la caractérisation de Scott et Suppes des *semi-ordres* [138] ; les semi-ordres (de l'anglais *semiorders*) sont des ordres partiels utilisés en théorie du choix social pour modéliser la préférence [115] (voir aussi [132, p. 534–540] et [75, p. 185–186]). Les implications triviales "unitaires $\Rightarrow$ propres $\Rightarrow$ sans $K_{1,3}$" pour les graphes d'intervalles lui permettaient alors d'obtenir le résultat. Récemment, Bogart et West [17] ont proposé une preuve constructive du résultat, dans laquelle les intervalles propres sont graduellement convertis en intervalles unitaires par des séquences de dilatations, contractions et translations.

Nous présentons ici une nouvelle preuve de ce résultat [2], complètement combinatoire et sans manipulation de coordonnées. La représentation par intervalles unitaires est construite directement à partir de la matrice d'incidence cliques maximales-*vs*-sommets, qui elle-même s'obtient sans difficulté à partir d'une représentation par intervalles propres. La validité de la construction repose sur le fait que cette matrice a la propriété des 1s consécutifs à la fois pour les lignes et pour les colonnes. La preuve fournit en sus un algorithme en temps et espace linéaire pour calculer une représentation par intervalles unitaires, étant donné en entrée un graphe d'intervalles propres.

Pour d'autres caractérisations concernant les graphes d'intervalles propres ou unitaires, le lecteur est renvoyé aux travaux de Wegner [157] et Roberts [130] sur le sujet (voir aussi [75, p. 195]).

Théorème 1.3 (Roberts, 1969) *Pour un graphe $G = (V, E)$, les assertions suivantes sont équivalentes :*

(1) G est un graphe d'intervalles propres ;

(2) la matrice d'incidence cliques maximales-vs-sommets de G a la propriété des uns consécutifs à la fois pour les lignes et pour les colonnes ;

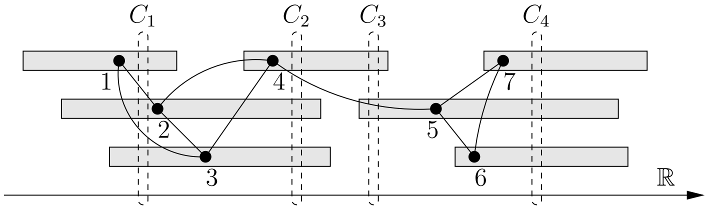
(3) G est un graphe d'intervalles unitaires ;

(4) G est un graphe d'intervalles sans $K_{1,3}$.

Preuve. (1) $\Rightarrow$ (2). Lorsqu'aucun intervalle n'en contient un autre, l'ordre induit par les extrémités gauches des intervalles est le même que l'ordre induit par les extrémités droites. Soit u et v les premier et dernier sommets d'une clique maximale sous cet ordre. L'intervalle I_u s'étend suffisamment vers la droite pour contenir l'extrémité gauche de I_v, ce qui est aussi vrai pour tout intervalle qui débute entre les extrémités droites de I_u et I_v. Ainsi, la clique consiste précisément en ces sommets, consécutifs dans l'ordre en question (voir la figure 1.10). Puisque les cliques occupent des intervalles de sommets consécutifs dans cet ordre, et que le caractère maximal des cliques implique qu'aucun de ces intervalles n'en contient un autre, il suffit de placer celles-ci par ordre croissant de leurs premiers sommets pour voir que les cliques contenant un sommet donné apparaissent consécutivement.

(2) $\Rightarrow$ (3). Considérons une matrice d'incidence cliques maximales-*vs*-sommets satisfaisant la propriété des uns consécutifs pour les lignes et les colonnes, de façon à ce que la $i^{ème}$ clique C_i soit composée des sommets consécutifs $v_{a_i}, \ldots, v_{b_i}$ dans l'ordre des sommets, avec $\langle a_i \rangle$ et $\langle b_i \rangle$ deux séquences strictement croissantes (voir la figure 1.10). Tout d'abord, représentons les sommets de C_1 par b_1 intervalles unitaires distincts s'intersectant deux-à-deux. Pour $i > 1$, ayant assigné des intervalles unitaires aux sommets de $\bigcup_{i<j} C_i$ pour représenter le sous-graphe qu'ils induisent, nous allons ajouter à cette représentation des

[2] Celle-ci peut être retrouvée dans [63] (en anglais).



	1	2	3	4	5	6	7
C_1	1	1	1	0	0	0	0
C_2	0	1	1	1	0	0	0
C_3	0	0	0	1	1	0	0
C_4	0	0	0	0	1	1	1
	a_1	a_2		a_3	a_4		
			b_1	b_2	b_3		b_4

FIG. 1.10 – Un graphe d'intervalles propres et sa matrice cliques-*vs*-sommets.

intervalles unitaires distincts correspondant aux sommets de $C_j - C_{j-1}$, de façon à obtenir une représentation par intervalles unitaires du sous-graphe induit par $\bigcup_{i \leq j} C_i$. Pour cela, plaçons $g(v_{b_{j-1}+1}), \ldots, g(v_{b_j})$, dans l'ordre, entre $d(v_{a_j-1})$ et $d(v_{a_j})$. Comme $v_{a_j-1} \in C_{j-1}$, nous avons $g(v_{b_{j-1}}) < d(v_{a_j-1})$, et par conséquent, les extrémités des intervalles apparaissent dans l'ordre désiré. Ensuite, étendons les intervalles jusqu'à la longueur unité en posant $d(v_k) = g(v_k) + 1$ pour $b_{j-1} + 1 \leq k \leq b_j$. Tous ces intervalles se terminent après $d(v_{b_{j-1}})$. Chaque membre de $C_j - C_{j-1}$ étant adjacent à tous les sommets $v_{a_j}, \ldots, v_{b_{j-1}}$ mais à aucun des sommets précédents, nous avons une représentation par intervalles unitaires du sous-graphe induit par $\bigcup_{i \leq j} C_i$.

(3) $\Rightarrow$ (4). Considérons une représentation de G par intervalles unitaires et supposons que ce même graphe G contienne une copie de $K_{1,3}$, induite par un intervalle I_d recouvrant trois intervalles $I_a \prec I_b \prec I_c$. L'intervalle I_b étant entièrement recouvert par l'intervalle I_d, ce dernier est nécessairement d'une longueur strictement supérieure à celle de I_b, ce qui est une contradiction.

(4) $\Rightarrow$ (1). Considérons une représentation de G par intervalles et supposons qu'un intervalle I_b soit inclus dans un intervalle I_d. Puisque G est sans $K_{1,3}$, il ne peut exister deux intervalles I_a et I_c intersectant chacun I_d tel que $I_a \prec I_b \prec I_c$. Si l'intervalle I_a (resp. I_c) chevauche I_d tel que $I_a \prec I_b$ (resp. $I_b \prec I_c$), alors nous pouvons étendre l'extrémité droite (resp. gauche) de I_b jusqu'à $d(I_d) + \epsilon$ (resp. $g(I_d) - \epsilon$), avec $\epsilon > 0$, sans modifier le graphe G. Après avoir répété cette opération tant qu'il existe un intervalle en contenant un autre, nous obtenons une représentation par intervalles propres du graphe G. $\square$

Note. L'assertion (2) du théorème apparaît en d'autres termes dans [54, p. 85] et peut être encore formulée comme suit : il existe un ordre linéaire $v_1, \ldots, v_n$ sur V tel que pour tout $i < j$, $v_i v_j \in E$ implique que tous les sommets entre v_i et v_j dans cet ordre induisent une clique. Notons que cette dernière assertion est la transposition naturelle pour les graphes d'intervalles propres d'une caractérisation des graphes d'intervalles qui apparaît dans [127, 97, 124] : l'existence d'un ordre linéaire $v_1, \ldots, v_n$ tel que pour tout $i < j$, $v_i v_j \in E$ implique que $v_k v_j \in E$ si $i < k < j$.

Généralement, les algorithmes de reconnaissance pour graphes d'intervalles propres produisent en temps et espace linéaire un ordre linéaire sur V comme décrit dans la note ci-dessus (voir par exemple [114, 31, 38, 39, 125, 30]). Or, certaines applications requièrent la connaissance d'une représentation par intervalles unitaires du graphe.

Connaissant l'ordre linéaire sur les sommets, l'ensemble ordonné des cliques maximales peut être simplement calculé en temps et espace linéaire. De là, la construction donnée dans la preuve de l'implication (2) $\Rightarrow$ (3) du théorème 1.3 fournit un algorithme en temps et espace linéaire pour calculer une représentation par intervalles unitaires. Celle-ci est plus efficace que la construction proposée par Bogart et West [17], qui calcule un modèle par intervalles unitaires en temps quadratique (*i.e.* en temps $O(n^2)$) à partir d'une représentation par intervalles propres. Toutefois, les représentations produites par notre algorithme, comme celui de Bogart et West [17], ne sont pas efficaces dans le sens où les extrémités des intervalles unitaires sont des nombres rationnels dont le dénominateur peut être exponentiel en n. Corneil *et al.* [31] ont, en revanche, montré que leur algorithme de reconnaissance basé sur la recherche en largeur peut être utilisé pour construire une représentation par intervalles unitaires dont les extrémités sont des rationnels de dénominateur n et de numérateur inférieur à n^2.

Proposition 1.2 (Représentation) *Tout graphe d'intervalles propres $G = (V, E)$ admet une représentation par intervalles unitaires dont les extrémités sont des rationnels de dénominateur n et de numérateur inférieur à n^2. De plus, une telle représentation s'obtient en temps et espace linéaire à partir du graphe.*

Note. Une question reste posée : existe-t-il un algorithme direct (sans l'utilisation de la recherche en largeur) en temps et espace linéaire permettant de calculer un modèle efficace par intervalles unitaires à partir d'un graphe d'intervalles propres ?

Les graphes à seuil

Les graphes à seuil ont été introduits par Chvátal et Hammer (cf. [75, p. 219–223]) dans un contexte lié à la programmation linéaire en nombres entiers et ont été redécouverts par Henderson et Zalcstein (cf. [75, p. 229–231]) dans le cadre de la synchronisation de processus parallèles.

Soit $V = \{v_1, v_2, \ldots, v_n\}$ l'ensemble des sommets d'un graphe G. Tout sous-ensemble $X \subseteq V$ de sommets peut être représenté par son vecteur caractéristique $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, où pour tout i, $x_i = 1$ si $v_i \in X$ et $x_i = 0$ sinon. De cette façon, à chaque sous-ensemble de sommets correspond un et un seul sommet de hypercube unité dans $\mathbb{R}^n$. Considérons maintenant l'ensemble de tous les stables de G. Nous posons alors la question suivante : existe-t-il un hyperplan qui coupe l'hypercube unité en deux tel que d'un côté, nous ayons tous les sommets correspondant aux stables de G et de l'autre tous les sommets correspondant aux sous-ensembles non stables de G ? Autrement dit, peut-on séparer les sous-ensembles stables de G à l'aide d'une seule inégalité linéaire $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq t$? Si la réponse est oui, alors le graphe en question est appelé graphe à seuil.

Ainsi, un graphe $G = (V, E)$ est un *graphe à seuil* si à chaque sommet $v \in V$ peut être associé un entier positif a_v de manière à ce que $X \subseteq V$ soit un stable si et seulement si $\sum_{x \in X} a_x \leq t$ avec t une constante entière (appelé *seuil*). Les graphes à seuil peuvent être reconnus en temps et espace linéaire du fait de leur structure très particulière : tout graphe à seuil est représentable par une clique $C = C_1 \cup \cdots \cup C_r$ et un stable $S = S_1 \cup \cdots \cup S_r$ ($r \leq n$ et C_i, S_i non vides pour $i = 1, \ldots, r$) tel qu'un sommet de S_i est adjacent à un sommet de $C_{i'}$ si et seulement si $i' > i$ pour tout $i, i' \in \{1, \ldots, r\}$ (cf. [75, p. 223–227]).

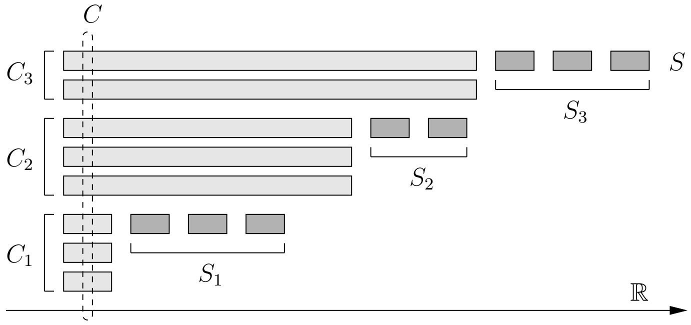


FIG. 1.11 – Une représentation d'un graphe à seuil par intervalles.

Les graphes à seuil forment une sous-classe des graphes scindés et des graphes de permutation ; par conséquent, tout graphe à seuil est aussi un graphe d'intervalles (voir la figure 1.11). Notons encore que tout sous-graphe induit d'un graphe à seuil ou de son complément est un graphe à seuil. Pour plus de détails sur ces graphes, leurs propriétés et leurs applications, nous renvoyons le lecteur à [75, p. 219–234].

Les graphes de tolérances

Un graphe $G = (V, E)$ est un *graphe de tolérances* si à chaque sommet $v \in V$ peut être associé un intervalle ouvert I_v de l'axe des réels et un réel $t(v)$ correspondant à sa tolérance, tel que pour toute paire $u, v \in V$ de sommets, $uv \in E$ si et seulement si $|I_u \cap I_v| \geq \min\{t_u, t_v\}$. L'ensemble des intervalles munis de leur tolérance forme une *représentation par tolérances* du graphe G (selon le contexte, le mot *tolérance* sera employé pour désigner la tolérance à proprement parler ou bien le couple composé de l'intervalle et de sa tolérance). Lorsqu'un graphe possède une représentation par tolérances où pour chaque sommet $v \in V$ la valeur de $t(v)$ est inférieure ou égale à la longueur de l'intervalle I_v, le graphe en question est appelé *graphe de tolérances bornées*. Les graphes de tolérances forment une sous-classe des graphes faiblement triangulés et de fait sont parfaits [76].

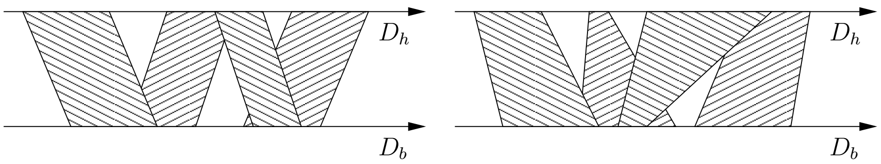


FIG. 1.12 – Une représentation du graphe P_3 par parallélogrammes, et par trapèzes.

Un graphe $G = (V, E)$ est un *graphe de parallélogrammes* s'il existe deux droites parallèles D_h et D_b tel qu'à chaque sommet $v \in V$ on peut faire un correspondre un parallélogramme P_v dont les côtés opposés se confondent avec D_h et D_b et pour toute paire de sommets $u, v \in V$, $uv \in V$ si et seulement si P_u et P_v s'intersectent (voir la figure 1.12). Les graphes de tolérances bornées correspondent exactement aux graphes de parallélogrammes, formant ainsi une sous-classe des compléments de graphes de comparabilité et des *graphes* *de trapèzes* [52]. Notons encore que les graphes de permutation sont des graphes de parallélogrammes (et donc des graphes de tolérances bornées).

De la même façon que pour les graphes d'intervalles, nous pouvons définir les *graphes de tolérances propres* et les *graphes de tolérances unitaires* [16] (attention, le mot unitaire signifie ici que l'*intervalle* est de longueur unitaire). Néanmoins, les graphes de tolérances unitaires (et donc les graphes de tolérances propres) ne sont pas nécessairement sans $K_{1,3}$ (voir la figure 1.13).

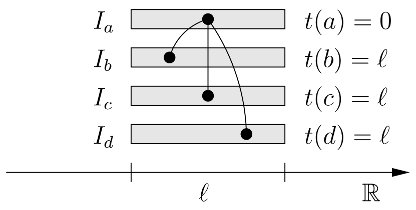


FIG. 1.13 – Une représentation du graphe $K_{1,3}$ par tolérances unitaires et bornées.

Le complexité du problème de la reconnaissance des graphes de tolérances reste à ce jour une question ouverte. Golumbic et Siani [77] ont donné un algorithme en temps $O(n^2)$ pour résoudre les problèmes de la clique maximum et de la coloration minimum, étant donnée en entrée une représentation du graphe par tolérances ; la méthode qu'ils proposent est basée sur l'algorithme en temps $O(n \log n)$ de Felsner *et al.* [52] pour résoudre les mêmes problèmes sur les graphes de trapèzes (ou de parallélogrammes).

Les graphes d'arcs circulaires

Le graphe des intersections d'une collection d'arcs sur un cercle est appelé *graphe d'arcs circulaires* (ou simplement *graphe d'arcs*). Tout graphe d'arcs $G = (V, E)$ possède une *représentation par arcs* $\mathcal{A} = \{A_v\}_{v \in V}$ (tous ouverts ou tous fermés) dans laquelle chaque arc A_v est défini par son *extrémité "dans le sens contraire des aiguilles d'une montre"*, noté $scm(A_v)$, et son *extrémité "dans le sens des aiguilles d'une montre"*, noté $sm(A_v)$ (voir la figure 1.14). Nous dirons qu'une représentation par arcs est *ordonnée* lorsque les extrémités scm (ou sm) des arcs apparaissent dans l'ordre circulaire. Lorsque un point p du cercle n'est pas couvert par au moins un arc de la représentation, le cercle peut être coupé au point p et redressé de façon à obtenir une droite, les arcs devenant des intervalles. Ainsi, il est aisé de constater que tout graphe d'intervalles est un graphe d'arcs.

Comme pour les graphes d'intervalles et les graphes de tolérances, nous pouvons distinguer les *graphes d'arcs propres* et les *graphes d'arcs unitaires*. En revanche, l'égalité "propre = unitaire" n'est plus valide dans le cadre des graphes d'arcs. La figure 1.15 montre deux graphes qui admettent une représentation par arcs propres, mais aucune représentation par arcs unitaires. Considérons par exemple le graphe de gauche sur cette figure, dont les sommets sont numérotés, et admettons que celui-ci puisse être représenté à l'aide d'arcs unitaires. L'ensemble de sommets $\{2, 4, 6\}$ induit un cycle de longueur trois ; la longueur u des arcs unitaires ouverts (resp. fermés) représentant les sommets de cet ensemble doit donc satisfaire l'inégalité $u > 2\pi/3$ (resp. $u \geq 2\pi/3$). D'un autre côté, l'ensemble de sommets

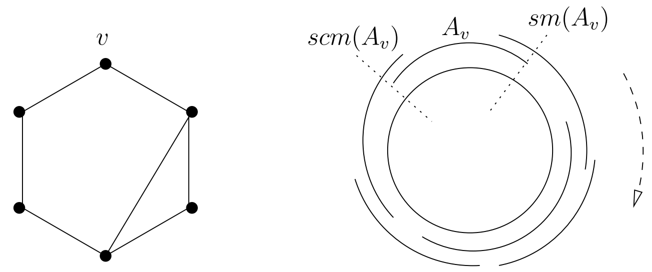


FIG. 1.14 – Un graphe d'arcs et sa représentation.

$\{1,3,5\}$ induit lui un stable de taille trois ; la longueur u des arcs unitaires ouverts (resp. fermés) représentant les sommets de cet ensemble doit donc satisfaire l'inégalité $u \leq 2\pi/3$ (resp. $u < 2\pi/3$), ce qui contredit l'affirmation précédente. Notons que les graphes d'arcs propres sont, comme les graphes d'intervalles propres, des graphes sans $K_{1,3}$ (se reporter à [151] pour une caractérisation des graphes d'arcs propres et des graphes d'arcs unitaires par sous-graphes induits interdits). Le cycle induit C_5 possédant une représentation par arcs unitaires, les graphes d'arcs unitaires ne sont en général pas des graphes parfaits (voir par exemple la figure 1.14).

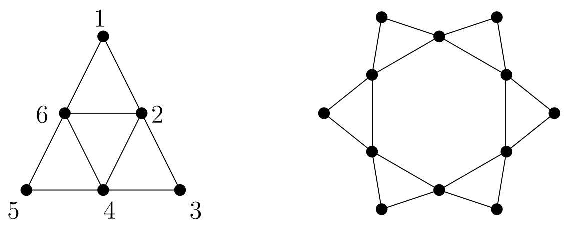


FIG. 1.15 – "propre $\neq$ unitaire" pour les graphes d'arcs.

Les graphes d'arcs et les graphes d'arcs propres peuvent être reconnus en temps et espace linéaire [120, 39] ; lorsque le test est positif, une représentation par arcs (ou par arcs propres) ordonnée du graphe est alors fournie en sortie. Récemment, Durán *et al.* [47] ont proposé un algorithme en temps $O(n^2)$ pour reconnaître les graphes d'arcs unitaires ; la représentation par arcs unitaires que retourne leur algorithme n'est toutefois pas efficace (les extrémités des arcs sont des entiers de taille exponentielle en n).

Le problème de la clique maximum peut être résolu en temps $O(n \log n + m)$ pour les graphes d'arcs [11]. D'un autre côté, le problème de la coloration minimum est $\mathcal{NP}$-difficile [61, 118], même pour les graphes d'arcs qui satisfont la propriété de Helly [71] (en revanche, le problème de la q-coloration devient polynomial lorsque q est un paramètre fixé [61]). Les problèmes du stable maximum et de la partition minimum en cliques sont solubles en temps et espace linéaire [93]. Pour les graphes d'arcs propres, le problème de la clique maximum peut être résolu en temps et espace linéaire [10, 117] et le problème de la coloration minimum en temps $O(n^{1.5})$ [141] (lorsque q est fixé, le problème de la q-coloration est même soluble en temps et espace linéaire [10]).

Considérons à présent le graphe des intersections d'un ensemble de cordes du cercle (et non plus d'arcs) : nous obtenons ce que l'on appelle un *graphe de cordes* (voir la figure 1.16). En général, les graphes de cordes ne sont pas parfaits ; on peut voir par exemple sur la figure 1.16 que l'ensemble de sommets $\{2, 4, 6, 7, 3\}$ induit un cycle sans corde C_5. Il est à noter que la classe des graphes de cordes contient les graphes de permutation ainsi que les graphes d'arcs propres.

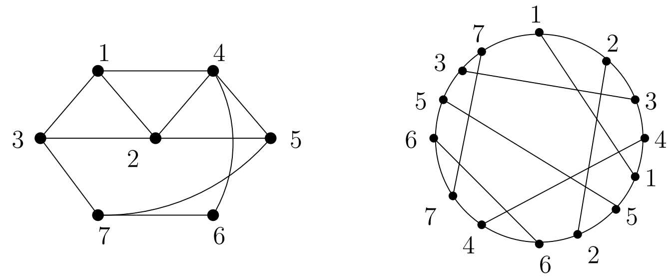


FIG. 1.16 – Un graphe de cordes et sa représentation.

Les graphes de cordes peuvent être reconnus en temps $O(n^2)$ [143] ; toutefois, on ne sait pas construire en temps et espace polynomial une représentation d'un graphe de cordes à l'aide de cordes sur un cercle. Les problèmes de la clique maximum et du stable maximum peuvent être résolus en temps $O(n^2)$, étant donnée en entrée une représentation du graphe de cordes [4]. D'un autre côté, le problème de la q-coloration d'un graphe de cordes est $\mathcal{NP}$-complet [61], même lorsque q est un paramètre fixé supérieur ou égal à quatre [153, 154] (pour $q = 3$, le problème peut être résolu en $O(n \log n)$ [154]). La complexité du problème de la partition minimum en cliques reste à ce jour inconnue.

Notons que le problème de la coloration des graphes d'arcs trouvent des applications dans le routage à travers les réseaux optiques [69, 8] et dans l'allocation des registres en compilation [152, 155]. Une application du problème de la coloration des graphes de cordes concerne le tri de permutations à l'aide du minimum de piles (cf. [75, p. 235–236], voir aussi [105, p. 168]). Pour plus de détails sur ces deux classes de graphes et leurs applications, nous invitons le lecteur à consulter respectivement [75, p. 188–202] et [75, p. 235–253].

Les graphes sans $\mathbf{K_{1,3}}$

Les *graphes sans* $K_{1,3}$ (*i.e.* qui ne possèdent pas de copie induite du graphe $K_{1,3}$) peuvent être reconnus en temps $O(n^{3.5})$ (cf. [49]) ; notons que la reconnaissance par force brute requiert un temps $O(n^4)$. Tout comme les graphes d'arcs et les graphes de cordes, les graphes sans $K_{1,3}$ ne sont en général pas parfaits : le cycle induit C_5 est sans $K_{1,3}$.

Le problème du stable maximum peut être résolu en temps $O(n^4)$ pour les graphes sans $K_{1,3}$ [123, 135] (voir aussi [49]). Néanmoins, les problèmes de la clique maximum, de la coloration minimum et de la partition minimum en cliques restent $\mathcal{NP}$-difficiles (cf. [92] et [49]). Pour les graphes parfaits sans $K_{1,3}$, Hsu et Nemhauser [92] proposent des algorithmes polynomiaux combinatoires pour ces trois problèmes ; Hsu [91] donne notamment un algorithme en temps $O(n^4)$ pour le problème de la coloration minimum.

La sous-classe la plus connue des graphes sans $K_{1,3}$ est la classe des *graphes duaux*. En effet, le problème de la coloration des arêtes d'un graphe, qui a de nombreuses applications dans la construction d'emplois du temps et en ordonnancement [41, 42, 43, 45, 44, 107], revient à colorier les sommets de son dual. Le problème de la clique maximum devient polynomial lorsque l'on se restreint à la classe des graphes duaux (cf. [49]). En dépit du théorème de Vizing (cf. [46, p. 103–105]) (le nombre de couleurs nécessaires à la coloration des arêtes d'un graphe G est soit $\Delta(G)$, soit $\Delta(G)+1$), le problème de coloration minimum d'un graphe dual reste $\mathcal{NP}$-difficile [90]. Notons qu'un résultat similaire tient pour les graphes planaires : colorier un graphe planaire à l'aide de trois couleurs est un problème $\mathcal{NP}$-complet [59], alors que celui-ci admet toujours une 4-coloration (cf. [46, p. 120–122]).

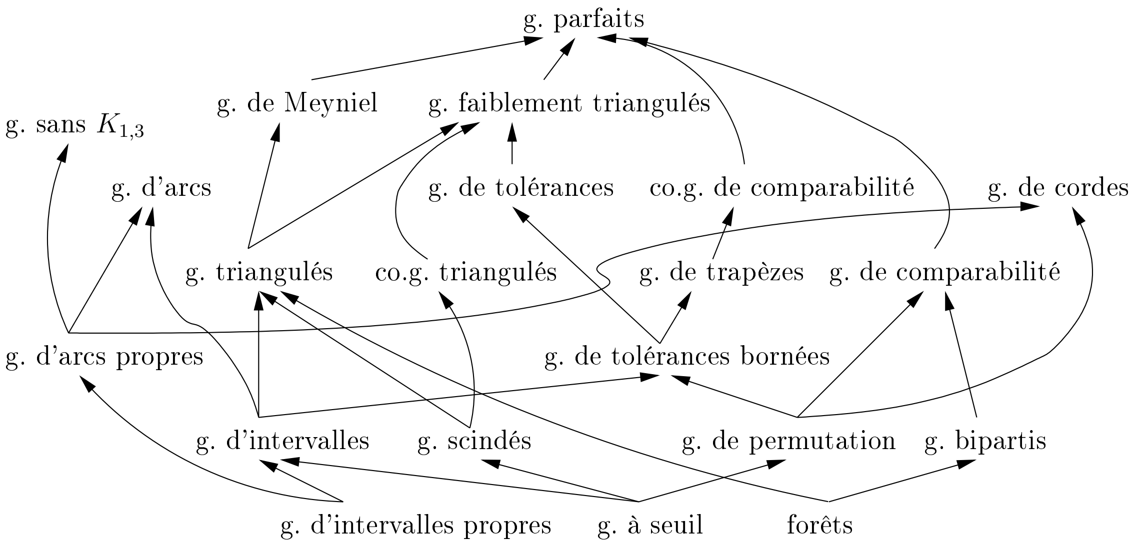


FIG. 1.17 – Une hiérarchie des différentes classes de graphes présentées : « classe A → classe B » signifie « classe A ⊂ classe B » (g. = graphes, co.g. = compléments de graphes).

Remarque. Bien que nous ne l'ayons pas mentionné sur la figure 1.17, les graphes bipartis forment une sous-classe des graphes faiblement triangulés et des graphes de Meyniel. En effet, un graphe biparti ne contient pas de cycle impair de longueur supérieure ou égale à cinq et tout cycle induit impair de longueur supérieure ou égale à cinq dans son complément possède au moins deux cordes.

1.4 État de l'art

Voici pour commencer un résultat basique concernant la complexité du problème d'ordonnancement avec exclusion mutuelle. Celui-ci découle simplement du fait que pour $k = 3$, le problème ORDO est équivalent au problème $\mathcal{NP}$-complet de la partition en triangles [60] dans le complément (lorsque n est un multiple de trois), alors que pour $k = 2$, celui-ci est équivalent au problème du couplage maximum dans ce même complément [48].

Proposition 1.3 (Folklore) *Le problème* Ordo *est $\mathcal{NP}$-difficile, même lorsque k est un paramètre fixé supérieur ou égal à trois. En revanche, il peut être résolu en temps et espace polynomial lorsque $k = 2$.*

De nombreuses améliorations ont été apportées à l'algorithme de couplage maximum proposé par Edmonds [48]. L'algorithme le plus rapide à ce jour est du à Golberg et Karzanov [74] et s'exécute en temps $O(\sqrt{n}\, m \log_n \frac{n^2}{m})$ (voir aussi l'algorithme de Micali et Vazirani [122] en temps $O(\sqrt{n}\, m)$). Toutefois, les structures de données employées par les deux auteurs sont si complexes que leur algorithme n'est que très peu considéré en pratique. Pour le couplage maximum dans les graphes bipartis, l'algorithme en temps $O(\sqrt{n}\, m)$ de Hopcroft et Karp [89] demeure une référence, même si depuis d'autres algorithmes légèrement plus efficaces en temps ont été proposés (voir par exemple [2, 51]). Pour une revue complète sur l'évolution de la complexité des algorithmes de couplage maximum dans les graphes généraux ou bipartis, nous invitons le lecteur à consulter [137, p. 267 et p. 422–423].

Lorsque l'on se restreint à certaines classes de graphes, le problème du couplage maximum peut être plus facile à résoudre. Pour les graphes d'arcs, Liang et Rhee [111] proposent un algorithme de couplage maximum en temps $O(n \log n)$, étant donnée en entrée une représentation par arcs du graphe (les mêmes auteurs affirment dans [129] que leur algorithme peut être implanté en temps $O(n \log \log n)$ en utilisant les structures de [156]) ; la méthode qu'ils proposent est basée sur un algorithme pour le couplage maximum dans les graphes d'intervalles introduit par Moitra et Johnson (cf. [111, 3]). Yu et Yang [158] donnent un algorithme linéaire pour le couplage maximum dans les cographes, et Rhee et Liang [129] un algorithme en temps $O(n \log \log n)$ pour le même problème dans les graphes de permutation, étant donnée en entrée une permutation représentant le graphe. Dahlhaus et Karpinski [36] proposent un algorithme linéaire pour le couplage maximum dans les graphes fortement triangulés, une sous-classe des graphes triangulés qui contient les graphes d'intervalles et les forêts, et montrent que le problème du couplage maximum pour les graphes d'intersection des chemins dans un arbre (et *a fortiori* pour les graphes triangulés) est aussi difficile que pour les graphes bipartis. Enfin, Andrews *et al.* [3] montrent que le même problème peut être résolu en temps $O(n \log n)$ pour les compléments de graphes d'intervalles, étant donnée une représentation par intervalles du graphe en entrée.

Ensuite, les résultats de [59, 90, 61] au sujet de la complexité du problème de la coloration minimum ont comme corollaire immédiat la proposition suivante.

Proposition 1.4 (Folklore) *Le problème* Ordo *est $\mathcal{NP}$-difficile pour les graphes planaires, les graphes duaux, les graphes d'arcs et les graphes de cordes.*

La complexité du problème d'ordonnancement avec exclusion mutuelle n'a été véritablement étudiée que depuis une dizaine d'années. Les premiers résultats parus sur le sujet se rapportaient en fait à des problèmes de décomposition de graphes. Ainsi, un corollaire d'un résultat de Alon [1] est que le problème Ordo peut être résolu en temps et espace polynomial pour les graphes duaux lorsque k est un paramètre fixé. D'un autre côté, Cohen et Tarsi [26] ont établi que le problème est $\mathcal{NP}$-difficile pour les compléments des graphes duaux, même lorsque k est un paramètre fixé supérieur ou égal à trois.

Lonc [113] a étudié le problème Ordo pour les graphes d'ordres partiels, qui est étroitement lié aux problèmes d'ordonnancement avec des contraintes de précédence entre les tâches. Celui-ci établit que le problème reste $\mathcal{NP}$-difficile pour les compléments de graphes de comparabilité même avec $k \geq 3$ fixé, mais peut être résolu en temps et espace linéaire pour les compléments de graphes d'intervalles (il s'appuie pour cela sur un résultat de Papadimitriou et Yannakakis [126]). Il donne aussi un algorithme polynomial pour le problème restreint aux graphes scindés, et montre que pour les cographes, le problème devient polynomial lorsque k est fixé. En conclusion de son article, Lonc [113] pose la question de la complexité du problème Ordo pour les graphes de permutation.

Hansen *et al.* [83] ont consacré un article au sujet, dans lequel ils dérivent des bornes pour $\chi(G, k)$ et présentent quelques cas polynomiaux. En s'appuyant sur un résultat de De Werra [42], les trois auteurs montrent tout d'abord que le problème Ordo peut être résolu en temps et espace polynomial pour les graphes sans $K_{1,3}$, si l'on possède une coloration minimum du graphe en entrée. Ils évoquent ensuite le caractère polynomial du problème lorsque l'on se restreint aux graphes parfaits de stabilité au plus trois (c'est-à-dire tel que $\alpha(G) \leq 3$). Enfin, ils démontrent que le problème est polynomial pour les graphes bipartis lorsque k est fixé et posent la question de la complexité du problème pour les arbres.

Motivés par un problème d'allocation d'opérations à des processeurs, Bodlaender et Jansen [14] ont livré une série de résultats sur le sujet. Pour les cographes et les graphes bipartis, ils montrent que le problème est $\mathcal{NP}$-difficile, mais devient polynomial lorsque k est fixé (pour les graphes bipartis, ils démontrent même que déterminer une partition en trois stables de taille au plus k est un problème $\mathcal{NP}$-complet). Pour les graphes scindés, ils établissent le caractère polynomial du problème, réduisant celui-ci à un problème de flot maximum. Enfin, ils montrent que le problème est $\mathcal{NP}$-difficile pour les graphes d'intervalles, même lorsque k est un paramètre fixé supérieur ou égal à quatre. En conclusion de leur article, Bodlaender et Jansen [14] posent la question de la complexité du problème Ordo pour les graphes d'intervalles lorsque $k = 3$.

Dans le même temps, Baker et Coffman [6] apportaient une réponse positive à la question de Hansen *et al.* [83] en démontrant que le problème Ordo peut être résolu en temps $O(n + k^2 \log k)$ pour les forêts et même $O(n)$ pour les arbres (Jarvis et Zhou [102] ont récemment redécouvert le résultat concernant les arbres). La seconde moitié de leur article est consacrée à une application du problème d'ordonnancement avec exclusion mutuelle pour les graphes planaires : la résolution d'équations aux dérivées partielles en parallèle par décomposition du domaine. Les deux auteurs concluent d'ailleurs leur papier en posant la question de la complexité du problème k-Ordo pour les graphes planaires.

Une analyse polyédrale du problème apparaît dans un papier de De Werra [44], dans laquelle ce dernier établit que la matrice des contraintes du programme linéaire en nombres entiers associé au problème est totalement unimodulaire ou bien équilibrée si et seulement si le graphe en entrée est une union de cliques disjointes (pour plus de détails sur les propriétés de ces matrices, nous invitons le lecteur à consulter [136, p. 266–308]). De fait, le problème peut être résolu en temps et espace polynomial pour cette classe de graphes à l'aide de la méthode de l'ellipsoïde (cf. [119, p. 131–148] ou [136, p. 163–189]). De Werra [44] propose aussi un algorithme combinatoire pour résoudre ce problème (et même une extension de celui-ci), en le réduisant à un problème de flot maximum dans un réseau.

Dahlhaus et Karpinski [36] ont étudié la complexité d'un problème de partition en relation avec le problème Ordo pour les compléments de graphes triangulés ou fortement triangulés (rappelons que la classe des graphes fortement triangulés englobe les graphes d'intervalles et les forêts). Ils montrent notamment que les sommets d'un graphe fortement triangulé peuvent être couverts à l'aide d'un maximum de cliques disjointes de taille k en temps et espace linéaire ; leur algorithme peut être adapté pour résoudre le problème Ordo pour les compléments de graphes fortement triangulés. Ils prouvent aussi qu'une partition d'un graphe triangulé en n/k cliques de taille k (avec n multiple de k) peut être déterminée en temps et espace polynomial s'il en existe une. D'un autre côté, Corneil [29], qui aborde dans son article le problème de la couverture maximum d'un graphe triangulé par des cliques disjointes de taille k, rapporte que D.G. Kirkpatrick aurait montré que le problème est $\mathcal{NP}$-difficile pour $k \geq 3$; toutefois, nous n'avons trouvé aucune référence au sujet de ce résultat dans la littérature.

Nous conclurons cet historique par deux résultats récents. Le premier, du à Jansen [100], répond à la question posée par Lonc [113] : le problème Ordo est $\mathcal{NP}$-difficile pour les graphes de permutation, même lorsque k est un paramètre fixé supérieur ou égal à six. Le second, du à Bodlaender et Fomin [13], établit le caractère polynomial du problème restreint aux graphes de largeur-arbre bornée (de l'anglais *bounded treewidth graphs*) ; Kaller *et al.* [103] avaient auparavant montré que pour ce type de graphes, le problème peut être résolu en temps et espace polynomial si k est fixé en entrée. En outre, Jansen [100] et Bodlaender et Fomin [13] proposent un état de l'art assez complet du problème d'ordonnancement avec exclusion mutuelle.

Pour résumer, les seules classes de graphes pour lesquelles le problème Ordo peut être résolu en temps et espace polynomial sont à ce jour : les graphes scindés [113, 14], les forêts et les arbres [6, 102], les graphes collections de cliques disjointes [44], les compléments de graphes fortement triangulés [36] et de graphes d'intervalles [113, 14], et les graphes de largeur-arbre bornée [13]. En revanche, le problème est $\mathcal{NP}$-difficile pour les graphes duaux [90] et leurs compléments (même avec $k \geq 3$ fixé) [26], les cographes, les graphes bipartis, les graphes d'intervalles (même avec $k \geq 4$ fixé) [14], les compléments de graphes de comparabilité (même avec $k \geq 3$ fixé) [113], et les graphes de permutation (même avec $k \geq 6$ fixé) [100].

1.5 Plan de la thèse

En premier lieu, nous détaillons la complexité du problème d'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles, travail que nous avions entamé dans [62]. Nous donnons un nouvel algorithme linéaire pour le problème lorsque $k = 2$, beaucoup plus simple à mettre à œuvre que celui de Andrews *et al.* [3]. Ensuite, nous étudions quelques problèmes connexes tels que le couplage maximum dans les graphes bipartis convexes ou encore la coloration minimum dans les graphes d'intervalles, pour lesquels nous proposons de nouveaux algorithmes. Enfin, nous montrons que le problème Ordo peut être résolu en temps et espace linéaire pour deux sous-classes des graphes d'intervalles : les graphes d'intervalles propres (qui englobent les graphes collections de cliques disjointes) et les graphes à seuil.

Le problème est aussi étudié pour certaines extensions des graphes d'intervalles comme les graphes d'arcs et les graphes de tolérances. Un algorithme quadratique est présenté pour résoudre le problème restreint aux graphes d'arcs propres, ainsi qu'un algorithme linéaire pour le même problème lorsque $k = 2$. Puis, nous complétons en partie le résultat de Bodlaender et Jansen [14] (voir le théorème 1.1) en montrant que le problème reste $\mathcal{NP}$-difficile pour les graphes de tolérances bornées (et les graphes de Meyniel), même lorsque k est un paramètre supérieur ou égal à trois.

Motivé par des aspects pratiques, nous étudions ensuite l'impact de la propriété suivante sur la complexité du problème : le graphe G d'exclusion mutuelle admet une coloration tel que chaque couleur apparaît au moins k fois. Nous montrons que si un graphe d'intervalles possède cette propriété, alors celui-ci admet une partition optimale en $\lceil n/k \rceil$ stables de taille au plus k. De plus, si une coloration du graphe d'intervalles satisfaisant la propriété en question est donnée en entrée, alors le problème Ordo peut être résolu en temps et espace linéaire. Cette assertion est étendue aux graphes sans $K_{1,3}$, aux graphes d'arcs, aux graphes de tolérances propres (lorsque $k = 2$) et aux graphes triangulés (lorsque $k \leq 4$).

Le dernier chapitre de la thèse est consacré à certains problèmes de partition relatifs aux graphes d'intervalles, qui nous ont été inspirés par l'étude du problème d'ordonnancement avec exclusion mutuelle. Nous nous attardons en particulier sur le problème de la partition d'un graphe d'intervalles en sous-graphes d'intervalles propres, pour lequel nous établissons la proposition suivante : tout graphe d'intervalles admet une partition en moins de $\lceil \log_3 n \rceil$ sous-graphes d'intervalles propres, et celle-ci peut être calculée en temps $O(n \log n + m)$. Ce résultat est mis à profit lors de la conception d'algorithmes polynomiaux d'approximation pour le problème Ordo restreint aux graphes d'intervalles ou d'arcs. Dans une dernière section, nous discutons de trois autres problèmes liés à la partition d'un graphe d'intervalles en sous-graphes d'intervalles propres.

Chapitre 2

Exclusion mutuelle par un graphe d'intervalles

Bodlaender et Jansen [14] ont démontré que le problème d'ordonnancement avec exclusion mutuelle est $\mathcal{NP}$-difficile pour les graphes d'intervalles, même lorsque k est un paramètre fixé supérieur ou égal à quatre. La preuve de ce résultat s'appuie sur une réduction complexe que nous ne détaillerons pas ici, mais dont nous reprendrons les grandes lignes dans le chapitre suivant afin de montrer la difficulté du problème pour les graphes de tolérances bornées lorsque $k = 3$.

L'étude de la complexité du problème pour les graphes d'intervalles lorsque $k = 3$ a été une des motivations premières de ce travail de thèse. Bien que la question demeure ouverte à ce jour, celle-ci nous a amené à reconsidérer le problème pour $k = 2$ et à exhiber plusieurs cas pour lesquels le problème s'avère être soluble en temps et espace polynomial. Les résultats de ces travaux sont exposés dans le présent chapitre [1].

2.1 Un nouvel algorithme linéaire pour k = 2

Comme nous l'avons évoqué dans l'introduction, le problème Ordo peut être résolu en temps et espace polynomial par couplage maximum dans le graphe complément lorsque $k = 2$ (voir la proposition 1.3). Toutefois, les algorithmes de couplage maximum dans un graphe quelconque sont difficiles à mettre en œuvre et leur temps d'exécution est plus que quadratique en le nombre de sommets du graphe [122]. C'est pourquoi la recherche d'algorithmes à la fois simples et efficaces, dédiés à certaines classes de graphes, reste d'actualité.

A notre connaissance, deux algorithmes seulement ont été proposés pour la résolution du problème 2-Ordo restreint à la classe des graphes d'intervalles. Le premier apparaîtrait dans un manuscript de M.G. Andrews et D.T. Lee, encore jamais publié à ce jour. Cet algorithme, qui est brièvement évoqué dans [3], considère une représentation par intervalles en entrée et effectue un balayage du plan pour construire en $O(n \log n)$ une solution optimale, et ce même si les extrémités des intervalles sont déjà triées en entrée. Le second, toujours de type

[1] Une partie de ces résultats apparaît dans [65] (en anglais).

“géométrique”, apparaît dans le papier plus récent d'Andrews *et al.* [3]. Ceux-ci donnent un algorithme récursif parallèle qui requiert un temps $O(\log^3 n)$ sur une architecture PRAM EREW à $O(n/\log^2 n)$ processeurs (se reporter à [28, p. 675–715] pour une introduction à l'algorithmique parallèle). La version séquentielle de leur algorithme s'exécute en temps $O(n \log n)$, mais les auteurs affirment que sa complexité peut être abaissée à $O(n)$ si les extrémités des intervalles sont triées en entrée. Malgré cela, leur algorithme récursif reste difficile à implanter et la preuve de sa validité est longue et fastidieuse.

Dans cette section, nous présentons un nouvel algorithme fondé sur des concepts issus de la théorie des graphes. Notre algorithme est simple, incrémentiel et la preuve de sa correction est courte. Nous montrons que cet algorithme s'exécute en temps et espace $O(n)$ si une représentation par intervalles ordonnée lui est fournie en entrée. Rappelons à ce propos que l'ordre défini par les extrémités gauches (resp. droites) croissantes des intervalles est noté $<_g$ (resp. $<_d$). Comme celui d'Andrews *et al.* [3], notre algorithme utilise comme routine un algorithme de couplage maximum dans les graphes bipartis convexes. Un graphe biparti $B = (X, Y, E)$ est Y*-convexe* s'il existe un ordre linéaire sur les sommets de Y tel que les sommets adjacents à tout sommet de X apparaissent consécutivement dans cette ordre. Nous nous appuierons sur un résultat de Steiner et Yeomans [147] qui établit qu'un couplage maximum dans un graphe biparti convexe peut être calculé en temps $O(|X|)$ et espace $O(|Y|)$ si pour chaque sommet de X, l'intervalle des sommets adjacents à celui-ci dans Y est donné en entrée. Nous reviendrons en détail sur le problème du couplage maximum dans les graphes bipartis convexes dans la section suivante.

Principe et validité de l'algorithme

Nous travaillons ici sur une représentation par intervalles ouverts plutôt que sur le graphe d'intervalles lui-même. Nous conserverons cependant le vocabulaire de la théorie des graphes lorsque nous manipulerons ces intervalles. Ainsi, un stable est un ensemble d'intervalles deux-à-deux disjoints et une clique est un ensemble d'intervalles se chevauchant deux-à-deux. Une coloration d'un ensemble d'intervalles correspond à une partition de cet ensemble en stables. Par analogie au problème de couplage, nous dirons que deux intervalles peuvent être couplés lorsqu'ils sont *disjoints*. Un ensemble d'intervalles pouvant être couplés deux-à-deux sera appelé couplage.

Considérons un ensemble $\mathcal{I} = \{I_1, \ldots, I_n\}$ d'intervalles et une partition minimum $\mathcal{S} = \{S_1, \ldots, S_{\chi(\mathcal{I})}\}$ de $\mathcal{I}$ en stables. Voici les quelques assertions sur lesquelles repose l'algorithme.

Lemme 2.1 *Si un stable $S_u \in \mathcal{S}$ ne contient qu'un seul intervalle, alors ce dernier appartient à toute clique maximum de $\mathcal{I}$.*

Preuve. Comme $\omega(\mathcal{I}) = \chi(\mathcal{I})$, toute clique maximum doit contenir un et un seul intervalle provenant de chaque stable de $\mathcal{S}$. Si l'unique intervalle de S_u n'appartient pas à une clique maximum de $\mathcal{I}$, nous obtenons alors une contradiction. □

Lemme 2.2 (Petit lemme de découpage) *Si chaque stable de $\mathcal{S}$ contient plus de deux intervalles et que n, le nombre d'intervalles dans $\mathcal{I}$, est pair, alors $\chi(\mathcal{I}, 2) = n/2$.*

Preuve. Nous montrons que les stables peuvent être, de manière imagée, “découpés” en exactement $n/2$ paires d'intervalles disjoints.

Soit deux stables $S_u, S_v \in \mathcal{S}$ de taille impaire et supérieure à trois. Nous montrons qu'il est toujours possible de coupler deux intervalles, l'un provenant de S_u et l'autre de S_v, de manière à redéfinir deux nouveaux stables de taille paire et supérieure à deux. Soit $I_a, I_b \in S_u$ et $I_c, I_d \in S_v$ tel que $I_a \prec I_b$ et $I_c \prec I_d$. Si I_a et I_d sont disjoints, alors ceux-ci forment le couplage désiré. Dans le cas contraire, nous affirmons qu'il en est de même pour I_b et I_c. En effet, I_a et I_d se chevauchant, nous avons que $g(I_d) \leq d(I_a)$. En utilisant les inégalités $d(I_c) \leq g(I_d)$ et $d(I_a) \leq g(I_b)$, nous obtenons alors $d(I_c) \leq g(I_b)$.

Pour conclure, la construction suivante établit le lemme. Puisque n est pair, le nombre de stables de taille impaire est nécessairement pair lui aussi. Grâce à la propriété démontrée précédemment, nous pouvons redéfinir deux par deux les stables de taille impaire en stables de taille paire, tout en exhibant des couples d'intervalles disjoints. Finalement, les stables restants, maintenant tous de taille paire, admettent une partition triviale en paires d'intervalles disjoints. □

Proposition 2.1 *Si le nombre $\vartheta(\mathcal{I})$ de stables ne contenant qu'un seul intervalle dans $\mathcal{S}$ est aussi petit que possible, alors l'égalité $\chi(\mathcal{I}, 2) = \lceil (n + \vartheta(\mathcal{I}))/2 \rceil$ est satisfaite.*

Preuve. La proposition est établie à l'aide des deux lemmes précédents. Le premier lemme impose que $\chi(\mathcal{I}, 2) \geq \lceil (n - \vartheta(\mathcal{I}))/2 \rceil + \vartheta(\mathcal{I})$, puisqu'au moins $\vartheta(\mathcal{I})$ intervalles de $\mathcal{I}$ ne peuvent être couplés. Après avoir extrait ces derniers, le second lemme nous permet alors d'obtenir un couplage parfait parmi les $n - \vartheta(\mathcal{I})$ intervalles restants (moins un si ce nombre est impair). □

D'après la proposition précédente, le problème 2-Ordo se réduit donc ici à déterminer une coloration de l'ensemble $\mathcal{I}$ tel que le nombre de stables de taille un soit le plus petit possible. En accord avec le lemme 2.1, ce nouveau problème peut être résolu en calculant un couplage maximum $\mathcal{M}_c$ (d'intervalles disjoints) entre les intervalles d'une clique maximum $C \in \mathcal{I}$ et le reste des intervalles. En effet, une fois ce couplage obtenu, une procédure Compléter-Stables minimisera $\vartheta(\mathcal{I})$ en ajoutant à chaque stable de taille un $S_u = \{I_i\}$ un intervalle $I_j \in S_v$ si le couple (I_i, I_j) appartient à $\mathcal{M}_c$. De là, une solution optimale au problème 2-Ordo s'obtiendra simplement à l'aide de la preuve constructive du petit lemme de découpage. Voici une description des étapes majeures de l'algorithme.

Algorithme 2-Ordo-Intervalles ;
Entrée : un ensemble d'intervalles $\mathcal{I} = \{I_1, \ldots, I_n\}$;
Sortie : une solution optimale $\mathcal{M}$ au problème 2-Ordo pour $\mathcal{I}$;
Début ;
étape 1 :
calculer une coloration minimum $\mathcal{S} = \{S_1, \ldots, S_{\chi(\mathcal{I})}\}$ de $\mathcal{I}$;
si tous les stables de $\mathcal{S}$ ont une taille au plus deux **alors** aller à l'*étape 3* ;
si tous les stables de $\mathcal{S}$ ont une taille au moins deux **alors** aller à l'*étape 3* ;
étape 2 :
calculer une clique maximum $C = \{c_1, \ldots, c_{\chi(\mathcal{I})}\}$ de $\mathcal{I}$;

construire le graphe biparti $B_c = (X, Y, E)$ avec :
. $X = C$ et $Y = \mathcal{I} \setminus C$;
. $E = \{(I_i, I_j) \,|\, I_i \in C, I_j \in \mathcal{I} \setminus C \text{ et } I_i \cap I_j = \emptyset\}$;
calculer un couplage maximum $\mathcal{M}_c$ dans B_c ;
$\mathcal{S} \leftarrow$ Compléter-Stables$(\mathcal{S}, \mathcal{M}_c)$;
étape 3 :
$\mathcal{M} \leftarrow \emptyset$;
pour chaque stable $S_u \in \mathcal{S}$ de taille un **faire**
retirer S_u de $\mathcal{S}$ et l'ajouter à $\mathcal{M}$;
si le nombre d'intervalles restants dans $\mathcal{S}$ est impair **alors**
retirer un intervalle de n'importe quel stable de taille impaire et l'ajouter à $\mathcal{M}$;
calculer un couplage parfait d'intervalles disjoints dans $\mathcal{S}$ et l'ajouter à $\mathcal{M}$;
retourner $\mathcal{M}$;
Fin ;

Dans la partie suivante, nous analysons la complexité de cet algorithme. Nous montrons en particulier qu'il peut s'exécuter en temps et espace linéaire, si la représentation par intervalles fournie en entrée est ordonnée.

Complexité de l'algorithme

Nous admettrons posséder en entrée de l'algorithme une représentation $\mathcal{I} = \{I_1, \ldots, I_n\}$ par intervalles, munie de ses deux ordres $<_g$ et $<_d$. Ceux-ci nous permettront notamment d'obtenir en temps $O(n)$ la liste des n intervalles ordonnés selon $<_g$ ou $<_d$.

La complexité de l'*étape 1* est dominée par la complexité du calcul d'une coloration minimum de $\mathcal{I}$. Ce calcul peut être effectué en temps et espace $O(n)$ lorsque les ordres $<_g$ et $<_d$ sont donnés en entrée [80] (voir aussi l'algorithme Coloration-Intervalles, p. 44). À présent, étudions la complexité de l'*étape 2* qui repose essentiellement sur la propriété suivante.

Lemme 2.3 *Le graphe biparti $B_c = (X, Y, E)$ est Y-convexe.*

Preuve. Rappelons que l'ensemble X correspond à une clique maximum $C \in \mathcal{I}$ et l'ensemble Y au reste des intervalles, c'est-à-dire à $\mathcal{I} \backslash C$. Ce dernier se subdivise en deux sous-ensembles complémentaires $\mathcal{I}_g$ et $\mathcal{I}_d$, respectivement l'ensemble des intervalles se trouvant à gauche de la clique C et l'ensemble des intervalles se trouvant à droite de la clique C (un intervalle ne peut pas appartenir à l'un et à l'autre sans appartenir à la clique). Après avoir ordonné $\mathcal{I}_d$ selon $<_g$ et $\mathcal{I}_g$ selon $<_d$, nous obtenons un ordre $<$ sur Y. À présent, pour chaque $c_u \in C$, posons $a_u = \min\{i \,|\, c_u \prec I_i \text{ et } I_i \in \mathcal{I}_d\}$ et $b_u = \min\{i \,|\, I_i \prec c_u \text{ et } I_i \in \mathcal{I}_g\}$. Il est alors aisé de vérifier que pour tout $i \in \{a_u, \ldots, b_u\}$, les intervalle c_u et I_i sont disjoints (voir la figure 2.1). En conséquence, le graphe biparti B_c est Y-convexe. □

Le graphe biparti B_c étant convexe, le couplage maximum $\mathcal{M}_c$ peut être obtenu en temps $O(n)$ par l'algorithme de Steiner et Yeomans [147]. Leur algorithme demande en entrée la représentation suivante du graphe B_c : l'ordre linéaire sur Y et pour chaque $u \in X$,

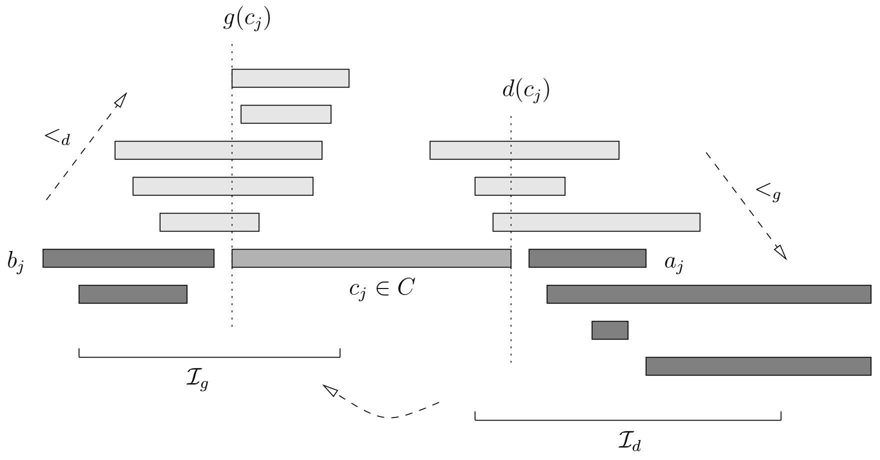


FIG. 2.1 – La preuve du lemme 2.3.

les deux valeurs a_u et b_u. Voici comment nous procédons pour calculer efficacement cette représentation. Une clique maximum peut être exhibée en temps et espace $O(n)$, si les intervalles sont triés [81] (voir aussi l'algorithme CLIQUE-INTERVALLES, p. 45). Cette clique maximum, notée $C = \{c_1, \ldots, c_{\chi(\mathcal{I})}\}$, est définie telle que c_u corresponde à l'intervalle de C appartenant au stable $S_u \in \mathcal{S}$. Ensuite, l'ordre $<$ sur Y (tel que nous l'avons défini dans le lemme précédent) s'obtient en temps $O(n)$ à l'aide des ordres $<_g$ et $<_d$. Enfin, les indices a_u sont déterminés par un balayage linéaire de l'ensemble $\mathcal{I}_d$ ordonné selon $<_g$, à condition que les intervalles de C soient, eux, ordonnés selon $<_d$ (dans ce cas, nous aurons en effet $a_{u-1} \leq a_u$ pour tout $u = 2, \ldots, \chi(\mathcal{I})$). Bien entendu, les indices b_u peuvent être déterminés de façon symétrique en parcourant l'ensemble $\mathcal{I}_g$ en temps linéaire, ce qui complète la construction du graphe biparti B_c.

Pour clore l'analyse de l'*étape 2*, voici une implantation de la procédure COMPLÉTER-STABLES dont le temps d'exécution est linéaire. Nous supposons que la taille de chaque stable peut être obtenue en $O(1)$, de même que pour un intervalle, l'indice du stable auquel il appartient. $\mathcal{M}_c$ peut être considéré comme un tableau dans lequel est stocké en u l'indice de l'intervalle de $\mathcal{I} \setminus C$ couplé à $c_u \in C$ (ou bien zéro si celui-ci n'est pas couplé).

Algorithme COMPLÉTER-STABLES ;
Entrée : une coloration minimum $\mathcal{S}$ de $\mathcal{I}$, un couplage maximum $\mathcal{M}_c$ de B_c ;
Sortie : une coloration minimum $\mathcal{S}$ avec $\vartheta(\mathcal{I})$ minimum ;
Début ;
    $\mathcal{S}' \leftarrow \emptyset$;
    pour chaque stable $S_u \in \mathcal{S}$ **faire**
        si S_u est de taille un **alors** $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{S_u\}$;
    tant que $\mathcal{S}' \neq \emptyset$ **faire**
        $\mathcal{S}' \leftarrow \mathcal{S}' \setminus \{S_u\}$;
        soit i l'indice stocké en u dans $\mathcal{M}_c$;
        si $i \neq 0$ **alors**
            soit v l'indice du stable auquel I_i appartient ;

$S_v \leftarrow S_v \setminus \{I_i\}$, $S_u \leftarrow S_u \cup \{I_i\}$;
si S_v est de taille un **alors** $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{S_v\}$;
retourner $\mathcal{S}$;
Fin ;

Enfin, l'*étape 3* peut elle aussi être implantée en temps $O(n)$. En particulier, la preuve du petit lemme 2.2 de découpage fournit un algorithme linéaire simple pour calculer un couplage parfait d'intervalles disjoints dans $\mathcal{S}$, celui-ci ne contenant alors plus que des stables de taille au moins deux et un nombre pair d'intervalles. L'espace utilisé tout au long de l'algorithme ne dépassant jamais $O(n)$ (y compris durant l'exécution de la procédure de Steiner et Yeomans [147]), nous tenons le résultat suivant.

Théorème 2.1 *L'algorithme* 2-Ordo-Intervalles *retourne en temps et espace $O(n)$ une solution optimale au problème* 2-Ordo *pour un ensemble $\mathcal{I}$ de n intervalles, étant donnés en entrée les ordres $<_g$ et $<_d$ sur $\mathcal{I}$.*

Puisqu'une représentation par intervalles ordonnée (selon $<_d$ ou $<_g$) peut être obtenue en temps et espace linéaire à partir d'un graphe d'intervalles ou de son complément [82], nous obtenons les corollaires suivants.

Corollaire 2.1 *Le problème* Ordo *peut être résolu en temps et espace linéaire pour les graphes d'intervalles lorsque $k = 2$.*

Corollaire 2.2 *Le problème du couplage maximum peut être résolu en temps et espace linéaire pour les compléments de graphes d'intervalles.*

2.2 Problèmes relatifs

La résolution du problème 2-Ordo pour les intervalles est essentiellement basée sur deux routines : le calcul d'une coloration minimum (et d'une clique maximum) d'un ensemble d'intervalles et le calcul d'un couplage maximum dans un graphe biparti convexe. Ces deux routines sont au cœur de l'algorithme 2-Ordo-Intervalles. Par conséquent, l'implantation de celles-ci aura une influence certaine sur la complexité de l'algorithme tout entier. Dans cette section, nous nous proposons d'étudier dans le détail ces deux problèmes que sont la coloration minimum d'un ensemble d'intervalles et le couplage maximum dans un graphe biparti convexe, en commençant par le second.

2.2.1 Du couplage maximum dans les graphes bipartis convexes

Le problème du couplage maximum dans un graphe consiste, rappelons-le, à déterminer un sous-ensemble d'arêtes de cardinal maximum tel que deux de ces arêtes ne soient pas incidentes au même sommet. Du fait de ses nombreuses applications en optimisation combinatoire et recherche opérationnelle, ce problème a été particulièrement étudié pour les graphes bipartis. Soit $B = (X, Y, E)$ un graphe biparti. Exceptionnellement, nous noterons ici n le nombre de sommets de X et m le nombre de sommets de Y. La notation ensembliste $|E|$ sera utilisée pour dénoter le nombre d'arêtes du graphe.

L'algorithme en temps $O(\sqrt{n+m}\,|E|)$ de Hopcroft et Karp [89] reste à ce jour un des plus rapides pour résoudre le problème du couplage maximum dans les graphes bipartis. Comme nous l'évoquions précédemment, le temps d'exécution d'un tel algorithme, bien que polynomial, n'est plus satisfaisant lorsque le nombre de sommets du graphe devient très grand. Ainsi, plusieurs algorithmes ont été proposés pour traiter des instances particulières, comme par exemple les graphes bipartis réguliers (voir [137, p. 267–274]). Les *graphes bipartis convexes*, qui apparaissent dans plusieurs applications industrielles [73, 37, 145], en sont un autre exemple important. Plusieurs algorithmes ont été donnés pour résoudre le problème du couplage maximum dans les graphes bipartis convexes [73, 112, 58, 37, 139, 147], mais tous utilisent des structures de données complexes à mettre en œuvre. Après une revue des différents résultats de la littérature sur le sujet, nous présentons un nouvel algorithme en temps $O(n \log n)$ et espace $O(n)$, dont l'implantation ne pose aucune difficulté. Nous verrons ensuite comment traiter le problème de manière encore plus efficace pour les graphes bipartis *doublement convexes* ou *proprement convexes*, que nous définirons par la suite.

Comme dans toute la littérature traitant du sujet, nous admettrons que le graphe biparti convexe $B = (X, Y, E)$ est toujours spécifié par sa *représentation convexe*, c'est-à-dire avec $X = \{1, \ldots, n\}$ et $Y = \{1, \ldots, m\}$, et pour chaque $i \in X$, un couple d'entiers (a_i, b_i) définissant un intervalle de Y.

État de l'art

Les graphes bipartis convexes ont été introduits en 1967 par Glover [73]. Dans son papier, il décrit une application industrielle du problème du couplage maximum dans les graphes bipartis convexes et fournit un algorithme glouton[2] en $O(|E|)$ pour le résoudre. Cet algorithme progresse de manière séquentielle à travers l'ensemble Y et pour chaque sommet $y \in Y$, cherche parmi tous les sommets non couplés de X adjacents à y celui ayant le plus petit indice b_i pour le coupler à celui-ci. Glover introduit aussi les graphes bipartis doublement convexes, c'est-à-dire à la fois convexes sur X et sur Y. Une fois les ensembles X et Y d'un graphe biparti doublement convexe ordonnés, Glover remarque que pour chaque $y \in Y$, le sommet de X candidat au couplage avec y est celui qui possède soit le plus petit indice, soit le plus grand, parmi les sommets de X non encore couplés et adjacents à y.

Plus tard, Lipski et Preparata [112] proposèrent une modification de l'heuristique de Glover s'exécutant en temps $O(n\,\alpha(n) + m)$, où $\alpha(n)$ est une fonction relative à l'inverse de la fonction d'Ackermann qui croît très lentement (cf. [149, p. 23–26] ou [28, p. 442–445] pour plus de détails). Leur algorithme est basé sur une idée suggérée par J.E. Hopcroft, très proche de celle employée pour résoudre efficacement le problème du minimum différé (cf. [28, p. 451] pour plus de détails). Le temps d'exécution presque linéaire de l'algorithme est la conséquence de l'utilisation de la structure de données pour ensembles disjoints proposée par Tarjan [148]. D'après les résultats plus récents de Gabow et Tarjan [57], celui-ci s'avère être linéaire en $O(n + m)$. Enfin, Lipski et Preparata donnent dans leur papier une version

[2] « This rather suggestive terminology is due to Jack Edmonds. », précise Glover !

efficace de l'algorithme de Glover pour le cas doublement convexe. A l'aide d'une file de priorité à double entrée, ils obtiennent un temps d'exécution en $O(n+m)$.

D'un autre côté, Gallo [58] donnait une implantation en temps $O(n \log n)$ d'une modification similaire. Pour obtenir ce temps d'exécution, Gallo utilise une structure de données particulière à deux niveaux de tas, à savoir un tas binaire dont chaque noeud est un tas à gauche (de anglais *leftist heap*, cf. [105, p. 150–152] ou [149, p. 33–43] pour plus de détails). Scutella et Scevola [139] ont donné une procédure qui s'exécute dans le temps minimum pris par l'algorithme de Lipski–Preparata et celui de Gallo.

Dekel et Sahni [37] ont été les premiers à proposer un algorithme parallèle pour résoudre le problème. Leur algorithme prend un temps $O(\log^2 n)$ sur $O(n)$ processeurs PRAM EREW (se reporter à [28, p. 675–715] pour une introduction à l'algorithmique parallèle). Atallah *et al.* [5] obtiennent un temps d'exécution en $O(\log n)$ à l'aide de $O(n^3)$ processeurs PRAM EREW. Récemment, Andrews *et al.* [3] ont abaissé le nombre de processeurs à $O(n/\log n)$ utilisés par l'algorithme de Dekel et Sahni. En effet, le nombre total d'opérations (et donc le temps d'exécution) de la version séquentielle de leur algorithme peut être ramené à $O(n \log n)$.

L'ultime amélioration apportée à l'heuristique de Glover est due à Steiner et Yeomans [147]. Ces derniers ont appliqué une méthode semblable à celle décrite par Frederickson [55, 146] pour résoudre un problème d'ordonnancement de tâches avec temps d'exécution unitaires et dates au plus tôt et au plus tard entières. Après une première étape permettant d'éviter un tri explicite des sommets de l'ensemble Y à l'aide de $O(n+m)$ espace mémoire supplémentaire, le graphe est partagé en temps $O(n)$ en composantes distinctes sur Y qui peuvent être traitées séparément par l'algorithme de Glover. Le point crucial de la méthode est que lors de ce partage, le nombre de sommets de Y appartenant à l'ensemble des composantes est réduit à n, les autres sommets ne s'avérant pas nécessaires au couplage. Enfin, un couplage maximum est extrait dans chacune des composantes en exécutant une séquence d'au plus $2z$ opérations pour un problème de minimum différé (cf. [28, p. 451] pour plus de détails), où z est le nombre de sommets de la composante appartenant à Y. D'après Gabow et Tarjan [57], l'exécution d'une telle séquence d'opérations ne requiert qu'un temps $O(z)$ dans le pire des cas. En sommant sur toutes les composantes, nous obtenons alors un temps total d'exécution en $O(n)$.

Enfin, citons deux extensions intéressantes des graphes bipartis convexes. Tout d'abord, les graphes bipartis *convexes sur un arbre orienté*, introduits par Lipski et Preparata [112], sont tels les sommets de Y adjacents à un sommet de X forment des chemins sur un arbre orienté. Les deux auteurs montrent qu'après avoir effectué un tri topologique des sommets de Y (des sommets les plus proches de la racine aux sommets les plus éloignés), une modification de l'algorithme de Glover similaire à celle qu'ils proposent pour le cas convexe permet d'obtenir un temps d'exécution en $O(n+m)$. Ensuite, les graphes bipartis *convexes sur un cercle*, introduits par Liang et Blum [110], sont tels que les sommets de Y adjacents à un sommet de X forment des arcs autour d'un cercle. Ces derniers ont montré que deux passes de l'heuristique de Glover sur des sous-graphes appropriés suffisaient à extraire un couplage maximum. En s'appuyant sur la procédure de Lipski et Preparata, ils obtiennent ainsi un temps total d'exécution en $O(n+m)$.

Un nouvel algorithme à la fois efficace et pratique

Nous présentons une nouvelle modification de l'heuristique de Glover [73] inspirée du travail de Gallo [58]. Nous montrerons ensuite comment implanter cet algorithme à l'aide d'un simple *tas binaire* et d'une structure à base de tableaux, afin que celui-ci puisse s'exécuter en temps $O(n \log n)$ et espace $O(n)$.

Modification de l'heuristique de Glover :

Étape 1. Poser $\mathcal{M} = \emptyset$, $\mathcal{Q} = X$ et pour chaque $i \in X$, $d_i = a_i$. $\mathcal{M}$ dénote l'ensemble des arêtes appartenant au couplage ; $\mathcal{Q}$ dénote l'ensemble des sommets non couplés de X ; d_i, lorsque sa valeur est inférieure ou égale à b_i, représente le premier sommet non couplé de Y pour lequel une arête (i, d_i) existe.

Étape 2. Soit $u \in \mathcal{Q}$ tel que, pour chaque $i \in \mathcal{Q}$, $d_u < d_i$ ou $b_u \leq b_i$ si $d_u = d_i$. Poser $\mathcal{Q} = \mathcal{Q} \setminus \{u\}$. Si $d_u \leq b_u$, alors poser $\mathcal{M} = \mathcal{M} \cup \{(u, d_u)\}$. Pour chaque $i \in \mathcal{Q}$ tel que $d_i = d_u$, poser $d_i = d_i + 1$.

Étape 3. Si $\mathcal{Q} \neq \emptyset$, alors aller à l'*Étape 2*, sinon retourner $\mathcal{M}$.

La différence majeure entre la modification proposée par Gallo [58] et celle décrite ci-dessus réside dans le fait que le sommet de X candidat au couplage à l'*étape 2* est celui qui possède la valeur de d_i minimum (et non pas, comme chez Gallo, la valeur de b_i minimum). De cette façon, les sommets de Y sont sélectionnés *dans l'ordre* pour être couplés avec les sommets de X. La validité de l'algorithme peut être établie à l'aide des mêmes techniques de preuve que celles développées dans [73, 112, 58]. Nous reprenons ici celle de Gallo [58] qui utilise le célèbre théorème de Berge (cf. [149, p. 114–115]) caractérisant le couplage maximum dans un graphe : un couplage $\mathcal{M}$ dans un graphe est maximum si et seulement s'il n'admet aucun chemin améliorant. Un *chemin améliorant* est une chaîne composée d'arêtes alternativement dans $\mathcal{M}$ et hors de $\mathcal{M}$ qui connecte deux sommets *exposés*, c'est-à-dire deux sommets n'appartenant à aucune arête du couplage $\mathcal{M}$.

Proposition 2.2 *La modification de l'heuristique de Glover retourne un couplage maximum dans le graphe biparti convexe $B = (X, Y, E)$.*

Preuve. Tout d'abord, il est facile de voir que $\mathcal{M}$ est un couplage, puisque après l'ajout de l'arête (u, d_u) les sommets u et d_u ne peuvent plus être considérés par l'algorithme. À présent, nous montrons que ce couplage est maximum. Pour cela, nous dirons qu'un chemin améliorant est de longueur k s'il contient k arêtes qui n'appartiennent pas à $\mathcal{M}$. Clairement, il n'existe aucun chemin améliorant de longueur un. En effet, un tel chemin n'est composé que d'une seule arête reliant deux sommets exposés $x \in X$ et $y \in Y$. Or, l'algorithme sélectionne chaque sommet de X et le couple à un sommet non couplé de son voisinage (s'il en existe au moins un). Par conséquent, un chemin de longueur un ne peut exister.

Pour conclure, nous montrons que s'il n'existe pas de chemin améliorant de longueur $k \geq 1$, alors il n'existe pas non plus de chemin améliorant de longueur $k + 1$.

Soit $(x_0, y_1, x_1, y_2, x_2, \ldots, y_k, x_k, y_{k+1})$ un chemin améliorant de longueur $k+1$, avec $(x_i, y_i) \in \mathcal{M}$ pour tout $i = 1, \ldots, k$ et $(x_i, y_{i+1}) \in E \setminus \mathcal{M}$ pour tout $i = 0, \ldots, k$. Notons (x_j, y_j) la première arête dans ce chemin à avoir été ajoutée au couplage $\mathcal{M}$. Nous affirmons maintenant que $y_j < y_{j+1}$ et $b_{x_j} \leq b_{x_{j-1}}$. En effet, si la première inégalité n'est pas vérifiée, alors d'après l'algorithme, le sommet x_j doit être couplé au sommet y_{j+1}, ce qui est impossible en accord avec l'hypothèse précédente. Si la seconde inégalité n'est pas vérifiée, nous obtenons d'après l'algorithme que y_j doit être couplé à x_{j-1}, ce qui de nouveau est impossible. De ces inégalités nous pouvons déduire que $a_{x_{j-1}} \leq y_j < y_{j+1} \leq b_{x_j} \leq b_{x_{j-1}}$ et, par convexité, que $(x_{j-1}, y_{j+1}) \in E$. Par conséquent, nous obtenons que $(x_0, y_1, x_1, \ldots, y_{j-1}, x_{j-1}, y_{j+1}, x_{j+1}, \ldots, y_k, x_k, y_{k+1})$ est un chemin de longueur k, ce qui contredit l'hypothèse première. $\square$

Nous allons maintenant nous intéresser à l'implantation de la modification de l'heuristique de Glover décrite ci-dessus. Les structures de données que nous utilisons sont simples, bien que les interactions entre celles-ci puissent paraître compliquées au premier abord.

Soit $\ell_1 < \ell_2 < \cdots < \ell_r$ les valeurs distinctes prises par les a_i ($1 \leq i \leq n$ et $r \leq n$). Comme dans [58], l'ensemble $\mathcal{Q}$ défini lors de l'*étape 1* est subdivisé en r sous-ensembles $Q_1, Q_2, \ldots, Q_r$ tel que $Q_k = \{i \in X \mid a_i = \ell_k\}$. Les sommets de chaque sous-ensemble Q_k sont ordonnés selon les b_i croissants. Désormais, l'indice d_i n'est plus défini pour chaque $i \in X$, mais pour chaque $Q_i \in \mathcal{Q}$. Ce travail d'initialisation peut être effectué à l'aide d'un simple tri, les a_i et b_i étant stockés en entrée dans des tableaux de dimension n.

Dans son implantation, Gallo [58] réalise l'*étape 2* de la manière suivante. À chaque fois qu'un sommet $d_k \in Y$ est couplé à un sommet de X, l'indice d_k est mis à jour. Si le nouvel indice se trouve être égal à l'indice d_t d'un sous-ensemble Q_t pour $t > k$, alors les deux sous-ensembles Q_t et Q_k sont fusionnés en $Q_t = Q_t \cup Q_k$. Pour réaliser cette opération en temps $O(\log n)$, Gallo implante chaque sous-ensemble $Q_k \in Q$ comme un tas à gauche (cf. [105, p. 150–152] ou [149, p. 33–43] pour plus de détails). L'implantation que nous proposons ici permet d'éviter l'emploi d'une telle structure de données grâce à l'utilisation d'une simple *file de priorité* pour stocker l'ensemble C des sommets de $\mathcal{Q}$ candidats au couplage. Nous rappelons qu'une file de priorité est une structure de données permettant de gérer un ensemble C d'éléments, chacun ayant une valeur associée appelée clé. Une file de priorité supporte les trois opérations suivantes : Insérer(C, x) qui insère l'élément x dans l'ensemble C, Minimum(C) qui retourne l'élément de C ayant la plus petite clé et Extraire-Min(C) qui supprime puis retourne l'élément de C ayant la plus petite clé. Pour plus de détails, le lecteur est renvoyé à [28, p. 146–148].

Voici comment est utilisée la file de priorité C. En accord avec la modification de l'heuristique de Glover proposée précédemment, les sommets de Y sont balayés dans l'ordre en vue d'un couplage. Notons y le premier sommet non couplé de Y et k le plus grand indice tel que $\ell_k \leq y$. Les sommets de X candidats au couplage avec le sommet y appartiennent de fait à l'ensemble $Q_1 \cup \cdots \cup Q_k$. Parmi tous les sommets de ces k sous-ensembles, le but est donc de trouver celui qui a le plus petit indice b_i. L'idée est alors de réduire l'espace de recherche de ce sommet à l'ensemble C des premiers sommets non couplés de chaque sous-ensemble Q_j ($1 \leq j \leq k$). En effet, les sommets de chaque sous-ensemble Q_j étant rangés selon les b_i croissants, le premier sommet non couplé correspond exactement au sommet non couplé de plus petit indice b_i. En implantant l'ensemble C comme une file de priorité avec pour clé l'indice b_i, le sommet recherché, noté x, peut s'obtenir par une simple extraction de l'élément minimum de la file. Après son extraction, x sera couplé à $y \in Y$ à condition que $y \leq b_x$. Si tel est le cas, le couple (x, y) est ajouté à l'ensemble $\mathcal{M}$ et l'on passe au prochain sommet de Y à coupler (soit $y \leftarrow y+1$). Enfin, la file C de priorité doit être mise à jour. Si le sommet x que l'on vient d'extraire appartient à un sous-ensemble Q_u $(1 \leq u \leq k)$ dans lequel il reste des sommets non couplés, alors il faut insérer le premier d'entre eux (celui de plus petit indice b_i) dans la file C. En effet, les sommets de Q_u sont nécessairement des candidats au couplage avec le nouveau sommet y puisque l'inégalité $\ell_u \leq \ell_k \leq y$ reste valide (elle le restera d'ailleurs jusqu'à la fin de l'algorithme). Il faut aussi vérifier si des sommets provenant des sous-ensembles $Q_{k+1}, \ldots, Q_r$ peuvent être candidats au couplage avec le nouveau sommet y. En fait, les sous-ensembles $Q_1, \ldots, Q_r$ étant ordonnés $(\ell_1 < \cdots < \ell_r)$, le seul ensemble pouvant contenir de nouveaux candidats au couplage sera Q_{k+1} si $y = \ell_{k+1}$.

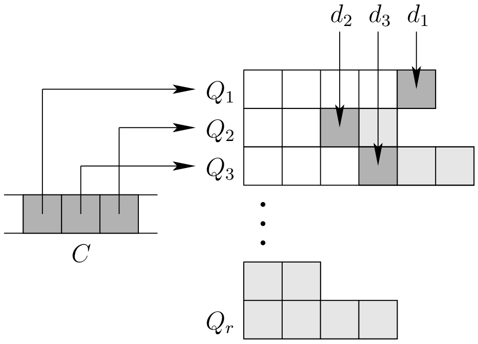


FIG. 2.2 – La file de priorité C couplée à la structure Q.

Concrètement, chaque sous-ensemble Q_j $(1 \leq j \leq r)$ est implanté par un tableau. Les l_j correspondants sont rangés dans un tableau noté ℓ. Par souci d'efficacité, nous ne stockons dans la file C que l'indice j du sous-ensemble Q_j auquel un sommet appartient $(1 \leq j \leq r)$. Afin de retrouver le sommet en question, d_j pointe sur le premier sommet non-couplé de Q_j. De cette façon, le sommet correspondant à un élément j de la file, ainsi que sa clé b_i, peuvent être obtenus en temps $O(1)$ *via* d_j. Un aperçu des structures de données utilisées et de leurs interactions est donné figure 2.2. La procédure complète est détaillée ci-après. Par concision, nous utilisons les routines et notations suivantes : taille(Q_j) renvoie la taille du tableau Q_j et candidat(Q_j) retourne le premier sommet non couplé de Q_j, c'est-à-dire le sommet d'indice d_j dans Q_j. Ainsi, la clé d'un élément j de la file de priorité correspond à la valeur de b_i pour $i = $ candidat(Q_j). Notons que les instructions relatives à la récupération de la clé sont considérées comme implantées au cœur des opérations INSÉRER, MINIMUM et EXTRAIRE-MIN. Enfin, les notations ensemblistes $C = \emptyset$ et $C \neq \emptyset$ sont utilisées pour tester si la file C est vide ou pas.

Algorithme COUPLAGE-BIPARTI-CONVEXE
Entrée : une représentation convexe de $B = (X, Y, E)$;
Sortie : un couplage $\mathcal{M}$ maximum dans B ;
Début ;
    initialiser ℓ, $\mathcal{Q}$, d, $\mathcal{M}$;

soit C une file de priorité vide ;
$k \leftarrow 1$, $y \leftarrow \ell_1$, INSÉRER$(C, 1)$;
tant que $C \neq \emptyset$ **faire**
  $u \leftarrow$ EXTRAIRE-MIN(C), $x \leftarrow$ candidat(Q_u) ;
  si $y \leq b_x$ **alors** $\mathcal{M} \leftarrow \mathcal{M} \cup \{(x, y)\}$, $y \leftarrow y + 1$;
  si $d_u <$ taille(Q_u) **alors** $d_u \leftarrow d_u + 1$, INSÉRER(C, u) ;
  si $k < r$ et $(C = \emptyset$ ou $\ell_{k+1} = y)$ **alors** $k \leftarrow k + 1$, $y \leftarrow \ell_k$, INSÉRER(C, k) ;
retourner $\mathcal{M}$;
Fin ;

Pour conclure, analysons la complexité de la procédure COUPLAGE-BIPARTI-CONVEXE. L'initialisation des différentes structures peut se faire en temps $O(n \log n)$, notamment après avoir trié les a_i et b_i. L'espace requis par les différentes structures n'excède pas $O(n)$. En particulier, la file de priorité ne contiendra pas plus de $r \leq n$ éléments (au plus un par ensemble $Q_j \in \mathcal{Q}$). Ensuite, le nombre de passages dans la boucle **tant que** est clairement limité à n. En effet, à chaque tour de boucle un élément de la file C est extrait ; d'un autre côté, le nombre d'éléments insérés dans la file ne dépasse pas n, le cardinal de X. Ainsi, la complexité en temps de la procédure sera donnée par l'expression $O(n \log n) + n \times (E(n) + I(n) + O(1))$ avec $E(n)$ et $I(n)$ les temps consommés respectivement par un appel à EXTRAIRE-MIN et un appel à INSÉRER (les routines taille(Q_j) et candidat(Q_j) peuvent s'exécuter en temps $O(1)$, de même que tester si la file est vide ou non).

Une file de priorité peut être implantée simplement à l'aide d'une structure de *tas binaire* [28, p. 138–148]. Un tas binaire peut supporter en temps $O(\log r)$ n'importe laquelle des trois opérations INSÉRER, MINIMUM et EXTRAIRE-MIN, si la file de priorité contient au plus r éléments. Dans ce cas, nous aurons $E(n) = I(n) = O(\log n)$ et aussitôt, la proposition suivante.

Proposition 2.3 *Implanté à l'aide d'un tas binaire, l'algorithme* COUPLAGE-BIPARTI-CONVEXE *détermine un couplage maximum dans un graphe biparti convexe $B = (X, Y, E)$ en temps $O(n \log n)$ et espace $O(n)$, étant donnée en entrée une représentation convexe de B.*

Remarque. Si $r = O(1)$, alors le temps d'exécution de la boucle **tant que** est en $O(n)$, puisque $E(n) = I(n) = O(1)$. D'un autre côté, si $r = n$, alors chaque $Q_j \in \mathcal{Q}$ est de taille un $(1 \leq j \leq n)$. Dans ce cas, la file ne contient jamais plus d'un élément à la fois et le temps d'exécution de la boucle **tant que** reste en $O(n)$.

L'implantation d'un tas binaire peut se faire à l'aide d'un tableau (de longueur r si celui-ci doit contenir au plus r éléments). Les primitives d'accès aux éléments du tas (qui peut être vu comme un arbre binaire presque complet) peuvent alors être réalisées par simples décalages de bits. Une telle implantation s'avèrera donc très efficace en pratique, en comparaison avec les implantations des procédures de Lipski-Preparata [112] et Steiner-Yeomans [147] basées sur des structures de données pour ensembles disjoints représentées par des arborescences et manipulées à l'aide de pointeurs (se reporter à [149, p. 23–31] ou [28, p. 439–450] pour plus de détails). De la même manière, on préférera sans doute l'utilisation de cette structure de tas binaire à la structure à deux niveaux de tas préconisée par Gallo [58], qui nécessite notamment l'utilisation d'une structure de tas plus complexe supportant l'opération de fusion en temps $O(\log n)$ comme par exemple les tas à gauche, les tas binomiaux ou encore les tas de Fibonacci. Pour plus de détails sur ces différentes structures de tas, le lecteur est renvoyé à [149, p. 33–43] et [28, p. 392–431].

Remarque. Une fois implantée, la structure de tas binaire peut aussi servir de base à un algorithme de tri par tas, utile lors de l'initialisation de la structure de données $\mathcal{Q}$.

D'un autre côté, l'implantation de la file de priorité peut tout aussi bien être réalisée à l'aide des structures de Gabow et Tarjan [57] (théoriquement plus performantes) et ainsi atteindre le même temps d'exécution que la procédure de Lipski–Preparata [112]. En effet, il est possible d'affecter une clé entière entre 1 et n à chaque b_i $(1 \leq i \leq n)$ de manière à ce que la clé d'un indice b_i corresponde à son numéro dans l'ordre $b_1 \leq \cdots \leq b_n$ (lorsque plusieurs b_i ont la même valeur, leurs clés peuvent être rangées dans n'importe quel ordre). Après avoir trié les a_i et b_i (cette fois-ci en $O(n+m)$ à l'aide d'un tri par paquets), nous pouvons établir la correspondance entre les b_i et leur clé en temps $O(1)$ à l'aide de deux tableaux auxiliaires. L'univers des clés de la file de priorité peut ainsi être ramené à l'ensemble $\{1, \ldots, n\}$ et permettre l'utilisation de la structure pour ensembles disjoints de Gabow et Tarjan [57], qui garantit un temps amorti $O(1)$ par opération Insérer ou Extraire-Min. La complexité totale en temps de la procédure sera alors en $O(n + m)$.

Proposition 2.4 *Implanté à l'aide de la structure pour ensembles disjoints de Gabow et Tarjan, l'algorithme* Couplage-Biparti-Convexe *détermine un couplage maximum dans un graphe biparti convexe $B = (X, Y, E)$ en temps $O(n + m)$ et espace $O(n)$, étant donnée en entrée une représentation convexe de B.*

Remarque. Il est aussi possible d'utiliser la structure pour ensembles disjoints de Tarjan [148] moins complexe à mettre en œuvre et garantissant un temps amorti $O(\alpha(n))$ par opération, où $\alpha(n)$ est une fonction relative à l'inverse de la fonction d'Ackermann qui croît très lentement (cf. [149, p. 25–26] ou [28, p. 442–445] pour plus de détails), ou encore la structure de Van Emde Boas [156] qui assure dans le pire des cas un temps $O(\log \log n)$ par opération.

Dans le cas où les indices a_i et b_i sont déjà triés en entrée, l'initialisation des structures ne prend plus qu'un temps $O(n)$ et nous obtenons le résultat suivant.

Corollaire 2.3 *Implanté à l'aide de la structure pour ensembles disjoints de Gabow et Tarjan, l'algorithme* Couplage-Biparti-Convexe *détermine un couplage maximum dans un graphe biparti convexe $B = (X, Y, E)$ en temps et espace $O(n)$, étant donnée en entrée une représentation convexe ordonnée de B.*

Remarque. L'algorithme de Steiner et Yeomans [147] atteint un temps $O(n)$ même lorsque la représentation convexe n'est pas ordonnée en entrée. Pour cela, ils utilisent un espace auxiliaire de taille $O(n + m)$; cet espace peut toutefois être ramené en $O(n + m^{1/c})$, pour n'importe quel entier positif c, à l'aide d'une technique similaire à celle que décrit Frederickson [55] dans son papier.

Une nouvelle implantation pour le cas doublement convexe

Dans leur papier, Lipski et Preparata [112] proposent une version de l'heuristique de Glover dédiée au problème du couplage maximum dans les graphes bipartis doublement convexes. Un graphe biparti $B = (X, Y, E)$ est dit *doublement convexe* lorsqu'il est à la fois X-convexe et Y-convexe. Nous noterons $<_x$ (resp. $<_y$) l'ordre témoin de la convexité sur X (resp. Y). Les graphes bipartis doublement convexes se caractérisent aisément à l'aide de la propriété des uns consécutifs.

Lemme 2.4 *Un graphe biparti est doublement convexe si et seulement si sa matrice d'incidence sommets-vs-sommets possède la propriété des uns consécutifs à la fois pour les lignes et pour les colonnes.*

Ces graphes bipartis possèdent donc une représentation intéressante lorsque X et Y sont respectivement ordonnés selon $<_x$ et $<_y$, que nous appellerons représentation *doublement convexe* (voir la figure 2.3).

	y_1	y_2	y_3	y_4	y_5	y_6	y_7
x_1		1	1	1	1		
x_2	1	1	1	1	1	1	1
x_3		1	1	1	1	1	1
x_4			1	1	1	1	
x_5				1	1	1	

FIG. 2.3 – La matrice sommets-*vs*-sommets d'un graphe biparti doublement convexe.

Comme l'avait remarqué Glover [73], le problème du couplage maximum devient plus simple lorsque l'on se restreint aux graphes bipartis doublement convexes. Rappelons que l'heuristique de Glover balaye les sommets de Y dans l'ordre et couple à un sommet $y \in Y$ le sommet qui a le plus petit indice b_i parmi tous les sommets de X adjacents à y et encore non couplés. Lorsque ceux-ci sont rangés dans l'ordre $<_x$, alors le sommet candidat au couplage avec y est soit le premier dans cet ordre, soit le dernier. En effet, en supposant le contraire (c'est-à-dire l'existence d'un autre sommet ayant un indice b_i strictement plus petit), nous obtenons qu'au moins une des colonnes de la matrice sommets-*vs*-sommets n'a pas la propriété des uns consécutifs, ce qui contredit l'hypothèse de double convexité.

S'appuyant sur cette remarque, Lipski et Preparata [112] proposent d'implanter l'ensemble C comme une *file à double entrée*. Ce type de file permet l'insertion et la suppression d'un élément à chaque bout de la file [28, p. 198–199]. Ils montrent ainsi que l'heuristique de Glover peut s'exécuter en temps $O(n+m)$, à condition qu'une représentation doublement convexe du graphe biparti soit fournie en entrée (notamment les ordres $<_x$ et $<_y$) et que celui-ci soit connexe. Ils donnent en outre une méthode pour déterminer en temps $O(n+m)$ les composantes connexes d'un graphe biparti convexe puis un algorithme en temps $O(n)$ pour tester si un graphe biparti convexe et connexe est doublement convexe (si tel est le cas, leur algorithme renvoie une représentation doublement convexe), étant donnée en entrée une représentation convexe ordonnée.

La modification de l'heuristique de Glover que nous avons décrite précédemment peut elle aussi être implantée de manière plus efficace lorsque l'on se restreint au cas doublement convexe. Nous parvenons même à améliorer le résultat de Lipski et Preparata [112], en montrant que le problème du couplage maximum pour un graphe biparti doublement convexe et connexe peut être résolu en temps et espace $O(n)$, étant donnée en entrée une représentation doublement convexe de ce graphe. Avant de donner le détail de cette implantation, nous allons en donner les grandes lignes.

Les sommets de Y sont balayés dans l'ordre $<_y$ pour être couplés. Soit $y \in Y$ le premier sommet non couplé dans cette ordre et C l'ensemble des sommets de X adjacents à y et encore non couplés. Nous savons que le sommet de C candidat au couplage avec y est soit le premier, soit le dernier dans l'ordre $<_x$. Admettons que des indices g et d pointent initialement sur un sommet quelconque de C. Afin de faire pointer l'indice g sur le premier sommet C et l'indice d sur le dernier, il suffit de faire descendre l'indice d dans l'ordre $<_x$ et remonter l'indice g dans l'ordre inverse. Si $g < d$ (l'ensemble C contient au moins deux sommets), alors le sommet x candidat au couplage avec y s'obtient en testant quel est, de g ou de d, celui qui a le petit indice b_i. Le sommet x devant par la suite être supprimé de la liste, l'indice g ou d qui pointe sur x doit être mis à jour (voir la figure 2.4). Comme dans le cas simplement convexe, les sommets de $C \setminus \{x\}$ (ensemble non vide puisque $|C| \geq 2$) restent des candidats potentiels pour un couplage avec le sommet de Y suivant.

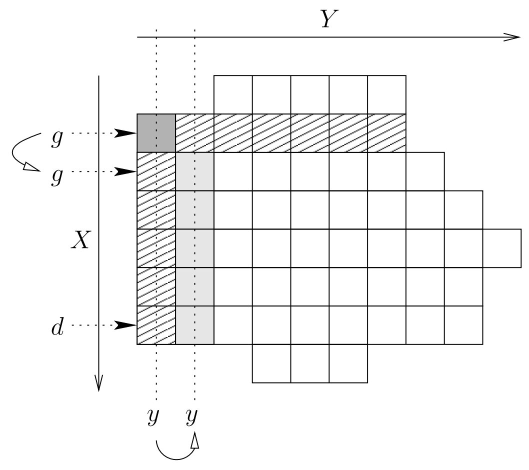


FIG. 2.4 – Le cas $g < d$.

Si $g = d$ (l'ensemble C contient un seul sommet), le sommet x s'obtient immédiatement mais la mise à jour des indices g et d est légèrement plus complexe. Il nous faut déterminer d'une part le prochain sommet de Y à coupler et d'autre part mettre à jour l'ensemble C des candidats (voir la figure 2.5). Du fait de la convexité sur X, ces derniers se trouvent nécessairement parmi les prédécesseurs de x dans l'ordre $<_x$ ou bien parmi ses successeurs. Comme l'exige la modification de l'heuristique de Glover, nous devons choisir le sommet ayant le plus petit indice a_i. Le nouveau sommet $y \in Y$ à coupler sera alors donné par la valeur de cet indice a_i.

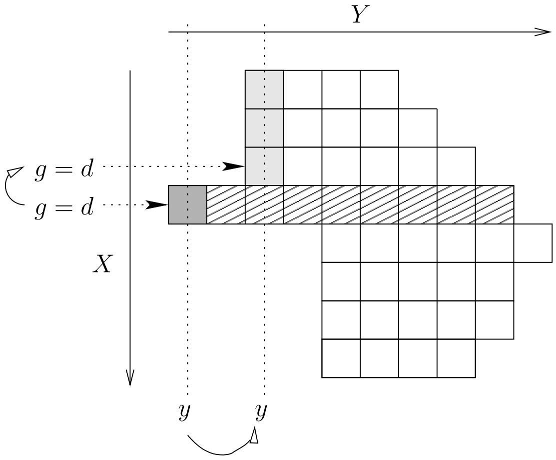


FIG. 2.5 – Le cas $g = d$.

L'algorithme complet est détaillé ci-dessous. L'ensemble X, ordonné selon $<_x$, est représenté sous la forme d'une *liste doublement chaînée* avec les liens $pred_i$ et $succ_i$ associés à chaque élément $i \in X$ (des sentinelles sont placées en début et en fin de liste). Chaque fois qu'un candidat $x \in X$ au couplage avec un sommet $y \in Y$ est trouvé, celui-ci est supprimé de la liste et y est mis à jour (si le couplage entre x et y est possible), ce qui équivaut à supprimer une colonne et éventuellement une ligne de la matrice sommets-*vs*-sommets correspondante. Suite à cela, la matrice ne perd pas la propriété de double convexité, ce qui assure la correction de l'algorithme. Il est à noter que l'algorithme balaye tous les sommets de X sans exception (comme l'algorithme COUPLAGE-BIPARTI-CONVEXE). L'algorithme se termine lorsqu'il ne reste plus que les sentinelles dans liste ($g = d = 0$ ou $g = d = n + 1$).

Algorithme COUPLAGE-BIPARTI-DOUBLEMENT-CONVEXE
Entrée : une représentation doublement convexe de $B = (X, Y, E)$;
Sortie : un couplage maximum $\mathcal{M}$ dans B ;
Début ;
  pour chaque $i \in X$ **faire** $pred_i \leftarrow i - 1$, $succ_i \leftarrow i + 1$;
  $succ_0 \leftarrow 1$, $a_0 \leftarrow 0$, $pred_{n+1} \leftarrow n$, $a_{n+1} \leftarrow 0$;
  déterminer un indice i pour lequel a_i est minimum ($1 \leq i \leq n$) ;
  $\mathcal{M} \leftarrow \emptyset$, $y \leftarrow a_i$, $g \leftarrow i$, $d \leftarrow i$;
  tant que $g \geq 1$ et $d \leq n$ **faire**
    tant que $a_{pred_g} = y$ **faire** $g \leftarrow pred_g$;
    tant que $a_{succ_d} = y$ **faire** $d \leftarrow succ_d$;
    si $g = d$ **alors**
      $x \leftarrow g$;
      si $a_{pred_g} \geq a_{succ_g}$ **alors** $g \leftarrow d \leftarrow pred_g$;
      sinon $g \leftarrow d \leftarrow succ_g$;
    sinon (**si** $g < d$)
      si $b_g \leq b_d$ **alors** $x \leftarrow g$, $g \leftarrow succ_g$;
      sinon $x \leftarrow d$, $d \leftarrow pred_d$;

si $y \leq b_x$ **alors** $\mathcal{M} \leftarrow \mathcal{M} \cup \{(x,y)\}$, $y \leftarrow \max\{y+1, a_g\}$;
$succ_{pred_x} \leftarrow succ_x$, $pred_{succ_x} \leftarrow pred_x$;
retourner $\mathcal{M}$;
Fin ;

Chaque sommet $x \in X$ n'étant visité que deux fois et sa suppression de la liste ne prenant qu'un temps $O(1)$, nous avons le résultat suivant.

Proposition 2.5 *L'algorithme* Couplage-Biparti-Proprement-Convexe *retourne un couplage maximum dans un graphe biparti doublement convexe et connexe* $B = (X, Y, E)$ *en temps et espace* $O(n)$*, étant donnée en entrée une représentation doublement convexe de* B*.*

Remarque. L'algorithme Couplage-Biparti-Proprement-Convexe peut servir de base à un algorithme plus générique permettant de trouver en temps et espace $O(n)$ un couplage maximum dans un graphe biparti doublement convexe $B = (X, Y, E)$, pas nécessairement connexe, mais donné en entrée par une représentation convexe ordonnée. En effet, il est possible de déterminer en temps et espace $O(n)$ les composantes connexes de B. Il suffit pour cela de balayer les sommets de X dans l'ordre des a_i croissants et de retenir le plus grand indice b_i parmi ceux des sommets inspectés : lorsque que le sommet $x \in X$ courant a un indice a_x strictement supérieur à ce dernier, alors la composante connexe courante se termine et une nouvelle débute. Ensuite, Lipski et Preparata [112] donne une procédure pour déterminer la représentation doublement convexe d'un graphe biparti convexe et connexe (s'il en existe une) en temps et espace $O(n)$, étant donnée en entrée une représentation convexe ordonnée de ce graphe.

Pour conclure, attardons-nous quelque peu sur certains graphes bipartis doublement convexes que nous serons amenés à rencontrer plus tard, en particulier lors de l'étude du problème Ordo pour les graphes d'intervalles propres et les graphes d'arcs propres. Soit $B = (X, Y, E)$ un graphe biparti convexe sur Y. Par analogie aux graphes d'intervalles, nous dirons que B est *proprement convexe* si pour toute paire de sommets $i, j \in X$ telle que $a_i \leq a_j$, nous avons aussi $b_i \leq b_j$. En d'autres termes, les intervalles $[a_i, b_i]$ $(1 \leq i \leq n)$ forme un ensemble d'intervalles propres sur Y et il existe un ordre linéaire $<_x$ sur X tel que si $i <_x j$ avec $i, j \in X$, alors $a_i \leq a_j$ et $b_i \leq b_j$. Clairement, cet ordre sur $<_x$, qui s'obtient simplement en triant les paires (a_i, b_i) par ordre lexicographique croissant, assure la double convexité. En effet, une fois les sommets de X ordonnés selon $<_x$, il est aisé de vérifier que la matrice sommets-*vs*-sommets possède la propriété des uns consécutifs pour les lignes et les colonnes. Ainsi, tout graphe biparti proprement convexe est aussi doublement convexe et le problème du couplage maximum dans ces graphes peut être résolu de façon efficace à l'aide de l'algorithme Couplage-Biparti-Doublement-Convexe.

Corollaire 2.4 *Un couplage maximum peut être déterminé en temps et espace* $O(n)$ *dans les graphes bipartis proprement convexes, étant donnée en entrée une représentation convexe ordonnée du graphe.*

Toutefois, exhiber un couplage maximum dans un graphe biparti proprement convexe s'avère être plus facile encore que pour les graphes bipartis doublement convexes. En effet, l'ordre des a_i étant le même que celui des b_i, le sommet candidat au couplage avec un sommet $y \in Y$ est nécessairement le premier dans l'ensemble C (ordonné selon $<_x$) des sommets de X adjacents à y et encore non couplés. Par conséquent, un couplage maximum s'obtient simplement par un balayage linéaire des sommets de X et Y dans l'ordre $<_x$.

Remarque. Tester si un graphe biparti convexe $B = (X, Y, E)$ est proprement convexe se fait en temps et espace $O(n)$, étant donnée une représentation convexe ordonnée de B. D'après la définition, il suffit de balayer les sommets de X en vérifiant que pour chaque $i \in X$, $a_{i-1} \leq a_i$ et $b_{i-1} \leq b_i$.

2.2.2 De la coloration des graphes d'intervalles

La première étape de l'algorithme 2-Ordo-Intervalles consiste à déterminer une coloration minimum de l'ensemble d'intervalles $\mathcal{I} = \{I_1, \ldots, I_n\}$; dans la deuxième étape, c'est une clique maximum qu'il faut extraire de $\mathcal{I}$. Souvent formalisés en d'autres termes que ceux de la théorie des graphes, ces problèmes ont été étudiés il y a déjà plusieurs dizaines d'années et divers algorithmes ont été proposés pour les résoudre. Les références sur le sujet (regroupées pour la plupart dans [80, 81, 75]) font apparaître deux types d'algorithmes. Nous avons d'un côté les algorithmes de type “géométrique”, qui travaillent sur les points induits par les extrémités des intervalles sur l'axe des réels (voir par exemple [80, 81]) et de l'autre les algorithmes de type “graphique”, qui travaillent sur le graphe d'intervalles lui-même, généralement représenté par listes d'adjacence (voir par exemple [70, 124]).

Dans cette partie, nous présentons un nouvel algorithme de coloration pour les graphes d'intervalles, qui, tout en se fondant sur des éléments de théorie des graphes, travaille sur la représentation $\mathcal{I} = \{I_1, \ldots, I_n\}$ du graphe (comme l'algorithme 2-Ordo-Intervalles). Cet algorithme s'exécute en temps et espace $O(n)$ si les ordres $<_d$ et $<_g$ sont donnés en entrée, et peut être modifié pour exhiber dans le même temps une clique maximum. À notre connaissance, seuls Gupta *et al.* [80] ont donné un algorithme d'une telle complexité (sans toutefois prouver sa validité). Leur algorithme prend en entrée la liste des extrémités des intervalles triées dans un certain ordre, qui correspond en fait à la fusion des deux ordres $<_d$ et $<_g$. Tous les autres algorithmes de la littérature reposent sur des heuristiques gloutonnes basées sur le seul ordre $<_g$ (ou $<_d$) et s'exécutent en temps $O(n \log n)$ ou $O(n^2)$, même si l'ordre en question est fourni en entrée. D'ailleurs, nous conjecturons que si un seul des deux ordres $<_g$ ou $<_d$ est donné en entrée, alors $\Omega(n \log n)$ est une borne inférieure de complexité pour ce problème.

Un nouvel algorithme de coloration

Tout d'abord, nous montrons que le problème de la coloration d'un graphe d'intervalles peut se réduire à un problème de couplage maximum dans un graphe biparti. Notons $B_i = (X, Y, E)$ le graphe biparti tel que $X = Y = \mathcal{I}$ et $E = \{(I_i, I_j) \mid I_i \in X, I_j \in Y \text{ et } I_i \prec I_j\}$.

Lemme 2.5 *Si $c(B_i)$ dénote le cardinal d'un couplage maximum dans B_i, alors le nombre chromatique $\chi(\mathcal{I})$ de l'ensemble $\mathcal{I}$ d'intervalles est exactement $n - c(B_i)$.*

Preuve. La preuve que nous donnons est constructive : nous montrons comment construire une partition $\mathcal{S}$ de $\mathcal{I}$ en stables à partir d'un couplage $\mathcal{M}_i$ de B_i. Initialement, $\mathcal{S}$ est constitué de n stables de taille un, chaque stable contenant un intervalle de $\mathcal{I}$. Ensuite, pour chaque arête $(I_i, I_j) \in \mathcal{M}_i$, un nouveau stable S_{ij} est construit par union du stable $S_i \in \mathcal{S}$ contenant l'intervalle I_i et du stable $S_j \in \mathcal{S}$ contenant l'intervalle I_j. L'ensemble $\mathcal{M}_i$ étant un couplage, celui-ci ne peut contenir deux arêtes incidentes à un même sommet. Par conséquent, l'intervalle I_i ne pourra être ici que l'intervalle le plus à droite de S_i et I_j l'intervalle le plus à gauche de S_j (puisque $I_i \prec I_j$). De là, nous obtenons que l'ensemble S_{ij} est un stable et nous pouvons réaliser les opérations suivantes sur $\mathcal{S}$ de façon à diminuer de un son cardinal : $\mathcal{S} \leftarrow \mathcal{S} \setminus \{S_i\}$, $\mathcal{S} \leftarrow \mathcal{S} \setminus \{S_j\}$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_{ij}\}$. Après un passage sur toutes les arêtes de $\mathcal{M}_i$, nous aurons donc $|S| = n - |\mathcal{M}_i|$. Lorsque $\mathcal{M}_i$ est maximum, nous obtenons $|S| = n - c(B_i)$ et aussitôt l'inégalité (i) $\chi(\mathcal{I}) \leq n - c(B_i)$.

En constatant que la construction précédente est complètement réversible (c'est-à-dire qu'en suivant le procédé inverse, nous pouvons construire un couplage $\mathcal{M}_i$ de cardinal $n - \chi(\mathcal{I})$ étant donnée une partition minimum de $\mathcal{I}$ en stables), nous avons l'inégalité (ii) $c(B_i) \geq n - \chi(\mathcal{I})$. En joignant les inégalités (i) et (ii), nous obtenons finalement que $\chi(\mathcal{I}) + c(B_i) = n$. $\square$

Remarque. Il est à noter que le lemme 2.5 peut être généralisé en la proposition suivante, établie en d'autres termes par Hopcroft et Karp [89] dans le cadre du problème de la décomposition d'un ordre partiel en chaînes. Une *chaîne* (resp. *antichaîne*) dans un ordre partiel correspond à une clique (resp. un stable) dans le graphe de comparabilité induit par celui-ci.

Proposition 2.6 *Soit $G = (V, A)$ un graphe dont le complément est transitivement orientable et $B_g = (X, Y, E)$ le graphe biparti tel que $X = Y = V$ et $E = \{(x, y) \mid x \in X, y \in Y$ et $xy \in \vec{A}\}$. Si $c(B_g)$ dénote le cardinal d'un couplage maximum dans B_g, alors $\chi(G) = n - c(B_g)$.*

Le lemme précédent traite du cas particulier des graphes d'intervalles. Toutefois, la preuve que nous donnons repose uniquement sur le fait que le complément du graphe est le graphe d'un ordre partiel, en l'occurrence le graphe de l'ordre $\prec$ d'intervalles. Nous laisserons au lecteur le soin de vérifier qu'en utilisant un ordre partiel quelconque, la preuve en question reste correcte.

Le lemme suivant dévoile, quant à lui, la structure singulière du graphe biparti B_i.

Lemme 2.6 *Le graphe biparti B_i est proprement convexe. De plus, si (a_i, b_i) dénote l'intervalle des sommets de Y adjacents au sommet $I_i \in X$ et ordonnés selon $<_d$, alors nous avons $b_i = n$ pour tout sommet $I_i \in X$ non isolé.*

Preuve. La propriété se vérifie aisément après avoir ordonné les intervalles de X selon $<_d$ et les intervalles de Y selon $<_g$. Soit $I_i \in X$ un sommet non isolé (c'est-à-dire adjacent à au moins un sommet Y) et $I_j \in Y$ le sommet de plus petit indice j tel que $(I_i, I_j) \in E$. Par définition, nous avons que $d(I_i) < g(I_j)$. Nous en déduisons que pour tout sommet $I_{j'} \in Y$ avec $j' \geq j$, $(I_i, I_{j'}) \in E$. Cette assertion établit la Y-convexité de B_i, mais aussi que pour tout sommet $I_i \in X$ non isolé, $b_i = n$. Puisque les sommets non isolés de X ont tous la même extrémité b_i, le graphe biparti est proprement convexe. $\square$

Remarque. De façon symétrique, si (a_j, b_j) dénote l'intervalle des sommets de X adjacents au sommet $I_j \in Y$ et ordonnés selon $<_g$, alors nous avons $a_j = 1$ pour tout sommet $I_j \in Y$ non isolé.

D'après les lemmes 2.5 et 2.6, le problème de la coloration d'un ensemble $\mathcal{I}$ d'intervalles se réduit au problème du couplage maximum dans un graphe biparti proprement convexe. L'algorithme de coloration est donc le suivant : construire le graphe biparti proprement convexe B_i, déterminer un couplage maximum dans B_i, puis construire les stables à partir du couplage. Chaque étape peut être effectuée en temps et espace linéaire, étant donnés en entrée les ordres $<_d$ et $<_g$ sur $\mathcal{I}$ (en utilisant notamment la corollaire 2.4 et la construction de la preuve du lemme 2.5). Par souci d'efficacité, l'algorithme que nous présentons ci-après ne marque pas ces trois étapes. En effet, il n'est pas nécessaire de construire explicitement le graphe biparti B_i (comme pour l'algorithme 2-Ordo-Intervalles) et la construction des stables peut se faire en même temps que le couplage. Nous utilisons deux tableaux auxiliaires $\{x_i\}_{i=1,\dots,n}$ et $\{y_i\}_{i=1,\dots,n}$ pour stocker les indices des intervalles de $\mathcal{I}$ triés selon $<_d$ ou $<_g$, et un troisième tableau $\{z_i\}_{i=1,\dots,n}$ pour retenir l'indice k du stable auquel un intervalle $I_i \in \mathcal{I}$ appartient.

Algorithme Coloration-Intervalles ;
Entrée : un ensemble d'intervalles $\mathcal{I} = \{I_1, \dots, I_n\}$, les ordres $<_d$ et $<_g$ sur $\mathcal{I}$;
Sortie : une coloration optimale $\mathcal{S}$ de $\mathcal{I}$;
Début ;
    soit $x_1, \dots, x_n$ les indices des intervalles de $\mathcal{I}$ dans l'ordre $<_d$;
    soit $y_1, \dots, y_n$ les indices des intervalles de $\mathcal{I}$ dans l'ordre $<_g$;
    $\mathcal{S} \leftarrow \emptyset$, $S_1 \leftarrow \cdots \leftarrow S_n \leftarrow \emptyset$, $z_1 \leftarrow \cdots \leftarrow z_n \leftarrow 0$, $j \leftarrow 1$, $k \leftarrow 1$;
    pour i de 1 à n **faire**
        tant que $j \leq n$ et $d(I_{x_i}) \geq g(I_{y_j})$ **faire** $j \leftarrow j + 1$;
        si $j \leq n$ **alors**
            si $z_{x_i} = 0$ **alors**
                $k \leftarrow k + 1$, $S_k \leftarrow \{I_{x_i}, I_{y_j}\}$;
                $z_{x_i} \leftarrow k$, $z_{y_j} \leftarrow k$;
            sinon
                $k' \leftarrow z_{x_i}$; $S_{k'} \leftarrow S_{k'} \cup \{I_{y_j}\}$;
                $z_{y_j} \leftarrow k'$;
    retourner $\mathcal{S} \leftarrow \{S_1, \dots, S_k\}$;
Fin ;

Proposition 2.7 *L'algorithme* Coloration-Intervalles *retourne en temps et espace $O(n)$ une coloration minimum d'un ensemble $\mathcal{I}$ de n intervalles, étant donnés en entrée les ordres $<_d$ et $<_g$ sur $\mathcal{I}$.*

Puisqu'une représentation par intervalles ordonnée (selon $<_d$ ou $<_g$) peut être obtenue en temps et espace linéaire à partir d'un graphe d'intervalles ou de son complément [82], nous obtenons les corollaires suivants.

Corollaire 2.5 *Le problème de la coloration minimum peut être résolu en temps et espace linéaire pour les graphes d'intervalles.*

Corollaire 2.6 *Le problème de la partition minimum en cliques peut être résolu en temps et espace linéaire pour les compléments de graphes d'intervalles.*

Corollaire 2.7 *Le problème de la décomposition minimum en chaînes peut être résolu en temps et espace linéaire pour les ordres d'intervalles.*

Un algorithme pour la clique maximum

Dans leur article, Gupta *et al.* [81] affirment qu'il est possible de modifier leur algorithme de coloration pour trouver une clique maximum sans en augmenter la complexité. Malheureusement, ils ne donnent aucune indication sur comment y parvenir. Nous comblons ici cette lacune en proposant un algorithme permettant d'exhiber simplement une clique maximum de $\mathcal{I}$ à partir de la coloration fournie par l'algorithme Coloration-Intervalles.

Algorithme Clique-Intervalles ;
Entrée : un ensemble d'intervalles $\mathcal{I} = \{I_1, \ldots, I_n\}$, les ordres $<_d$ et $<_g$ sur $\mathcal{I}$;
Sortie : une clique maximum C de $\mathcal{I}$;
Début ;
  $\mathcal{S} \leftarrow$ Coloration-Intervalles$(\mathcal{I}, <_d, <_g)$;
  soit i^* l'indice du premier intervalle inséré dans le dernier stable $S_{\chi(G)} \in \mathcal{S}$;
  $C \leftarrow \emptyset$;
  pour i de 1 à n **faire**
    si $g(I_{i^*}) \in I_i$ **alors** $C \leftarrow C \cup \{I_i\}$;
  retourner C ;
Fin ;

Proposition 2.8 *L'algorithme* Clique-Intervalles *retourne en temps et espace $O(n)$ une clique maximum d'un ensemble $\mathcal{I}$ de n intervalles, étant donnés en entrée les ordres $<_d$ et $<_g$ sur $\mathcal{I}$.*

Preuve. Soit i^* l'indice du premier intervalle à avoir été inséré dans le dernier stable $S_{\chi(G)} \in \mathcal{S}$. Dans chaque stable de $\mathcal{S}$, il existe un et un seul intervalle contenant $g(I_{i^*})$ (deux intervalles d'un même stable ne peuvent contenir un même point de la droite). Supposons maintenant que dans le stable S_u $(1 \leq u \leq \chi(G))$, aucun intervalle ne contienne $g(I_{i^*})$ et notons j^* l'indice du premier intervalle à avoir été inséré dans S_u. Puisque l'intervalle I_{i^*} a été placé dans le dernier stable de $\mathcal{S}$, celui-ci a nécessairement la plus grande extrémité droite parmi tous les intervalles qui ont été insérés en première position dans les différents stables de la partition. Par conséquent, nous avons $d(I_{j^*}) < g(I_{i^*})$. D'autre part, si l'algorithme n'a pas placé I_{i^*} dans S_u, c'est nécessairement parce que celui-ci chevauche un intervalle $I_j \in S_u$. Or, par hypothèse, ce dernier doit avoir une extrémité gauche supérieure à celle de I_{i^*}. D'après l'algorithme, c'est donc I_{i^*} qui aurait dû être choisi pour intégrer le stable S_u et non l'intervalle I_j, ce qui est une contradiction. □

Corollaire 2.8 *Le problème de la clique maximum peut être résolu en temps et espace linéaire pour les graphes d'intervalles.*

Corollaire 2.9 *Le problème du stable maximum peut être résolu en temps et espace linéaire pour les compléments de graphes d'intervalles.*

Corollaire 2.10 *Le problème de l'antichaîne maximum peut être résolu en temps et espace linéaire pour les ordres d'intervalles.*

Bornes inférieures de complexité

Pour conclure, nous discutons des bornes inférieures de complexité relatives aux problèmes que nous venons d'étudier. Tout d'abord, voici un premier résultat de complexité pour un problème mettant en jeu un ensemble d'intervalles.

Proposition 2.9 (Shamos et Hoey, 1976 – Fredman et Weide, 1978) *Il n'existe aucun algorithme déterministe pouvant décider en moins de $\Omega(n \log n)$ comparaisons si n intervalles aux extrémités entières sont deux-à-deux disjoints.*

Gupta *et al.* [80, 81] donnent dans leurs articles plusieurs références au sujet de ce résultat. En remarquant que déterminer si n intervalles d'un ensemble $\mathcal{I}$ sont deux-à-deux disjoints revient à savoir si $\omega(\mathcal{I}) = \chi(\mathcal{I}) = 1$ ou encore $\alpha(\mathcal{I}) = \kappa(\mathcal{I}) = n$, ils déduisent de cette proposition le corollaire suivant.

Corollaire 2.11 (Gupta *et al.*, 1979) *Soit $\mathcal{I}$ un ensemble de n intervalles aux extrémités entières. Aucun algorithme déterministe ne peut calculer en moins de $\Omega(n \log n)$ comparaisons une coloration minimum, une clique maximum, une partition minimum en cliques, ou encore un stable maximum dans $\mathcal{I}$.*

En d'autres termes, si aucun des deux ordres $<_g$ ou $<_d$ sur $\mathcal{I}$ n'est fourni en entrée, alors aucun algorithme ne peut déterminer une solution à un de ces quatre problèmes en un temps inférieur à $\Omega(n \log n)$. Les algorithmes Coloration-Intervalles et Clique-Intervalles sont donc optimaux en temps et espace, les ordres $<_g$ et $<_d$ s'obtenant en temps $O(n \log n)$ et espace $O(n)$ par un simple tri des extrémités.

Toutefois, nous pensons que l'assertion suivante, plus fine, doit pouvoir être démontrée : *aucun algorithme déterministe ne peut calculer en moins de $\Omega(n \log n)$ comparaisons une coloration minimum, une clique maximum, ou encore un couplage maximum d'intervalles disjoints dans $\mathcal{I}$, étant donné en entrée un seul des deux ordres $<_g$ et $<_d$ sur $\mathcal{I}$.* Voici une instance $\mathcal{I}$ pour laquelle ces trois problèmes sont équivalents et où les deux ordres $<_g$ et $<_d$ semblent nécessaires à l'obtention une solution optimale. L'ensemble $\mathcal{I}$, de cardinal pair, est subdivisé en deux sous-ensembles $\mathcal{I}_g$ et $\mathcal{I}_d$ de taille égale, chacun contenant $n/2$ intervalles. Les intervalles de l'ensemble $\mathcal{I}_g$ ont comme extrémité gauche 1 et les intervalles de l'ensemble $\mathcal{I}_d$ ont comme extrémité droite n. Les extrémités droites (resp. gauches) des intervalles de $\mathcal{I}_g$ (resp. $\mathcal{I}_d$) sont arbitrairement comprises entre 1 et n (voir la figure 2.6).

Par définition, les ensembles $\mathcal{I}_g$ et $\mathcal{I}_d$ induisent chacun une clique de taille $n/2$. La question est de décider s'il existe une coloration de $\mathcal{I}$ de cardinal $n/2$. Clairement, cela équivaut à décider s'il existe une clique de taille strictement supérieure à $n/2$, ou encore un couplage d'intervalles disjoints de cardinal $n/2$. Si les deux ordres $<_g$ et $<_d$ sont connus, alors n'importe lequel des trois algorithmes Coloration-Intervalles, Clique-Intervalles ou bien 2-Ordo-Intervalles permet de répondre à cette question en temps et espace $O(n)$. Si aucun de ces deux ordres n'est connu, alors aucun algorithme ne peut répondre à cette question en un temps inférieur à $\Omega(n \log n)$, d'après le corollaire 2.11. Nous conjecturons alors qu'il en est de même si un seul des deux ordres $<_g$ et $<_d$ est connu.

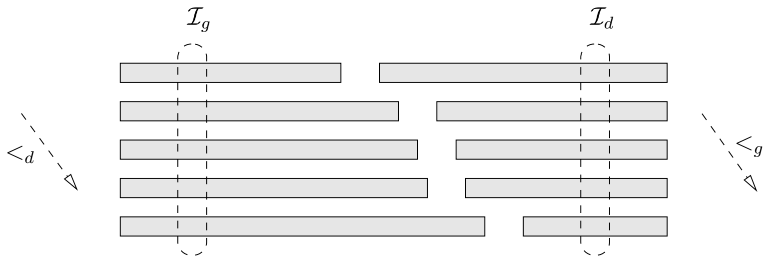


Fig. 2.6 – L'instance $\mathcal{I} = \mathcal{I}_g \cup \mathcal{I}_d$.

Remarque. D'un autre côté, le calcul d'un stable maximum ou d'une partition minimum en cliques ne prend qu'un temps et espace $O(n)$ lorsqu'un seul des deux ordres $<_g$ et $<_d$ est connu [28, p. 324–327].

2.3 Quelques cas polynomiaux

Baker et Coffman [6] ont donné un algorithme en temps et espace $O(n + k^2 \log k)$ pour résoudre le problème Ordo pour les forêts. Tout graphe d'intervalles biparti formant une forêt, nous avons immédiatement le résultat suivant.

Proposition 2.10 (Baker et Coffman, 1996) *Le problème* Ordo *peut être résolu en temps et espace $O(k^2 \log k + n)$ pour les graphes d'intervalles bipartis.*

Dans cette dernière section, nous montrons que le problème Ordo peut être traité en temps et espace linéaire pour deux autres sous-classes que nous avons présentées dans le chapitre introductif : les graphes d'intervalles propres et les graphes à seuil.

2.3.1 Un algorithme linéaire pour les graphes d'intervalles propres

Dans leur article, Andrews *et al.* [3] proposent un algorithme plus simple pour déterminer un couplage maximum d'intervalles disjoints lorsque les intervalles sont propres. Leur algorithme s'exécute en temps $O(\log n)$ sur une architecture PRAM EREW à $O(n/\log n)$ processeurs si les extrémités des intervalles sont triées en entrée (se reporter à [28, p. 675–715] pour une introduction à l'algorithmique parallèle). La version séquentielle de leur algorithme s'exécute en temps et espace $O(n)$ sous les mêmes conditions. Bien que l'algorithme soit plus simple, celui-ci fait tout de même appel à une procédure de couplage externe pour calculer le cardinal du couplage maximum.

Lorsque les intervalles sont propres, le graphe biparti B_c défini dans l'algorithme 2-Ordo-Intervalles n'est plus seulement convexe, mais proprement convexe. En effet, après avoir ordonné les intervalles de C et de $\mathcal{I} \setminus C$ selon $<_g$, le lecteur pourra vérifier que pour tout sommet $i \in X$ $(1 \leq i \leq n-1)$, nous avons $a_i \leq a_{i+1}$ et $b_i \leq b_{i+1}$. Suite à la proposition 2.4, l'implantation de l'algorithme 2-Ordo-Intervalles peut donc être largement simplifiée dans le cas des graphes d'intervalles propres. Dans cette partie, nous dépassons ce résultat en présentant un algorithme linéaire de type glouton pour résoudre le problème Ordo pour les graphes d'intervalles propres quel que soit k, et non plus uniquement pour $k = 2$. Pour faciliter l'écriture de l'algorithme, les intervalles et les stables seront ici numérotés à partir de zéro.

Algorithme Ordo-Intervalles-Propres ;
Entrée : un ensemble $\mathcal{I} = \{I_0, \ldots, I_{n-1}\}$ d'intervalles propres ordonné, un entier k ;
Sortie : une solution optimale $\mathcal{S}$ au problème Ordo pour $\mathcal{I}$;
Début ;
  calculer $\omega(\mathcal{I})$;
  $\chi(\mathcal{I}, k) \leftarrow \max(\omega(\mathcal{I}), \lceil n/k \rceil)$;
  $S_0 \leftarrow \cdots \leftarrow S_{\chi(\mathcal{I},k)-1} \leftarrow \emptyset$;
  pour i de 0 à $n-1$ **faire**
    $u \leftarrow i \bmod \chi(\mathcal{I}, k)$, $S_u \leftarrow S_u \cup \{I_i\}$;
  retourner $\mathcal{S} \leftarrow \{S_1, \ldots, S_{\chi(\mathcal{I},k)}\}$;
Fin ;

Proposition 2.11 *L'algorithme* Ordo-Intervalles-Propres *retourne une solution optimale en temps et espace $O(n)$ au problème* Ordo *pour un ensemble $\mathcal{I}$ de n intervalles propres, ordonné selon $<_g$ en entrée.*

Preuve. La taille $\omega(\mathcal{I})$ d'une clique maximum de $\mathcal{I}$ peut être obtenue en temps et espace $O(n)$ par l'algorithme Clique-Intervalles détaillé p. 45 (lorsque les intervalles sont propres, les ordres $<_g$ et $<_d$ se confondent). Ensuite, le reste de l'algorithme s'exécute en temps et espace $O(n)$. Pour conclure, nous prouvons que l'ensemble $\mathcal{S}$ de stables retourné par l'algorithme forme une solution optimale pour le problème Ordo.

Tout d'abord, nous affirmons que les stables $S_0, \ldots, S_{\chi(\mathcal{I},k)-1}$ ont tous une taille au plus k. D'après l'algorithme, les stables sont de taille égale (à une unité près, si n n'est pas un multiple de k). Ainsi, l'existence d'un stable de taille strictement supérieure à k impliquerait $n > k\chi(\mathcal{I}, k)$, ce qui est une contradiction. Maintenant, supposons que deux intervalles $I_i, I_j \in S_u$ avec $i < j$ se chevauchent $(0 \leq u \leq \chi(\mathcal{I}, k) - 1)$. Par l'algorithme, nous avons $i = u + \alpha\chi(\mathcal{I}, k)$ and $j = u + \beta\chi(\mathcal{I}, k)$ avec $\alpha < \beta$. Quand les intervalles sont propres, les extrémités gauches apparaissent dans le même ordre que les extrémités droites. De ce fait, les intervalles $I_i, I_{i+1}, \ldots, I_{j-1}, I_j$ contiennent tous la portion $[g(I_j), d(I_i)]$ de la droite, induisant ainsi une clique de taille $j - i + 1 = (\beta - \alpha)\chi(\mathcal{I}, k) + 1 > \chi(\mathcal{I}, k)$. L'existence d'une telle clique implique donc une contradiction et l'ensemble $\mathcal{S}$ forme bien une partition de G en stables de taille au plus k. Comme $\max(\omega(\mathcal{I}), \lceil n/k \rceil)$ est une borne inférieure pour $\chi(\mathcal{I}, k)$, l'ensemble $\mathcal{S}$ est en plus de cardinal minimum. □

Corollaire 2.12 *Pour tout graphe G d'intervalles propres, l'égalité*

$$\chi(G,k) = \max\{\omega(G), \lceil n/k \rceil\}$$

est satisfaite quel que soit k.

Puisqu'une représentation par intervalles propres ordonnée (selon $<_g$) peut être obtenue en temps et espace linéaire à partir d'un graphe d'intervalles [31], nous obtenons le corollaire suivant.

Corollaire 2.13 *Le problème* Ordo *peut être résolu en temps et espace linéaire pour les graphes d'intervalles propres.*

2.3.2 Un algorithme linéaire pour les graphes à seuil

Comme le montre la figure 1.17, les graphes à seuil forment une sous-classe des graphes d'intervalles, des graphes de permutation ou encore des graphes scindés. Alors que le problème est $\mathcal{NP}$-difficile pour les graphes d'intervalles ou de permutation [14, 100], celui-ci devient polynomial lorsque l'on se restreint aux graphes scindés [113, 14] et *a fortiori* aux graphes à seuil.

Après avoir rappelé le résultat de Lonc [113] et Bodlaender et Jansen [14] concernant les graphes scindés, nous montrons que le problème peut être résolu en temps et espace linéaire pour les graphes à seuil, et même plus généralement pour une classe de graphes que nous appellerons graphes scindés S-convexes, par analogie aux graphes bipartis convexes.

Exclusion mutuelle par un graphe scindé

Lonc [113] et Bodlaender et Jansen [14] ont indépendamment montré que le problème Ordo peut être résolu en temps et espace polynomial pour les graphes scindés. Nous rappelons qu'un graphe scindé est de la forme $G = (S, C, E)$ où S est un stable, C une clique et E l'ensemble des arêtes reliant les sommets de S à ceux de C. Par concision, nous écrirons $s = |S|$ et $c = |C|$.

Alors que Lonc [113] réduit le problème Ordo à un problème de couplage maximum dans un graphe biparti, Bodlaender et Jansen [14] démontrent que celui-ci est équivalent à un problème de flot maximum. Les deux algorithmes sont tous deux basés sur l'observation suivante, où $\vartheta(G)$ dénote le nombre maximum de sommets de S qui appartiennent à des stables disjoints contenant chacun un sommet de C et de taille au plus k.

Lemme 2.7 *Soit $G = (S, C, E)$ un graphe scindé et $k \geq 2$ un entier. Alors $\chi(G,k) = c + \lceil (s - \vartheta(G))/k \rceil$.*

Preuve. Les sommets de C doivent être placés dans des stables différents, ce qui implique $\chi(G,k) \geq c$. Après avoir extrait les $\vartheta(G)$ sommets qui appartiennent à des stables disjoints contenant chacun un sommet de C et de taille au plus k, les sommets de S restants peuvent donc être groupés en $\lceil (s - \vartheta(G))/k \rceil$ stables de taille au plus k. □

Lonc [113] montre que la valeur $\vartheta(G)$ correspond au cardinal d'un couplage maximum dans un certain graphe biparti, noté B_l. La partition optimale de G en stables de taille au plus k s'obtient ensuite à partir du couplage. Le graphe biparti B_l s'obtient de la façon suivante : supprimons toutes les arêtes de C, remplaçons chaque sommet de $u \in C$ par $k-1$ sommets $u_1, \ldots, u_{k-1}$ et joignons chacun de ces sommets à tous les sommets de S non connectés à u. Le nombre de sommets de ce graphe est alors $s+kc$ et son nombre d'arêtes de l'ordre de $O(ksc)$. Par conséquent, sa construction peut s'effectuer en temps et espace $O(ksc)$ et un couplage maximum s'obtenir en temps $O(ksc\sqrt{s+kc})$ [89]. Ensuite, voici comment obtenir une partition de G en stables de taille au plus k : pour chaque sommet $u \in C$, définissons un stable contenant u et les sommets de S couplés aux sommets $u_1, \ldots, u_{k-1}$, puis découpons de façon optimale l'ensemble des sommets qui restent dans S. Ce travail ne consommant qu'un temps $O(s+kc)$, la méthode de Lonc s'exécute finalement en temps $O(ksc\sqrt{s+kc})$ et espace $O(ksc)$, c'est-à-dire en temps $O(k^{1.5}n^{2.5})$ et espace $O(kn^2)$ avec n le nombre total de sommets du graphe.

Proposition 2.12 (Lonc, 1991) *Le problème* Ordo *peut être résolu en temps $O(k^{1.5}n^{2.5})$ et espace $O(kn^2)$ pour les graphes scindés.*

Bodlaender et Jansen [14] obtiennent le même résultat en calculant un flot maximum au travers d'un réseau D qu'ils définissent comme suit. Soit a et b les deux sommets représentant respectivement la source et le puits. Le sommet a est relié à tous les sommets de C par des arcs sortants et le sommet b à tous les sommets de S par des arcs entrants. Pour chaque paire de sommets $u \in C$ et $v \in S$, un arc de u vers v est défini si $uv \notin E$ et comme précédemment, toutes les arêtes de C sont supprimées. Les capacités des arcs sont $0 \leq c(a,u) \leq \min\{d(u), k-1\}$ pour tout $u \in C$ (où $d(u)$ dénote le degré sortant de u), $0 \leq c(u,v) \leq 1$ et $0 \leq c(v,b) \leq 1$ pour tout $u \in C$ et $v \in S$. La construction de ce graphe demande $O(sc)$ temps et espace. Ensuite, un flot maximum peut être calculé en utilisant la méthode d'augmentation du flot de Ford et Fulkerson (cf. [149, p. 97–99] ou [28, p. 577–589] pour plus de détails). La valeur d'un flot maximum est ici égal à $\vartheta(G) \leq s$ et les capacités des arcs sont entières. Par conséquent, au plus s augmentations de flot sont nécessaires à l'obtention d'un flot maximum. Chaque augmentation de flot peut se faire en temps et espace $O(sc)$ le long d'un chemin améliorant obtenu par une recherche en largeur dans le réseau résiduel (ce dernier comportant au plus $O(sc)$ arcs). Ainsi, la recherche d'un flot maximum prend un temps $O(s^2c)$ et un espace $O(sc)$. Enfin, nous pouvons déterminer une partition de G en stables de taille au plus k comme suit : pour chaque sommet de $u \in C$, définissons un stable contenant u et tous les sommets de S reliés à u par un arc saturé par le flot, puis découpons de façon optimale l'ensemble des sommets qui restent dans S. En résumé, la méthode de Bodlaender et Jansen s'exécute en temps $O(s^2c)$ et espace $O(sc)$, c'est-à-dire en temps $O(n^3)$ et espace $O(n^2)$ avec n le nombre total de sommets du graphe.

Proposition 2.13 (Bodlaender et Jansen, 1995) *Le problème* Ordo *peut être résolu en temps $O(n^3)$ et espace $O(n^2)$ pour les graphes scindés.*

Remarque. L'utilisation d'un algorithme générique de flot maximum ne permet pas ici d'améliorer significativement la complexité asymptotique de la méthode. Pour une revue complète sur l'évolution de la complexité des algorithmes de flot maximum, nous invitons le lecteur à consulter [149, p. 97–112] et [137, p. 160–161].

Du point de vue de la complexité, l'algorithme de Bodlaender et Jansen sera donc préféré à celui de Lonc, à moins que $k = O(1)$. À présent, nous montrons qu'abaisser les bornes des propositions 2.12 et 2.13 n'est pas tâche facile.

Proposition 2.14 *Le problème* ORDO *pour les graphes scindés est aussi difficile que le problème du couplage maximum dans un graphe biparti lorsque $k \geq 2$.*

Preuve. D'après la proposition 2.12, le problème ORDO pour les graphes scindés se réduit à un problème de couplage dans un graphe biparti. En fait, l'inverse est aussi vrai. Pour résoudre un problème de couplage maximum dans un graphe biparti $B = (X, Y, E)$, nous pouvons définir un graphe scindé $G' = (S, C, E')$ où $S = X$, $C = Y$ et E' contient toutes les paires de sommets uv tel que $u \in X$, $v \in Y$ et $uv \notin E$ ainsi que toutes les paires de sommets dans C. Nous remarquons alors qu'à toute solution du problème ORDO pour G' (pour n'importe quel $k \geq 2$) correspond un couplage maximum dans G. □

Corollaire 2.14 *Le problème* ORDO *pour les graphes triangulés est aussi difficile que le problème du couplage maximum dans un graphe biparti lorsque $k \geq 2$.*

Remarque. Dahlhaus et Karpinski [36] ont montré que le problème ORDO pour les compléments des graphes d'intersection des chemins dans un arbre, une sous-classe des compléments de graphes triangulés englobant les compléments de graphes d'intervalles, est aussi difficile que le problème du couplage maximum dans les graphes bipartis lorsque $k \geq 2$. Nous rappelons qu'à ce jour, la complexité du problème ORDO reste inconnue pour les graphes triangulés lorsque $k = 3$ et pour leurs compléments même lorsque k est fixé.

Le cas des graphes scindés S-convexes ou à seuil

Par analogie aux graphes bipartis, nous dirons qu'un graphe scindé $G = (S, C, E)$ est *S-convexe* s'il existe un ordre linéaire sur les sommets de S tel que pour tout sommet $i \in C$, les sommets de S adjacents à $i \in C$ apparaissent consécutivement dans cette ordre. Une représentation S-convexe de G est donnée par l'ordre sur les sommets de S et pour chaque sommet $i \in C$, deux valeurs a_i et b_i, respectivement les indices des premier et dernier sommets dans l'intervalle (ordonné) des sommets adjacents à i.

Proposition 2.15 *Le problème* ORDO *peut être résolu en temps et espace $O(n)$ pour les graphes scindés S-convexes, étant donnée en entrée une représentation S-convexe du graphe.*

Preuve. La preuve repose sur le fait que le graphe biparti B_l défini dans la méthode de Lonc (voir la proposition 2.12) est convexe sur un cercle dans le cas des graphes scindés S-convexes. En effet, soit $G = (S, C, E)$ un graphe scindé S-convexe et $<$ l'ordre linéaire sur les sommets de S. Les sommets de S adjacents à un sommet $i \in C$ apparaissent consécutivement dans l'ordre $<$. En étendant cet ordre $<$ en un ordre circulaire $<_c$ (le premier sommet de S dans l'ordre $<$ devient alors le successeur du dernier sommet de S dans ce même ordre), les sommets de S non adjacents à i apparaissent consécutivement dans l'ordre $<_c$.

Une représentation convexe sur le cercle de B_l peut être obtenue à partir de la représentation S-convexe de G par un simple balayage des sommets de C. D'après Liang et Blum [110], un couplage maximum dans le graphe biparti B_l peut être déterminé à l'aide de deux passes de l'algorithme de Glover [73]. En s'appuyant sur cette remarque et en modifiant l'algorithme de Glover de façon à autoriser la sélection de $k-1$ arêtes incidentes (et non plus une seule) pour chaque sommet $i \in C$, l'algorithme de Lonc peut être simulé en temps et espace $O(s+c)$, sans avoir construire explicitement le graphe B_l. En effet, le nombre de sommets sélectionnés pour coupler avec des sommets de C reste inférieur à $s \leq n$, le nombre de sommets de l'ensemble S. □

Comme pour les graphes bipartis, une représentation S-convexe peut être déterminée en temps et espace linéaire à l'aide d'un algorithme de reconnaissance de la propriété des uns consécutifs [82]. Par conséquent, nous avons le corollaire suivant.

Corollaire 2.15 *Le problème* Ordo *peut être résolu en temps et espace linéaire pour les graphes scindés S-convexes.*

Nous conclurons cette section en abordant le cas particulier des graphes à seuil. Nous rappelons qu'un graphe à seuil $G = (S, C, E)$ est composé d'une clique $C = C_1 \cup \cdots \cup C_r$ et d'un stable $S = S_1 \cup \cdots \cup S_r$ (avec $r \leq n$ et C_i, S_i non vides pour $i = 1, \ldots, r$) tel qu'un sommet de S_i est adjacent à un sommet de C'_i si et seulement si $i' > i$ pour tout $i, i' \in \{1, \ldots, r\}$. Une telle représentation peut être calculée en temps et espace linéaire à partir de la partition des sommets du graphe selon leur degré (cf. [75, p. 223–227] pour plus de détails). Les graphes à seuil s'avèrent être des *graphes scindés doublement convexes*, c'est-à-dire à la fois convexes sur S et sur C.

Lemme 2.8 *Tout graphe à seuil $G = (S, C, E)$ est un graphe scindé doublement convexe. De plus, les sommets de S peuvent être ordonnés tel que pour tout sommet $i \in C$ connecté par au moins une arête à S, nous ayons $a_i = 1$.*

La preuve du lemme s'obtient sans difficulté à partir de la définition d'un graphe à seuil. De là, il est tout aussi facile de voir qu'un simple balayage linéaire des sommets de $C = C_1 \cup \cdots \cup C_r$ et $S = S_1 \cup \cdots \cup S_r$ dans l'ordre permet d'obtenir une solution optimale au problème Ordo.

Corollaire 2.16 *Le problème* Ordo *peut être résolu en temps et espace linéaire pour les graphes à seuil.*

Chapitre 3

Exclusion mutuelle par un graphe d'arcs ou de tolérances

Comme nous l'avons mentionné dans le théorème 1.1, le résultat de Bodlaender et Jansen [14] concernant la complexité du problème Ordo pour les graphes d'intervalles implique de fait que le problème reste $\mathcal{NP}$-difficile pour les graphes d'arcs et les graphes de tolérances bornées, même lorsque k est un paramètre fixé supérieur ou égal à quatre. Dans ce chapitre, nous étudions avec plus de détails la complexité du problème Ordo pour ces deux classes de graphes [1]. Tout d'abord, nous montrons que le problème peut être résolu en temps $O(n^2)$ pour les graphes d'arcs propres ; un algorithme en temps et espace linéaire est aussi fourni pour le cas particulier $k = 2$. Ensuite, nous complétons le théorème 1.1 de Bodlaender et Jansen [14] en démontrant que le problème 3-Ordo reste $\mathcal{NP}$-difficile pour les graphes de tolérances bornées dont tout cycle de longueur supérieure ou égale à cinq possède au moins deux cordes. Ce résultat implique notamment que le problème reste $\mathcal{NP}$-difficile pour les graphes de Meyniel et les graphes faiblement triangulés dont le complément est de comparabilité, même avec $k \geq 3$ fixé.

3.1 Le cas des graphes d'arcs propres

Le théorème 1.1 semble condamner tout tentative de recherche d'un algorithme polynomial pour le problème Ordo restreint aux graphes d'arcs circulaires (excepté pour $k = 3$, qui reste un problème ouvert). Nous nous intéressons dans cette section à une sous-classe naturelle des graphes d'arcs : les graphes d'arcs propres. Cette classe englobe les graphes d'arcs unitaires et surtout les graphes d'intervalles propres que nous avons étudiés dans le chapitre précédent. Nous montrons que le problème peut être résolu en temps et espace $O(n^2)$ pour les graphes d'arcs propres et même en temps et espace linéaire lorsque $k = 2$.

Nous travaillerons sur une représentation par arcs ouverts du graphe d'arcs. Nous rappelons qu'un arc fermé $A = [a, b]$ (resp. un arc ouvert $A =]a, b[$) du cercle est toujours

[1] Les résultats présentés dans ce chapitre apparaissent en partie dans [66]. À ce propos, nous nous devons de procéder à la rectification suivante : contrairement à ce qui est annoncé dans [66], la complexité du problème Ordo pour les graphes d'arcs reste une question ouverte lorsque $k = 3$.

défini dans le sens des aiguilles d'une montre, c'est-à-dire de l'extrémité a vers l'extrémité b. L'extrémité a est appelée extrémité *scm* (*i.e.* dans le sens contraire des aiguilles d'une montre), tandis que l'extrémité b est appelée extrémité *sm* (*i.e.* dans le sens des aiguilles d'une montre).

Un algorithme quadratique pour le cas général

L'algorithme que nous proposons repose sur le paradigme de l'échange bichromatique de sommets, particulièrement étudié par De Werra [42] dans le contexte des problèmes de coloration d'arêtes et d'emploi du temps.

Lemme 3.1 *Soit G un graphe d'arcs propres et S_u, S_v deux stables disjoints de G. Si les stables S_u et S_v sont de taille différente, alors toute composante connexe du graphe biparti induit par ces deux mêmes stables est isomorphe à une chaîne.*

Preuve. G étant un graphe d'arcs propres, celui-ci ne peut contenir $K_{1,3}$ comme sous-graphe induit. De là, nous pouvons en déduire que toute composante connexe du graphe biparti induit par S_u et S_v forme une chaîne ou bien un cycle pair (tous les sommets du graphe biparti sont de degré au plus deux). Maintenant, considérons un cycle pair C dans une représentation par arcs propres de ce graphe biparti. Clairement, les arcs correspondant aux sommets du cycle C doivent couvrir la totalité du cercle autour duquel est définie la représentation (voir la figure 3.1).

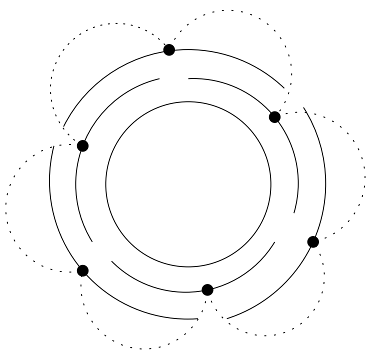


FIG. 3.1 – Une représentation du cycle pair C_6 par arcs propres.

Les stables S_u et S_v étant de taille différente, admettons sans perte de généralité que $|S_u| > |S_v| > 1$. Clairement, il existe un sommet de S_u n'appartenant pas au cycle C. Or, l'arc correspondant à ce sommet est nécessairement intercalé entre deux arcs du stable S_u qui appartiennent au cycle C. Par conséquent, celui-ci est entièrement recouvert par un arc de S_v, ce qui contredit le fait que les arcs sont propres. Ainsi, toute composante connexe du graphe biparti ne peut être isomorphe qu'à une chaîne. □

Lemme 3.2 *Soit G un graphe d'arcs propres et k un entier positif quelconque. Il existe une coloration minimum du graphe G tel qu'une des deux conditions suivantes soit satisfaite :*

(a) chaque couleur apparaît au plus k fois ;
(b) chaque couleur apparaît au moins k fois.

Preuve. Soit $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$ une coloration minimum du graphe G. Si cette coloration ne satisfait pas la condition (a), ni la condition (b), alors nous montrons que l'algorithme présenté ci-dessous nous ramène à une de ces deux conditions. Nous utilisons dans cet algorithme la procédure Composantes-Connexes qui prend en entrée deux stables disjoints S_u, S_v du graphe G et renvoie en sortie l'ensemble $\mathcal{B}$ des composantes connexes du graphe biparti induit par S_u et S_v. Pour chaque composante connexe $B_r \in \mathcal{B}$, nous pouvons accéder à l'ensemble B_r^u (resp. B_r^v) des sommets de B_r qui appartiennent à S_u (resp. S_v).

Algorithme Raffiner-Coloration ;
Entrée : une coloration minimum $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$ de G, un entier k ;
Sortie : une coloration $\mathcal{S}$ satisfaisant une des deux conditions (a) ou (b) ;
Début ;
  tant qu'il existe deux stables disjoints $S_u, S_v \in \mathcal{S}$ avec $|S_u| > k$ et $|S_v| < k$ **faire**
    $\mathcal{B} \leftarrow$ Composantes-Connexes(S_u, S_v) ;
    tant que $|S_u| > k$ et $|S_v| < k$ **faire**
      choisir une composante connexe $B_r \in \mathcal{B}$ tel que $|B_r^u| = |B_r^v| + 1$;
      échanger les sommets de S_u et S_v correspondant respectivement à B_r^u et B_r^v ;
  retourner $\mathcal{S}$;
Fin ;

Prouvons la correction de cet algorithme. Tout d'abord, nous affirmons qu'après avoir déterminé les composantes connexes du graphe biparti induit par les stables S_u et S_v, il existe effectivement une composante $B_r \in \mathcal{B}$ tel que $|B_r^u| = |B_r^v| + 1$, et ce tant que $|S_u| > k$ et $|S_v| < k$. D'après le lemme précédent, chaque composante connexe B_r doit satisfaire à une des trois conditions suivantes : (i) B_r est une chaîne impaire et $|B_r^u| = |B_r^v|$, (ii) B_r est une chaîne paire et $|B_r^u| + 1 = |B_r^v|$, (iii) B_r est une chaîne paire et $|B_r^u| = |B_r^v| + 1$. Les inégalités $|S_u| > k$ et $|S_v| < k$ imposant que $|S_u| \geq |S_v| + 2$, au moins deux composantes connexes du graphe biparti doivent respecter la condition (iii), ce qui justifie notre affirmation. Enfin, à chaque passage dans la première boucle **tant que**, un stable de $\mathcal{S}$ voit sa taille fixée à k. Ainsi, après au plus $\chi(G)$ tours de boucle, l'algorithme retournera une coloration satisfaisant une des deux conditions du lemme. □

A présent, nous pouvons décrire l'algorithme complet de résolution du problème Ordo pour les graphes d'arcs propres. Pour faciliter l'écriture de l'algorithme, les arcs ainsi que les stables sont numérotés à partir de zéro.

Algorithme Ordo-Arcs-Propres ;
Entrée : un ensemble $\mathcal{A} = \{A_0, \ldots, A_{n-1}\}$ d'arcs propres ordonné, un entier k ;
Sortie : une solution optimale $\mathcal{S}^*$ au problème Ordo pour $\mathcal{A}$;
Début ;
  calculer une coloration minimum $\mathcal{S} = \{S_0, \ldots, S_{\chi(\mathcal{A})-1}\}$ de $\mathcal{A}$;
  $\mathcal{S} \leftarrow$ Raffiner-Coloration$(\mathcal{S}, k)$, $\mathcal{S}^* \leftarrow \emptyset$;

si tous les stables de $\mathcal{S}$ sont de taille au plus k **alors** $\mathcal{S}^* \leftarrow \mathcal{S}$;
sinon
  si n n'est pas un multiple de k **alors**
    extraire de n'importe quel stable de $\mathcal{S}$ un stable S' de taille $n \bmod k$;
    ajouter S' à $\mathcal{S}^*$ et supprimer de $\mathcal{A}$ les arcs de S' $(n \leftarrow n - n \bmod k)$;
    numéroter les arcs qui restent dans $\mathcal{A}$ de 0 à $n-1$ dans l'ordre circulaire ;
  $S_0 \leftarrow \cdots \leftarrow S_{n/k-1} \leftarrow \emptyset$;
  pour i de 0 à $n-1$ **faire**
    $u \leftarrow i \bmod n/k$, $S_u \leftarrow S_u \cup \{A_i\}$;
  $\mathcal{S}^* \leftarrow \mathcal{S}^* \cup \{S_0, \ldots, S_{n/k-1}\}$;
retourner $\mathcal{S}^*$;
Fin ;

D'après le lemme 3.2, deux cas se présentent une fois la coloration $\mathcal{S}$ raffinée : soit (a) tous les stables ont une taille inférieure à k (c'est-à-dire $\chi(\mathcal{A}) \geq \lceil n/k \rceil$), soit (b) tous les stables ont une taille supérieure à k et au moins un a une taille strictement supérieure à k (c'est-à-dire $\chi(\mathcal{A}) < \lceil n/k \rceil$). Dans le cas (a), la coloration $\mathcal{S}$ est une solution triviale au problème Ordo. Analysons maintenant le comportement de l'algorithme dans le cas (b).

Tout d'abord, notons qu'extraire un stable S' de taille $n \bmod k < k$ de n'importe quel stable de $\mathcal{S}$ est possible puisque tous ont une taille supérieure à k. De même, il est facile de vérifier que les stables $S_0, \ldots, S_{n/k-1}$ ont tous une taille au plus k à l'issue de leur construction. Supposons maintenant que deux arcs $A_i, A_j \in \mathcal{A}$ avec $i < j$ se chevauchent dans un stable S_u $(0 \leq u \leq n/k - 1)$. Quand les arcs sont propres, l'ordre des extrémités *sm* est le même que celui des extrémités *scm*. De ce fait, les arcs $A_i, A_{i+1}, \ldots, A_{j-1}, A_j$ incluent tous la portion du cercle $[scm(A_j), sm(A_i)]$ et induisent ainsi une clique de taille strictement supérieure à $\lceil n/k \rceil > \chi(\mathcal{A})$, ce qui est une contradiction. Par conséquent, l'ensemble $\mathcal{S}^*$ forme effectivement une partition de $\mathcal{A}$ en stables de taille au plus k. Cette partition étant de cardinal $\lceil n/k \rceil$, une borne inférieure pour $\chi(\mathcal{A}, k)$, elle est en plus optimale.

À présent, voyons la complexité de l'algorithme. Une coloration minimum de $\mathcal{A}$ peut être calculée en temps $O(n^{1.5})$ [141], lorsque les arcs sont ordonnés. La complexité du reste de l'algorithme est dominée par celle de l'algorithme Raffiner-Coloration. Nous avons vu lors de la preuve du lemme 3.2 que cet algorithme s'arrêtait après $\chi(\mathcal{A})$ tours de boucle dans le pire des cas. Les composantes connexes du graphe biparti induit par S_u et S_v peuvent être déterminées en temps et espace $O(|S_u| + |S_v|)$ par un balayage des arcs de $S_u \cup S_v$ dans l'ordre circulaire (l'ordre sur $S_u \cup S_v$ est obtenu par simple fusion des ordres sur S_u et S_v). Correctement implanté, l'échange de sommets entre composantes peut aussi se faire en temps et espace $O(|S_u| + |S_v|)$. En résumé, l'algorithme Raffiner-Coloration s'exécute dans le pire des cas en temps $O(\chi(\mathcal{A})\, n)$ et espace $O(n)$.

Théorème 3.1 *L'algorithme* Ordo-Arcs-Propres *retourne en temps $O(n^2)$ et espace $O(n)$ une solution optimale au problème* Ordo *pour un ensemble $\mathcal{A}$ de n arcs propres, ordonné en entrée.*

Puisqu'une représentation par arcs propres ordonnée peut être obtenue en temps et espace linéaire [39], nous obtenons le corollaire suivant.

Corollaire 3.1 *Le problème* Ordo *peut être résolu en temps $O(n^2)$ et espace $O(n)$ pour les graphes d'arcs propres.*

En accord avec la discussion précédente, nous obtenons aussi le corollaire suivant.

Corollaire 3.2 *Pour tout graphe G d'arcs propres, l'égalité $\chi(G, k) = \max\{\chi(G), \lceil n/k \rceil\}$ est satisfaite quel que soit k.*

Un algorithme linéaire pour $\mathbf{k = 2}$

Pour clore cette section, nous proposons un algorithme linéaire pour résoudre le problème Ordo pour les graphes d'arcs propres lorsque $k = 2$. Comme dans le cas général, nous travaillerons sur une représentation $\mathcal{A} = \{A_1, \ldots, A_n\}$ par arcs propres ordonnée. Mais tout d'abord, voici les grandes lignes de l'algorithme.

Après avoir calculé une clique maximum C dans $\mathcal{A}$, deux cas seront considérés. Si cette clique maximum n'est pas trop grande, soit $\omega(\mathcal{A}) \leq \lfloor n/2 \rfloor$, alors un couplage maximum d'arcs disjoints dans $\mathcal{A}$ sera déterminé de façon gloutonne, comme nous l'avons fait dans la seconde partie de l'algorithme Ordo-Arcs-Propres. Dans le cas contraire, c'est-à-dire $\omega(\mathcal{A}) > \lfloor n/2 \rfloor$, les arcs de la clique C seront divisés en deux catégories : les α-arcs et les β-arcs. Notons c_i l'arc de plus petite extrémité *scm* dans C, puis c_j l'arc de plus grande extrémité *scm* dans C qui contient $sm(c_i)$ (voir la figure 3.2).

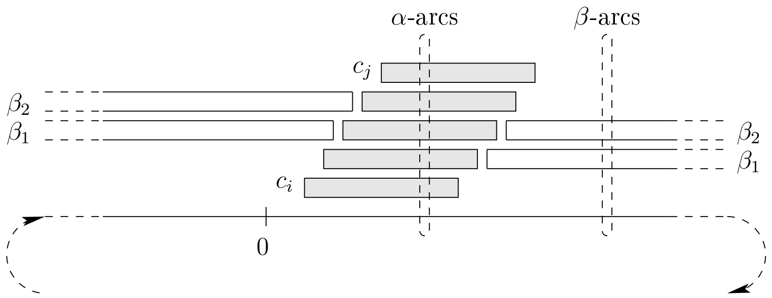


Fig. 3.2 – Les α-arcs et les β-arcs d'une clique.

Il peut alors exister dans C des arcs contenant successivement dans le sens des aiguilles d'une montre l'extrémité $sm(c_j)$ puis l'extrémité $scm(c_i)$, comme nous le montrons sur la figure 3.2. Ces arcs seront appelés β-arcs, et par opposition, tous les autres arcs de C seront baptisés α-arcs (y compris ceux qui ne contiennent ni l'extrémité $sm(c_i)$ ni l'extrémité $sm(c_j)$). Il est à noter que la définition des α-arcs et des β-arcs dans C variera selon la représentation du graphe. C_α et C_β dénoteront respectivement les ensembles des α-arcs et β-arcs de la clique C (nous avons $|C| = |C_\alpha| + |C_\beta|$). De même, $m(C_\alpha)$ (resp. $m(C_\beta)$) dénotera l'ensemble des arcs de $\mathcal{A} \setminus C$ candidats au couplage avec les arcs de C_α (resp. C_β), c'est-à-dire ne chevauchant pas au moins un arc de C_α (resp. C_β). Dans le cas $\omega(\mathcal{A}) > \lfloor n/2 \rfloor$, un couplage maximum s'obtiendra en réalisant un couplage maximum d'arcs disjoints entre C_α et $m(C_\alpha)$ puis entre C_β et $m(C_\beta)$, sa validité tenant au fait qu'aucun arc de $m(C_\alpha)$

ne peut être couplé avec un arc de C_β et *vice versa*. Par la suite, nous verrons comment déterminer efficacement ces deux couplages.

Algorithme 2-Ordo-Arcs-Propres ;
Entrée : un ensemble $\mathcal{A} = \{A_1, \ldots, A_n\}$ d'arcs propres ordonné ;
Sortie : une solution optimale $\mathcal{M}$ au problème 2-Ordo pour $\mathcal{A}$;
Début ;
    calculer une clique maximum C de $\mathcal{A}$;
    $\mathcal{M} \leftarrow \emptyset$;
    si $\omega(\mathcal{A}) \leq \lfloor n/2 \rfloor$ **alors**
        pour i de 1 à $\lfloor n/2 \rfloor$ **faire** $\mathcal{M} \leftarrow \mathcal{M} \cup \{(A_i, A_{\lceil n/2 \rceil + i})\}$;
        si n est impair **alors** ajouter à $\mathcal{M}$ l'arc $A_{\lceil n/2 \rceil}$ qui reste non couplé ;
    sinon
        soit $C = C_\alpha \cup C_\beta$;
        calculer un couplage maximum $\mathcal{M}_\alpha$ entre les arcs de C_α et de $m(C_\alpha)$;
        calculer un couplage maximum $\mathcal{M}_\beta$ entre les arcs de C_β et de $m(C_\beta)$;
        $\mathcal{M} \leftarrow \mathcal{M}_\alpha \cup \mathcal{M}_\beta$;
    retourner $\mathcal{M}$;
Fin ;

A présent, nous établissons la validité de l'algorithme. Tout d'abord, plaçons-nous dans le cas où $\omega(\mathcal{A}) \leq \lfloor n/2 \rfloor$ et rappelons que les arcs de $\mathcal{A}$ sont ordonnés. Supposons que l'ensemble $\mathcal{M}$ exhibé ne soit pas valide, c'est-à-dire que deux arcs A_i et $A_{\lceil n/2 \rceil + i}$ ($1 \leq i \leq \lfloor n/2 \rfloor$) se chevauchent. Dans une représentation par arcs propres, deux arcs ne peuvent pas se chevaucher de part et d'autre du cercle (cf. [75, p. 191–192]). Ainsi, nous avons soit $sm(A_i) \in A_{\lceil n/2 \rceil + i}$, ou bien $scm(A_i) \in A_{\lceil n/2 \rceil + i}$. Quand les arcs sont propres, l'ordre des extrémités scm autour du cercle est le même que celui des extrémités sm. Dans le premier cas, cela implique le chevauchement de tous les arcs entre A_i et $A_{\lceil n/2 \rceil + i}$ dans l'ordre circulaire et dans le second, le chevauchement de tous les arcs entre $A_{\lceil n/2 \rceil + i}$ et A_i dans ce même ordre. Dans un cas comme dans l'autre, l'existence d'une clique de taille $\lfloor n/2 \rfloor + 1$ est avérée, ce qui contredit notre hypothèse première. Comme $\lceil n/2 \rceil$ est une borne inférieure pour $\chi(\mathcal{A}, 2)$, l'ensemble $\mathcal{M}$ est non seulement valide mais aussi de cardinal minimum.

Plaçons-nous maintenant dans le cas où $\omega(\mathcal{A}) > \lfloor n/2 \rfloor$. Nous admettrons que les deux ensembles $C_\alpha = \{\alpha_1, \ldots, \alpha_u\}$ et $C_\beta = \{\beta_1, \ldots, \beta_v\}$ sont ordonnés de la façon suivante : le premier arc de l'ensemble contient les extrémités scm de tous les autres arcs et les arcs suivants sont rangés dans l'ordre circulaire dans le sens des aiguilles d'une montre. Clairement, si l'ensemble C_β est vide, alors une solution optimale au problème 2-Ordo s'obtient par un couplage maximum d'arcs disjoints entre C_α et $m(C_\alpha)$ (dans ce cas, $C_\alpha = C$ et $m(C_\alpha) = \mathcal{A} \setminus C$). D'un autre côté, nous montrons que si C_β n'est pas vide, alors aucun arc de $m(C_\beta)$ ne peut être couplé avec C_α et *vice versa*.

Lemme 3.3 *Si l'ensemble C_β n'est pas vide, alors tout arc de $m(C_\beta)$ (resp. $m(C_\alpha)$) induit une clique avec les arcs de C_α (resp. C_β). De plus, nous avons $|C_\alpha| \geq |m(C_\alpha)|$ et $|C_\beta| \geq |m(C_\beta)|$.*

Preuve. Tout d'abord, montrons que l'ensemble $C_\alpha \cup m(C_\beta)$ induit une clique. Cela impliquera immédiatement que chaque arc de $m(C_\beta)$ induit une clique avec C_α. Par définition de C_β, tout arc de $m(C_\beta)$ doit être inclus dans la portion $]scm(\alpha_1), sm(\alpha_u)[$ du cercle. Comme les arcs sont propres, tout arc de $m(C_\beta)$ doit contenir la portion $[scm(\alpha_u), sm(\alpha_1)]$ du cercle et $C_\alpha \cup m(C_\beta)$ induit bien une clique. Nous obtenons aussitôt que $|C_\beta| \geq |m(C_\beta)|$, puisque le contraire implique maintenant l'existence d'une clique de taille $|C_\alpha| + |m(C_\beta)| > |C_\alpha| + |C_\beta| = \omega(\mathcal{A})$.

Ensuite, démontrons que tout arc de $m(C_\alpha)$ induit une clique avec les arcs de C_β. Un arc de $m(C_\alpha)$ ne peut pas être inclus dans la portion $]sm(\alpha_u), scm(\alpha_1)[$ du cercle (sinon celui-ci serait strictement contenu par un arc de C_β). D'après la discussion précédente, il ne peut pas non plus avoir ses deux extrémités dans la portion $[scm(\alpha_1), sm(\alpha_u)]$ du cercle. Par conséquent, tout arc $m \in m(C_\alpha)$ doit satisfaire à une des deux conditions suivantes : (i) $scm(m) \in [scm(\alpha_1), sm(\alpha_u)]$ et $m \supseteq [scm(\beta_1), sm(\alpha_u)]$, (ii) $sm(m) \in [scm(\alpha_1), sm(\alpha_u)]$ et $m \supseteq [scm(\alpha_1), sm(\beta_v)]$. Dans le cas (i), l'arc m contient toutes les extrémités sm des arcs de C_β, alors que dans le cas (ii), celui-ci contient toutes les extrémités scm des arcs de C_β. Dans les deux cas, l'arc m chevauche donc tous les arcs de C_β.

Pour conclure, nous montrons que $|C_\alpha| \geq |m(C_\alpha)|$. Soit i le plus petit indice d'un arc de C_α contenant $scm(\beta_1)$ et j le plus grand indice d'un arc de C_α contenant $sm(\beta_v)$. Si plus de $i-1$ arcs de $m(C_\alpha)$ satisfont la condition (i), alors ceux-ci induisent une clique de taille strictement supérieure à $\omega(\mathcal{A})$ avec les arcs $\{\alpha_i, \dots, \alpha_u\} \cup \{\beta_1\}$. De même, si plus de $u-j$ arcs de $m(C_\alpha)$ satisfont la condition (ii), alors ceux-ci induisent une clique de taille strictement supérieure à $\omega(\mathcal{A})$ avec les arcs $\{\alpha_1, \dots, \alpha_j\} \cup \{\beta_v\}$. Par conséquent, il ne peut pas exister plus de $u - j + i - 1 < u$ arcs dans $m(C_\alpha)$ (par définition, nous avons que $i \leq j - 1$). □

D'après ce dernier lemme, un couplage maximum d'arcs disjoints dans $\mathcal{A}$ est bien l'union des deux couplages $\mathcal{M}_\alpha$ et $\mathcal{M}_\beta$. Toutefois, une assertion plus forte encore peut être obtenue.

Lemme 3.4 $|\mathcal{M}_\alpha| = |m(C_\alpha)|$ *et* $|\mathcal{M}_\beta| = |m(C_\beta)|$ *(si l'ensemble* C_β *n'est pas vide).*

Preuve. Nous établirons seulement l'égalité $|\mathcal{M}_\alpha| = |m(C_\alpha)|$, la preuve de l'inégalité $|\mathcal{M}_\beta| = |m(C_\beta)|$ étant complètement symétrique. Pour cela, supposons que $|\mathcal{M}_\alpha| < |m(C_\alpha)|$ et notons m^* un des arcs de $m(C_\alpha)$ qui reste non couplé à l'issue de l'algorithme.

Tout d'abord, nous affirmons qu'il existe au moins un arc $\alpha_j \in C_\alpha$ tel que $scm(m^*) \in \alpha_j$. Pour cela, considérons qu'un tel arc n'existe pas et notons i le plus grand indice d'un arc de C_α non couplé (il en existe au moins un puisque m^* n'est pas couplé et que $|m(C_\alpha)| \leq |C_\alpha|$). Puisque $scm(m^*) \notin \alpha_i$, nous devons avoir $sm(m^*) \in \alpha_i$ (sinon m^* serait couplé à α_i par l'algorithme). Maintenant considérons l'ensemble d'arcs $M \subseteq m(C_\alpha)$ couplés avec les arcs $\alpha_j \in C_\alpha$ tel que $j > i$. Puisque l'algorithme ne les a pas couplés à α_i, nous devons avoir $sm(m) \in \alpha_i$ pour tout arc $m \in M$. Comme les arcs sont propres, nous obtenons que l'ensemble d'arcs $\{\alpha_1, \dots, \alpha_i\} \cup \{m^*\} \cup M$ induit une clique de taille $|C_\alpha| + 1$, ce qui est une contradiction, et l'affirmation est démontrée. Nous noterons j^* le plus petit indice d'un arc de C_α tel que $scm(m^*) \in \alpha_{j^*}$. Clairement, nous avons que $1 < j \leq u$ (si $j^* = 1$, nous obtenons immédiatement une clique de taille $|C_\alpha| + 1$).

Ensuite, nous affirmons qu'il existe au moins un arc $\alpha_i \in C_\alpha$ avec $i < j^*$ qui n'a pas été couplé. La preuve repose sur le fait suivant : tout arc couplé à un arc $\alpha_j \in C_\alpha$ avec $j < j^*$

voit son extrémité scm contenue par α_{j^*}. Supposons le contraire : il existe un arc m couplé à α_j dont l'extrémité n'est pas contenue par α_{j^*}. Par définition de j^*, l'arc m^* ne contient pas l'extrémité scm de α_j et comme les arcs sont propres, m^* ne contient pas non plus son extrémité sm (le contraire impliquerait en effet $m \subset m^*$). Par conséquent, l'algorithme aurait dû coupler m^*, et non pas m, à α_j puisque m^* précède m dans l'ordre circulaire, ce qui est une contradiction. Le fait étant justifié, nous pouvons maintenant démontrer notre affirmation première. Supposons que tous les arcs de C_α et d'indice $j < j^*$ soient couplés et notons M l'ensemble des arcs de $m(C_\alpha)$ auxquels ceux-ci sont couplés. En s'appuyant sur le fait précédent, nous obtenons que l'ensemble $M \cup \{m^*\} \cup \{\alpha_{j^*}, \dots, \alpha_u\}$ induit une clique de taille $|C_\alpha| + 1$, ce qui est une contradiction.

À présent, nous sommes en mesure d'établir la preuve du lemme. En accord avec la discussion précédente, notons M l'ensemble des arcs de $m(C_\alpha)$ couplés aux arcs $\alpha_j \in C_\alpha$ tel que $i^* < j < j^*$. Rappelons alors que $scm(m^*) \in \alpha_{j^*}$ mais que $scm(m^*) \notin \alpha_{i^*}$, et que $sm(m^*) \in \alpha_{i^*}$ mais que $sm(m^*) \notin \alpha_{j^*}$. Puisque α_{i^*} n'a pas été couplé, tout arc $m \in M$ doit satisfaire $sm(m) \in \alpha_{i^*}$ (en effet, les arcs sont propres et $scm(m) \notin \alpha_{i^*}$). De la même manière que dans le paragraphe précédent, nous pouvons aussi prouver que tout arc $m \in M$ doit aussi satisfaire à $scm(m) \in \alpha_{j^*}$ (les arcs étant propres, le contraire impliquerait une contradiction avec l'algorithme). Par conséquent, nous obtenons que l'ensemble $\{\alpha_1, \dots, \alpha_{i^*}\} \cup M \cup \{m^*\} \cup \{\alpha_{j^*}, \dots, \alpha_u\}$ induit une clique de taille $|C_\alpha| + 1$, ce qui est une contradiction, et le lemme est démontré. □

La correction de l'algorithme étant établie, nous analysons enfin sa complexité. Le calcul d'une clique maximum ne prend qu'un temps et espace $O(n)$ lorsque les arcs sont propres et ordonnés en entrée [10, 117]. Ensuite, la détermination de $\mathcal{M}$, qui ne prend qu'un temps linéaire dans le cas où $\omega(\mathcal{A}) \leq \lfloor n/2 \rfloor$, semble plus compliquée dans le cas $\omega(\mathcal{A}) > \lfloor n/2 \rfloor$. Le lemme suivant nous montre qu'en dépit des apparences, les couplages $\mathcal{M}_\alpha$ et $\mathcal{M}_\beta$ sont faciles à obtenir. Soit $B_\alpha = (X, Y, E)$ le graphe biparti avec $X = C_\alpha$, $Y = m(C_\alpha)$ et $E = \{(\alpha_j, A_i) \,|\, \alpha_j \in C_\alpha, A_i \in m(C_\alpha) \text{ et } \alpha_j \cap A_i = \emptyset\}$. Le graphe biparti B_β se définit de la même façon avec les ensembles C_β et $m(C_\beta)$.

Lemme 3.5 *Les graphes bipartis B_α et B_β sont proprement convexes.*

Preuve. Nous démontrerons seulement l'assertion pour le graphe biparti B_α, la preuve étant symétrique pour B_β.

Les arcs de C_α ont été ordonnés afin que leurs extrémités apparaissent dans le sens des aiguilles d'une montre dans l'ordre circulaire. Faisons de même pour l'ensemble $m(C_\alpha)$. Considérons maintenant le voisinage d'un sommet de X correspondant à l'arc $\alpha_j \in C_\alpha$ $(1 \leq j \leq u)$. Notons a_j le plus petit indice d'un arc $m \in m(C_\alpha)$ tel que $scm(m) \notin \alpha_j$ et b_j le plus grand indice d'un arc $m \in m(C_\alpha)$ tel que $sm(m) \notin \alpha_j$. Nous observons alors que l'intervalle $[a_j, \dots, |m(C_\alpha)|] \cap [1, \dots, b_j]$ correspond aux indices des arcs de $m(C_\alpha)$ qui peuvent être couplés à α_j. Si $i \leq j$, alors cet intervalle n'est pas vide et correspond exactement à $[a_j, \dots, b_j]$. Pour conclure, prenons un arc $\alpha_{j'} \in C_\alpha$ tel que $j' > j$. Les arcs étant propres, nous avons que $a_j \leq a_{j'}$ et $b_j \leq b_{j'}$. Par conséquent, B_α est proprement convexe. □

D'après le corollaire 2.4, un couplage maximum $\mathcal{M}_\alpha$ (resp. $\mathcal{M}_\beta$) peut donc être déterminé par un simple balayage des arcs de C_α et $m(C_\alpha)$, ceux-ci étant correctement ordonnés.

Théorème 3.2 *L'algorithme* 2-Ordo-Arcs-Propres *retourne en temps et espace* $O(n)$ *une solution optimale au problème* 2-Ordo *pour un ensemble* $\mathcal{A}$ *d'arcs propres, ordonné en entrée.*

Puisqu'une représentation par arcs propres ordonnée peut être obtenue en temps et espace linéaire [39] à partir d'un graphe d'arcs propres, nous obtenons les deux corollaires suivants.

Corollaire 3.3 *Le problème* Ordo *peut être résolu en temps et espace linéaire pour les graphes d'arcs propres lorsque* $k = 2$.

Corollaire 3.4 *Le problème du couplage maximum peut être résolu en temps et espace* $O(n)$ *pour les compléments de graphes d'arcs propres, étant donnée en entrée une représentation par arcs propres ordonnée du graphe.*

En accord avec la discussion précédente (et en particulier grâce au lemme 3.4), nous obtenons aussi deux corollaires assez inattendus.

Corollaire 3.5 *Soit* G *un graphe d'arcs propres. Si* $\omega(G) \geq \lceil n/2 \rceil$, *alors l'égalité* $\omega(G) = \chi(G)$ *est satisfaite.*

Corollaire 3.6 *Pour tout graphe* G *d'arcs propres, l'égalité* $\chi(G, 2) = \max\{\omega(G), \lceil n/2 \rceil\}$ *est satisfaite.*

Lorsque G est un graphe d'arcs propres *parfaits*, le corollaire 3.2 devient $\chi(G, k) = \max\{\omega(G), \lceil n/2 \rceil\}$ quel que soit k. Aussi nous pensons pouvoir étendre ce type d'approche pour résoudre en temps et espace linéaire le problème Ordo restreint aux graphes d'arcs propres parfaits.

3.2 Le cas des graphes de tolérances bornées

Bodlaender et Jansen [14] établissent la difficulté du problème Ordo pour les graphes d'intervalles en effectuant une réduction depuis le problème Couplage Numérique 3-D [60]. Ce type de réduction avait déjà été utilisé par Jansen [98] pour établir la difficulté d'un problème d'ordonnancement restreint aux ordres d'intervalles. À notre tour, nous nous inspirons de cette technique de preuve pour montrer que le problème 3-Ordo est $\mathcal{NP}$-difficile pour une classe de graphes très proche de celle des graphes d'intervalles.

Théorème 3.3 *Lorsque* k *est un paramètre fixé supérieur ou égal à trois, le problème* Ordo *reste* $\mathcal{NP}$*-difficile pour les graphes de tolérances bornées, même si le plus grand stable dans le graphe est de taille* $k + 1$ *et que tout cycle de longueur supérieure ou égale à cinq possède deux cordes.*

Preuve. Une instance du problème Couplage Numérique 3-D est donnée par trois ensembles disjoints $W = \{w_1, \ldots, w_m\}$, $X = \{x_1, \ldots, x_m\}$ et $Y = \{y_1, \ldots, y_m\}$ contenant chacun m éléments, la taille $s(a) \in \mathbb{N}$ pour chaque élément $a \in W \cup X \cup Y$, et une borne Z tel que $\sum_{a \in W \cup X \cup Y} s(a) = mZ$. La question est alors de décider si $W \cup X \cup Y$ admet une partition en m ensembles disjoints $\{A_i\}_{i=1,\ldots,m}$ tel que chacun de ces ensembles contienne exactement un élément provenant de chaque ensemble W, X et Y et que $\sum_{a \in A_i} s(a) = Z$ pour $1 \leq i \leq m$. Le problème reste $\mathcal{NP}$-complet si $0 < s(a) < Z/2$ pour tout $a \in W \cup X \cup Y$ et $1 < m < Z$. Ceci peut être prouvé en transformant le problème original en un autre problème où l'assertion est vérifiée, en ajoutant la valeur $Z + m$ à chaque $a \in W \cup X \cup Y$ puis en posant $Z' = Z + 3(Z + m)$.

À partir d'une instance de ce problème de couplage numérique, nous construisons un graphe correspondant à une instance du problème 3-Ordo. Ce graphe est représenté par un ensemble d'intervalles, munis chacun d'une tolérance. Tous les intervalles sont ouverts et leurs extrémités sont entières ; de même, toutes les tolérances sont bornées et entières. Le graphe sous-jacent est donc un graphe de tolérances bornées ; il nous restera à démontrer que tout cycle de longueur supérieure ou égale à cinq dans celui-ci possède toujours deux cordes. Voici l'ensemble d'intervalles en question :

1. pour $1 \leq i \leq m$, soit m intervalles $a_{i,l} =]0, i+1[$ de tolérance $t(a_{i,l}) = 0$;
2. pour tout couple $w_i \in W$, $x_j \in X$, soit un intervalle $b_{i,j} =]i+1, w_i + x_j + jZ + 1[$ de tolérance $t(b_{i,j}) = 0$;
3. pour tout couple $x_j \in X$, $y_k \in Y$, soit un intervalle $c_{j,k} =](j+1)Z - y_k + 1, (m+3)Z + k + 1[$ avec la tolérance $t(c_{j,k}) = 2k + 1$;
4. pour $1 \leq k \leq m$, soit $(m-1)$ intervalles $d_{k,l} =](m+3)Z - k, (m+5)Z + 1[$ de tolérance $t(d_{k,l}) = 2k + 1$;
5. pour $1 \leq j \leq m$, soit $(m-1)$ intervalles $h_{j,l} =]0, jZ + 1[$ et $(m-1)$ intervalles $g_{j,l} =](j+1)Z, (m+3)Z + 1[$ de tolérance $t(g_{j,l}) = t(h_{j,l}) = 0$.

Un exemple de construction est donné sur la figure 3.3 avec $w_1 = 1$, $w_2 = 2$, $x_1 = 1$, $x_2 = 1$, $y_1 = 2$, $y_2 = 3$ et $Z = 5$, $m = 2$.

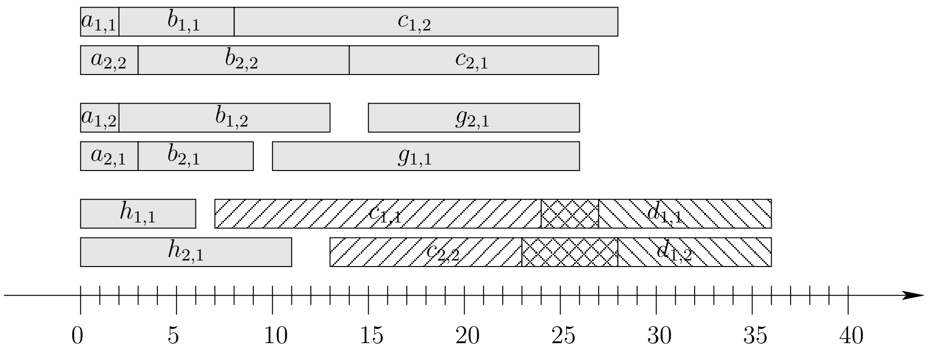


Fig. 3.3 – Un exemple de construction.

Nous noterons A, B, C, D, G, H les ensembles contenant respectivement tous les intervalles dénotés par la même lettre. Certains de ces ensembles forment une clique, indépendamment de l'instance considérée. Par construction, nous avons les ensembles $A \cup H$, $B \cup H$, $C \cup G$, $D \cup G$. En effet, les intervalles des ensembles A, B, G, H ne tolérant aucune intersection, il suffit de vérifier que les ensembles C et D induisent chacun une clique. Les intervalles de C (resp. D) recouvrent tous la portion de droite $[(m+1)Z+1, (m+3)Z+1)]$ (resp. $[(m+3)Z, (m+5)Z])$. La longueur de cette portion étant supérieure au maximum de leurs tolérances ($2Z > 2m+1$), les intervalles de C (resp. D) induisent effectivement une clique. Il est à noter que n'importe quel stable est de taille au plus quatre dans ce graphe. Ensuite, le cardinal des différents ensembles est donné par $|A| = |B| = |C| = m^2$, $|D| = |G| = |H| = m(m-1)$. Cela fait au total $6m^2 - 3m$ intervalles. Ainsi, le problème consiste à déterminer une partition du graphe de tolérances en $2m^2 - m$ stables de taille au plus trois, ce qui revient à déterminer une solution optimale au problème 3-Ordo pour ce même graphe. En fait, chaque stable de la partition devra être exactement de taille trois.

Considérons un stable U de taille trois contenant un intervalle $h \in H$. La seule possibilité pour compléter ce stable U est de choisir un intervalle $c \in C$ et un intervalle $d \in D$. De même, pour un stable U contenant $g \in G$, nous ne pourrons prendre qu'un intervalle $b \in B$ et un intervalle $a \in A$. Après suppression de ces intervalles, il ne restera donc plus que m éléments dans A, B et C. À présent, nous allons étudier plus en détail la composition des stables appartenant à une solution optimale. Pour cela, nous nous intéressons aux couplages disjoints entre les intervalles provenant des ensembles suivants : A et B, $B \cup H$ et $C \cup G$, C et D.

Cas (a) : les ensembles A et B. Ces deux ensembles contiennent chacun m^2 intervalles. D'après la discussion précédente, chaque intervalle de A doit être couplé à un intervalle de B de façon à appartenir à un même stable de la partition optimale. Nous allons montrer que les sommets $a_i \in A$ et $b_{i',j} \in B$ appartiennent à un même stable si et seulement si $i = i'$. Autrement dit, la partition optimale ne contient pas de stable U avec $\{a_i, b_{i',j}\} \subset U$ si $i \neq i'$. Pour cela, admettons que ce ne soit pas le cas. Nous avons m^2 stables contenant chacun un et un seul élément de A et de B. Soit $a_i \in A$ l'intervalle de plus petit indice i qui appartient à un stable contenant un intervalle $b_{i',j}$ avec $i \neq i'$. Si $i > i'$, alors les intervalles en question se chevauchent par construction, ce qui entraîne une contradiction. Voyons maintenant le cas où $i < i'$. Par hypothèse, les intervalles de A d'indice inférieur à i sont correctement couplés à un intervalle de B. Parmi les m intervalles de B ayant comme premier indice i, il en existe donc au moins un qui est couplé avec un intervalle $a_{i''}$ avec $i'' > i$. Comme ces intervalles se chevauchent par construction, nous obtenons une nouvelle contradiction.

Cas (b) : les ensembles $B \cup H$ et $C \cup G$. Ces deux ensembles contiennent chacun $2m^2 - m$ intervalles et induisent chacun une clique. Par conséquent, chaque stable de la partition optimale doit inclure un élément de $B \cup H$ et un élément de $C \cup G$. Nous pouvons observer que l'intervalle $b_{i,j}$ chevauche l'intervalle $g_{j',l}$ si $j' < j$ et que l'intervalle $c_{j,k}$ chevauche l'intervalle $h_{j',l}$ si $j' > j$. Ainsi, nous retrouvons la même configuration qu'en (a) et pouvons prouver de manière similaire qu'un couple d'intervalles $\{b_{i,j}, c_{j',k}\}$, $\{b_{i,j}, g_{j',l}\}$ ou $\{h_{j,l}, c_{j',k}\}$ appartient à un même stable de la solution optimale si et seulement $j = j'$.

Cas (c) : les ensembles C et D. Pour toute paire d'intervalles $c_{j,k}$ et $d_{k',l}$ ($1 \leq k, k' \leq m$), nous avons $|c_{j,k} \cap d_{k',l}| = k + k' + 1$. Les tolérances de ces deux intervalles étant respectivement égales à $2k+1$ et $2k'+1$, ceux-ci ne pourront être couplés que si $k + k' + 1 \leq 2k + 1$ et $k + k' + 1 \leq 2k' + 1$, c'est-à-dire $k = k'$.

Les résultats de l'analyse des trois cas (a), (b) et (c) sont récapitulés sur la figure 3.4. Chaque ligne du tableau représente les paires d'intervalles que doivent contenir les stables d'une solution optimale. La première ligne signifie par exemple qu'il n'existe aucun stable dans la solution contenant une paire d'intervalle $\{a_i, b_{i',l}\}$ avec $i \neq i'$.

premier intervalle	second intervalle
$a_{i,-}$	$b_{i,-}$
$b_{-,j}$ ou $h_{j,-}$	$c_{j,-}$ ou $g_{j,-}$
$c_{-,k}$	$d_{k,-}$

FIG. 3.4 – Un récapitulatif des résultats de l'analyse des trois cas.

À présent, supprimons de la solution tous les stables U contenant $h \in H$ ou $g \in G$. Puisque chaque intervalle $g_{j,-}$ (resp. $h_{j,-}$) est couplé à un intervalle $b_{-,j}$ (resp. $c_{j,-}$), il reste dans B (resp. C) un et un seul intervalle $b_{-,j}$ (resp. $c_{j,-}$) pour tout $j = 1, \dots, m$. De la même façon, les stables contenant des intervalles de G et de B (resp. de H et de C) sont nécessairement complétés par des intervalles de A (resp. D). Par conséquent, il reste aussi un intervalle $a_{i,l}$ pour tout $i = 1, \dots, m$, mais aucun intervalle de D (puisque ce dernier comporte autant d'éléments que H). En définitive, seuls m stables $U_i = \{a_i, b_{i,j}, c_{j,k}\}$ demeurent après suppression de tous les stables U.

Nous sommes enfin en mesure de prouver qu'il existe une partition de $W \cup X \cup Y$ en m ensembles $\{A_i\}_{i=1,\dots,m}$ contenant chacun un élément différent de W, X et Y avec $\sum_{a \in A_i} s(a) = Z$ pour $1 \leq i \leq m$ si et seulement si le graphe de tolérances sous-jacent admet une partition en $2m^2 - m$ stables de taille au plus trois.

Soit $U_1, \dots, U_{2m^2-m}$ une telle partition. D'après l'analyse menée précédemment, nous pouvons admettre sans perte de généralité que les m premiers stables de la partition sont de la forme $U_i = \{a_i, b_{i,j}, c_{j,k}\}$ de telle sorte que les indices i, j et k ($1 \leq i, j, k \leq m$) n'apparaissent qu'une et une seule fois. Comme U_i est un stable, nous avons $w_i + x_j + jZ + 1 \leq (j+1)Z - y_k + 1$ et donc $w_i + x_j + y_k \leq Z$. Puisque chaque indice apparaît exactement une fois, nous obtenons que $\sum_{i=1}^m w_i + \sum_{j=1}^m x_j + \sum_{k=1}^m y_k = mZ$. Ainsi, $w_i + x_j + y_k = Z$ et les ensembles $A_i = \{w_i, x_j, y_k\}$ définis à partir des stables U_i solutionnent le problème de couplage numérique.

Prouvons l'implication dans le sens inverse. Soit $A_i = \{w_i, x_j, y_k\}$ ($1 \leq i \leq m$) les m ensembles tel que $\sum_{a \in A_i} s(a) = Z$. À partir de ces m ensembles, nous allons construire une partition optimale du graphe de tolérances en stables de taille au plus trois. Tout d'abord, pour tout $i = 1, \dots, m$, définissons un stable $U_i = \{a_i, b_{i,j}, c_{j,k}\}$. Les intervalle a_i et $b_{i,j}$ ne se chevauchent pas, tout comme les intervalles $c_{j,k}$ et d_k. Pour prouver que $b_{i,j}$ et $c_{j,k}$ ne se chevauchent pas non plus, comparons respectivement leurs extrémités droite et gauche. L'égalité $w_i + x_j + y_k = Z$ nous donne $w_i + x_j + jZ + 1 \leq (j+1)Z - y_k + 1$. Ainsi, chaque ensemble U_i induit effectivement un stable. Notons maintenant B' l'ensemble des sommets de B qui ne sont pas couverts par les U_i ($1 \leq i \leq m$). Pour chaque $b_{i,j} \in B'$, définissons un ensemble contenant les intervalles $a_{i,l}, b_{i,j}$ et $g_{j,l'}$. Clairement, chacun de ces ensembles induit un stable. De plus, la construction est correcte, puisque que chaque indice i et j apparaît seulement $(m-1)$ fois dans B'. Notons enfin C' l'ensemble des sommets de C qui ne sont pas couverts et définissons un ensemble contenant les sommets $h_{j,l}, c_{j,k}, d_{k,l'}$ pour chaque $c_{j,k} \in C'$. Les tolérances des intervalles $c_{j,k}$ et $d_{k,l'}$ sont telles que leurs sommets correspondants dans le graphe de tolérances ne sont pas connectés. D'un autre côté, les intervalles $d_{k,l'}$ ne peuvent pas chevaucher les intervalles $h_{j,l}$. Par conséquent, chacun de ces ensembles induit aussi un stable. À présent, tous les sommets sont couverts par $2m^2 - m$ stables.

Cette réduction établit la $\mathcal{NP}$-difficulté du problème Ordo pour les graphes de tolérances bornées lorsque $k = 3$. Pour les valeurs $k > 3$, il suffit de reprendre la construction décrite ci-dessus et d'y ajouter $(k-4)$ cliques disjointes K_m pour obtenir le résultat. Le plus grand stable dans le graphe sera alors de taille $k + 1$. Pour clore définitivement la preuve, nous montrons que tous les cycles de longueur supérieure ou égale à cinq dans celui-ci contiennent deux cordes.

Les seuls sommets pouvant induire des cycles ne vérifiant pas la condition en question sont ceux qui ont une tolérance non nulle, c'est-à-dire les intervalles de $C \cup D$. En effet, si l'on considère le sous-graphe induit par tous les intervalles exceptés ceux de C (resp. D), alors celui-ci est clairement un graphe d'intervalles et donc ne contient aucun cycle de longueur supérieure ou égale à quatre sans corde. Admettons maintenant l'existence d'un cycle ne vérifiant pas la condition. Étant de longueur supérieure à cinq, celui-ci contient toujours au moins trois sommets provenant soit de l'ensemble C, soit de l'ensemble D. S'il en contient quatre ou plus, alors ceux-ci induisent au moins deux cordes (puisque C et D induisent chacun une clique). Il en est de même s'il en contient trois et que ceux-ci n'apparaissent pas consécutivement sur le cycle. Par conséquent, il nous reste à traiter le cas où le cycle contient exactement trois sommets d'un même ensemble et que ces trois sommets apparaissent consécutivement sur le cycle.

Sans perte de généralité, admettons que les trois sommets en question appartiennent à l'ensemble C et notons $\{c_{j_1,k_1}, c_{j_2,k_2}, c_{j_3,k_3}, d_{k_4,l_4}, d_{k_5,l_5}, c_{j_1,k_1}\}$ le cycle en question. L'ensemble C formant une clique, les sommets c_{j_1,k_1} et c_{j_3,k_3} sont reliés par une corde. Nous avons précédemment vu que les sommets $c_{j,k} \in C$ et $d_{k',l} \in D$ du graphe de tolérances ne sont pas reliés par une arête à la seule condition que $k = k'$. Le sommet c_{j_1,k_1} étant relié à d_{k_5,l_5} mais pas à d_{k_4,l_4}, nous avons que $k_1 \neq k_5$ et $k_1 = k_4$ (de manière symétrique, nous avons avec le sommet c_{j_3,k_3} que $k_3 = k_5$ et $k_3 \neq k_4$). De là, nous en déduisons que $k_4 \neq k_5$, et par conséquent que le sommet c_{j_2,k_2} est nécessairement connecté à un des deux sommets d_{k_4,l_4} ou d_{k_5,l_5}, ce qui complète la preuve. □

Remarque. Les graphes concernés par le théorème satisfont les conditions suivantes : *tout cycle de longueur supérieure ou égale à cinq possède deux cordes* et *le complément du graphe est transitivement orientable*. En effet, tout complément d'un graphe de tolérances bornées est aussi un graphe de comparabilité. D'un autre côté, les graphes d'intervalles sont précisément les graphes qui satisfont les conditions suivantes : *tout cycle de longueur supérieure ou égale à quatre possède une corde* et *le complément du graphe est transitivement orientable*. Les graphes visés par le théorème diffèrent essentiellement de ces derniers par le fait qu'ils peuvent induire, sous certaines conditions, des cycles de longueur quatre.

Corollaire 3.7 *Lorsque k est un paramètre fixé supérieur ou égal à trois, le problème* Ordo *est $\mathcal{NP}$-difficile pour les graphes suivants, même si leur complément est de comparabilité :*

. *les graphes de tolérances ;*

. *les graphes de trapèzes ;*
. *les graphes faiblement triangulés ;*
. *les graphes de Meyniel.*

Remarque. Le corollaire 3.7 généralise le résultat de Lonc [113] qui établit que le problème Ordo est $\mathcal{NP}$-difficile pour les compléments de graphes de comparabilité, même lorsque k est un paramètre fixé supérieur ou égal à trois. D'un autre côté, alors que le problème k-Ordo peut être résolu en temps et espace polynomial pour les graphes parfaits de stabilité au plus k, le théorème 3.3 établit aussi que le même problème est $\mathcal{NP}$-difficile pour les graphes de tolérances bornées (et *a fortiori* pour les graphes parfaits) de stabilité au moins $k+1$.

3.3 Synthèse des résultats

Pour conclure ce chapitre, nous présentons une synthèse des résultats obtenus concernant la complexité du problème d'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles, les graphes d'arcs circulaires et les graphes de tolérances.

	Graphes d'intervalles propres	Graphes à seuil	Graphes d'intervalles
$k=2$	$O(n+m)$	$O(n+m)$	$O(n+m)$
$k=3$	$O(n+m)$	$O(n+m)$	*ouvert*
$k\geq 4$	$O(n+m)$	$O(n+m)$	$\mathcal{NP}$-difficile [14]

Fig. 3.5 – La complexité du problème Ordo pour les graphes d'intervalles.

	Graphes d'arcs propres	Graphes d'arcs
$k=2$	$O(n+m)$	couplage maximum [122, 74]
$k=3$	$O(n^2)$	*ouvert*
$k\geq 4$	$O(n^2)$	$\mathcal{NP}$-difficile (même si parfaits ou de Helly) [14]

Fig. 3.6 – La complexité du problème Ordo pour les graphes d'arcs circulaires.

	Graphes de tolérances propres	Graphes de tolérances
$k=2$	couplage maximum [122, 74]	couplage maximum [122, 74]
$k\geq 3$	*ouvert* (même si unitaires et bornées)	$\mathcal{NP}$-difficile (même si bornées)

Fig. 3.7 – La complexité du problème Ordo pour les graphes de tolérances.

Les questions suivantes restent donc posées : le problème 3-Ordo peut-il être résolu en temps et espace polynomial pour les graphes d'intervalles ou d'arcs circulaires ? Quel est la complexité du problème Ordo lorsque l'on se restreint aux graphes de tolérances unitaires et bornées ? La recherche d'algorithmes directs pour le problème 2-Ordo restreint aux graphes d'arcs ou de tolérances est aussi une question intéressante.

Chapitre 4

Une condition suffisante pour l'optimalité

En dépit de quelques résultats positifs, le bilan du travail de classification que nous avons mené jusqu'à présent concernant la complexité du problème d'ordonnancement avec exclusion mutuelle est plutôt négatif. Pour une majorité de classes de graphes – et en particulier celles qui nous intéressent (graphes d'intervalles, d'arcs ou de tolérances), le problème reste $\mathcal{NP}$-difficile même lorsque l'on se restreint à de petites valeurs de k fixées. Des algorithmes polynomiaux, et même linéaires, ont été proposés pour traiter le problème lorsque l'on se restreint par exemple aux graphes d'intervalles propres, aux graphes de seuil, ou encore aux graphes d'arcs propres. Malheureusement, ces cas particuliers ne couvrent pas suffisamment de cas en pratique : de nombreux problèmes font intervenir des intervalles ou des arcs qui se recouvrent entièrement.

Se restreindre seulement à des classes de graphes n'étant pas satisfaisant d'un point de vue pratique, nous nous proposons dans ce chapitre d'attaquer le problème sous un nouvel angle en abordant la question suivante : peut-on exhiber des conditions intéressantes (en pratique) dans lesquelles le problème Ordo peut être résolu efficacement ? À notre connaissance, aucune étude n'est encore parue à ce sujet. Nous tentons ici d'apporter une première réponse à cette question, en étudiant une propriété que nous avons vu apparaître au travers du petit lemme 2.2 de découpage, lors de l'étude du problème 2-Ordo pour les graphes d'intervalles. D'après ce petit lemme, les graphes d'intervalles possèdent la propriété suivante : si l'on dispose d'une partition en stables où chaque stable est de taille au moins deux, alors cette partition peut être "découpée" en stables de taille deux. Plus tard, nous avons retrouvé cette propriété lors de l'étude du problème pour les graphes d'arcs propres. En effet, il ressort de la preuve de validité de l'algorithme Ordo-Arcs-Propres que les graphes d'arcs propres possèdent la propriété suivante.

Propriété de redécoupage : *Soit G un graphe et k un entier. Si le graphe G admet une coloration tel que chaque couleur apparaît au moins k fois, alors $\chi(G,k) = \lceil n/k \rceil$.*

Cette propriété a un premier intérêt pratique qui le suivant. Le paramètre k, qui concerne ici le nombre de processeurs ou de machines disponibles, est généralement assez petit comparé à n, le nombre de tâches à ordonnancer. Ainsi, dans bien des cas, une simple heuristique de coloration permet d'obtenir une partition en stables de taille au moins k. Dans certains problèmes comme la planification de personnel, ce sont des propriétés structurelles qui garantissent l'existence d'une telle partition. Prenons par exemple le cas de la planification des chauffeurs de bus municipaux ou bien des personnels d'aérogares – problèmes résolus par la firme Prologia du Groupe Air Liquide [7] : les rotations des bus ou des avions induisent très souvent des paquets de tâches consécutives de taille supérieure à k (pour des valeurs de k comme $k \leq 5$).

Le second intérêt que peut avoir cette propriété en pratique intervient lors de l'allocation d'un ou plusieurs nouveaux processeurs. En effet, lorsqu'un ordonnancement parfait des tâches a été réalisé sur chaque processeur (c'est-à-dire que n/k tâches sont allouées à chaque processeur, n étant un multiple de k), il n'est pas évident que l'on puisse obtenir un gain de productivité en ajoutant un nouveau processeur, à cause du graphe d'exclusion mutuelle. Lorsque le graphe en question a la propriété de redécoupage, nous avons la garantie suivante.

Lemme 4.1 ("Qui peut le plus peut le moins") *Soit G un graphe ayant la propriété de redécoupage. Si G admet une partition exacte en stables de taille k, alors G admet aussi une partition optimale en $\lceil n/k' \rceil$ stables de taille au plus k', quel que soit $k' > k$.*

Enfin, cette propriété de redécoupage peut trouver des applications dans la conception d'algorithmes polynomiaux d'approximation pour le problème Ordo, comme nous le verrons au chapitre suivant (voir l'algorithme 2-Approx-Planif, p. 110).

Au travers des sections suivantes, nous montrons que la propriété de redécoupage est partagée par les *graphes sans $K_{1,3}$* (qui généralisent les graphes d'intervalles propres et d'arcs propres), les *graphes d'intervalles* et les *graphes d'arcs*, et même les *graphes triangulés* lorsque $k \leq 4$. De plus, des algorithmes efficaces sont présentés pour traiter le problème Ordo lorsqu'une partition en stables de taille supérieure à k est donnée en entrée. Ces résultats sont d'autant plus inattendus qu'un contre-exemple très simple peut être trouvé ne satisfaisant pas cette propriété [1].

Contre-exemple et premières difficultés

Considérons le graphe biparti complet $K_{k+1,k+1}$, avec $k \geq 2$ (voir la figure 4.1). Nous pouvons observer que $\chi(K_{k+1,k+1}, k) = 4$ puisque deux sommets appartenant à deux stables différents ne peuvent pas être couplés. La borne inférieure étant $\lceil n/k \rceil = \lceil (2k+2)/k \rceil = 3$, nous avons donc la proposition suivante.

Proposition 4.1 (Contre-exemple) *Le graphe biparti $K_{k+1,k+1}$ ne partage pas la propriété de redécoupage, quel que soit $k \geq 2$.*

Nous pouvons remarquer que le graphe $K_{k+1,k+1}$ est un graphe de permutation (en effet, celui-ci est transitivement orientable car biparti et son complément l'est aussi car composé

[1] Les résultats présentés dans ce chapitre apparaissent en partie dans [68] (en anglais).

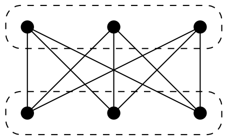


FIG. 4.1 – Le graphe biparti complet $K_{3,3}$.

de deux cliques disjointes). Nous rappelons que les graphes de permutation forment une sous-classe des graphes de tolérances bornées. D'un autre côté, $K_{k+1,k+1}$ est trivialement un graphe biparti proprement convexe ou encore un graphe de Meyniel (car ne contenant pas de cycle impair).

Corollaire 4.1 (Contre-exemple) *Il existe dans les classes suivantes des graphes qui ne partagent pas la propriété de redécoupage, quel que soit $k \geq 2$:*

. *les graphes de permutation et les graphes de cordes ;*
. *les graphes de comparabilité et leurs compléments ;*
. *les graphes de tolérances bornées et les graphes de trapèzes ;*
. *les graphes de graphes faiblement triangulés ;*
. *les graphes bipartis proprement convexes ;*
. *les graphes de Meyniel.*

Du point de vue de la complexité, la partie semble tout aussi compromise. Dans leur papier, Bodlaender et Jansen [14] abordent la complexité du problème ORDO pour les graphes bipartis et montrent que le problème de la partition d'un graphe biparti en trois stables de taille au plus k est $\mathcal{NP}$-complet. Nous avons alors remarqué que le graphe biparti utilisé lors de la réduction est composé de deux stables de taille supérieure à k, avec $k > 5$.

Proposition 4.2 (Bodlaender et Jansen, 1995) *Le problème de la partition d'un graphe biparti en trois stables de taille au plus k est $\mathcal{NP}$-complet lorsque $k > 5$, même si les deux stables composant le graphe sont de taille supérieure à k.*

Corollaire 4.2 ($\mathcal{NP}$-difficulté) *Le problème de la partition minimum en stables de taille au plus k est $\mathcal{NP}$-difficile pour les graphes appartenant aux classes suivantes lorsque $k > 5$, même si le graphe admet une partition en stables de taille supérieure à k :*

. *les graphes bipartis ;*
. *les graphes de comparabilité ;*
. *les graphes de graphes faiblement triangulés ;*
. *les graphes de Meyniel.*

4.1 Une caractérisation des graphes sans $K_{1,3}$

Les graphes sans $K_{1,3}$ ne possèdent pas $K_{k+1,k+1}$ comme sous-graphe induit quel que soit $k \geq 2$. Pour autant, les graphes sans $K_{1,3}$ partagent-ils la propriété de redécoupage ? Au travers du théorème suivant, nous montrons que ces graphes possèdent en fait une propriété plus forte que celle-ci.

Ce théorème, qui caractérise de façon algorithmique les graphes sans $K_{1,3}$, a été établi en d'autres termes par De Werra [42] alors que celui-ci étudiait des problèmes d'emploi du temps. Le problème de la *coloration équitable* d'un graphe, lié aux problèmes d'emploi du temps (consulter [41, 42] pour plus de détails), consiste à déterminer une coloration de ce graphe tel que le nombre de sommets de chaque couleur soit le même à un près (voir [13] pour une revue de résultats sur le sujet). Comme l'ont justement remarqué Hansen *et al.* [83], il existe une relation étroite entre le problème de coloration équitable et le problème d'ordonnancement avec exclusion mutuelle. Dans cette section, nous unifions et complétons les travaux de De Werra [42] et de Hansen *et al.* [83] sur le sujet. En particulier, un algorithme muni d'une structure de données spéciale est présenté pour résoudre efficacement le problème Ordo ainsi que le problème de la coloration équitable pour les graphes sans $K_{1,3}$, étant donnée en entrée une coloration minimum du graphe. Ces résultats offrent dans le même temps une généralisation des corollaires 2.12, 2.13, 3.1 et 3.2, concernant le problème Ordo pour les graphes d'intervalles propres et les graphes d'arcs propres.

Note. Curieusement, aucune allusion aux travaux de De Werra [42] et de Hansen *et al.* [83] n'est faite dans la très large revue de résultats proposée récemment par Faudree *et al.* [49] sur les graphes sans $K_{1,3}$, ni même dans celles de Jansen [100] et Bodlaender et Fomin [13] concernant le problème d'ordonnancement avec exclusion mutuelle.

Théorème 4.1 (De Werra, 1985 – Hansen *et al.*, 1993) *Pour un graphe G, les conditions suivantes sont équivalentes :*

(1) le graphe G est sans $K_{1,3}$;

(2) toute composante connexe d'un sous-graphe de G induit par deux stables disjoints est isomorphe à une chaîne ou bien un cycle pair ;

(3) pour tout sous-graphe $G' \subseteq G$ induit par n' sommets, l'égalité

$$\chi(G', k) = \max\{\chi(G'), \lceil n'/k \rceil\}$$

est satisfaite quel que soit $k \geq 1$;

(4) pour tout sous-graphe $G' \subseteq G$ induit par n' sommets, G' admet une q-coloration équitable quel que soit $q \geq \chi(G')$.

Preuve. (1) $\Rightarrow$ (2). Tout sous-graphe induit par deux stables disjoints est biparti et ne contient donc pas de cycle impair. Dans le cas présent, ce graphe biparti est aussi sans $K_{1,3}$, ce qui implique que tous ces sommets sont de degré au plus deux. Par conséquent, chacune de ces composantes connexes ne peut être isomorphe qu'à une chaîne ou bien un cycle pair.

Les implications (2) $\Rightarrow$ (3) et (2) $\Rightarrow$ (4) sont établies pour le graphe G seulement, puisque pouvant être étendues immédiatement à tout sous-graphe induit G' par hérédité du caractère sans $K_{1,3}$.

(2) $\Rightarrow$ (3). La preuve repose sur le lemme 3.2 qui reste valide pour les graphes sans $K_{1,3}$, d'après l'assertion (2). L'algorithme Raffiner-Coloration, que nous rappelons ci-dessous,

peut donc être utilisé en l'état pour obtenir une coloration raffinée de G, tel que (a) chaque couleur apparaisse au plus k fois ou bien (b) chaque couleur apparaisse au moins k fois.

Algorithme RAFFINER-COLORATION ;
Entrée : une coloration minimum $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$ de G, un entier k ;
Sortie : une coloration $\mathcal{S}$ satisfaisant une des deux conditions (a) ou (b) ;
Début ;
  tant qu'il existe deux stables disjoints $S_u, S_v \in \mathcal{S}$ avec $|S_u| > k$ et $|S_v| < k$ **faire**
    $\mathcal{B} \leftarrow$ COMPOSANTES-CONNEXES(S_u, S_v) ;
    tant que $|S_u| > k$ et $|S_v| < k$ **faire**
      choisir une composante connexe $B_r \in \mathcal{B}$ tel que $|B_r^u| = |B_r^v| + 1$;
      échanger les sommets de S_u et S_v correspondant respectivement à B_r^u et B_r^v ;
  retourner $\mathcal{S}$;
Fin ;

Soit $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$ une coloration minimum de G raffinée à l'aide de l'algorithme RAFFINER-COLORATION. Dans le cas où $\mathcal{S}$ satisfait la condition (a), alors nous avons immédiatement $\chi(G,k) = \chi(G)$. Dans le cas où $\mathcal{S}$ satisfait la condition (b), nous montrons comment obtenir une partition de G en $\lceil n/k \rceil$ stables de taille au plus k. Notons que dans ce cas $n > k\chi(G)$.

Soit $|S_u| = \alpha_u k + \beta_u$ la taille d'un stable S_u $(1 \leq u \leq \chi(G))$, avec α_u un entier positif non nul et $0 \leq \beta_u \leq k-1$. Tout d'abord, extrayons de chaque stable S_u $(1 \leq u \leq \chi(G))$ $\alpha_u - 1$ stables de taille exactement k, plus un si $\beta_u = 0$. À l'issue de cette opération, notons χ^* le nombre de stables qui restent non vides et n^* le nombre de sommets dans ces χ^* stables. Clairement, chaque stable S_u non vide ne contient plus que $k + \beta_u$ sommets avec $\beta_u > 0$ et l'inégalité $n^* > k\chi^*$ est toujours vérifiée. À raison d'un seul par stable, extrayons maintenant $\lfloor n^*/k \rfloor - \chi^*$ stables de taille k, plus un stable de taille $n^* \bmod k$ si n^* n'est pas un multiple de k. De cette façon, le nombre de sommets restant dans les χ^* stables est exactement $k\chi^*$ et au moins deux stables S_u et S_v de la partition sont tels que $|S_u| > k$ et $|S_v| < k$. Par conséquent, une nouvelle application de l'algorithme RAFFINER-COLORATION sur cette partition (qui reste sans $K_{1,3}$) nous permet d'obtenir χ^* stables de taille exactement k. En définitive, nous n'avons extrait que des stables de taille k, sauf un de taille $n \bmod k$ si k ne divise pas n.

(2) $\Rightarrow$ (4). Soit $S_1, \ldots, S_q$ une q-coloration de G avec $q \geq \chi(G)$. Une q-coloration équitable s'obtient en utilisant l'algorithme RAFFINER-COLORATION dans lequel les conditions $|S_u| > k$ et $|S_v| < k$ dans les deux boucles **tant que** sont remplacées par $|S_u| > \lceil n/q \rceil$ et $|S_v| < \lfloor n/q \rfloor$. La validité de l'algorithme s'obtient alors avec les mêmes arguments que pour RAFFINER-COLORATION, à ceci près qu'ici, la q-coloration équitable est retournée après q tours de la boucle principale.

Les implications (3) $\Rightarrow$ (1) et (4) $\Rightarrow$ (1) étant immédiates (le graphe $K_{1,3}$, qui est 2-colorable, n'admet aucune partition en deux stables de taille deux), la preuve du théorème est complète. □

Corollaire 4.3 (Propriété de redécoupage) *Les graphes sans $K_{1,3}$ partagent la pro-*

priété de redécoupage, quel que soit k.

Questions de complexité

Dans cette partie, nous allons discuter des questions de complexité relatives au problème Ordo ainsi qu'au problème de coloration équitable pour les graphes sans $K_{1,3}$. Pour cela, nous commençons par analyser la complexité de la procédure Raffiner-Coloration qui joue, comme nous l'avons vu dans les preuves de l'implication (2) $\Rightarrow$ (3), un rôle central dans la recherche d'une solution optimale au problème Ordo.

Le point crucial de l'algorithme Raffiner-Coloration réside dans la recherche des composantes connexes du sous-graphe biparti induit par les deux stables disjoints S_u et S_v. Nous verrons que l'échange de sommets entre les deux stables est aussi un point délicat : son efficacité dépend fortement de la représentation des composantes connexes. Nous allons voir comment implanter de façon efficace ces deux étapes de l'algorithme à l'aide d'une structure de données appropriée. Nous considérons pour cela que l'entrée de l'algorithme est composée du graphe $G = (V, E)$ sans $K_{1,3}$ représenté par listes d'adjacence, d'une coloration minimum $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$ représentée par $\chi(G)$ tableaux (le tableau S_u contenant les sommets de couleur u), et de l'entier positif k. Précisons encore que la taille d'un tableau ou d'une liste pourra ici être obtenue en temps $O(1)$.

Cette structure, notée F, est un tableau de taille n. Pour chaque sommet $i \in V$, la cellule F_i du tableau possède trois champs : un entier F_i.couleur qui représente la couleur de i, c'est-à-dire l'indice du stable auquel il appartient, un entier F_i.rang qui représente le rang de i dans ce stable, et un tableau $F_i.S$ de taille $\chi(G)$ qui contient dans chaque cellule $F_i.S_u$ la liste (des indices) des sommets adjacents à i dans le stable S_u (la taille de cette liste sera donnée par $F_i.|S_u|$). D'après l'assertion (2) du théorème 4.1, le nombre de sommets stockés dans une liste $F_i.S_u$ ne doit pas excéder deux. Ainsi, l'espace requis par la structure F est borné par $O(\chi(G)n)$. Le remplissage de la structure peut être réalisé en temps $O(\chi(G)n)$ en calculant tout d'abord les champs F_i.couleur et F_i.rang pour chaque sommet $i \in V$ (par un simple balayage des ensembles $S_1, \ldots, S_{\chi(G)}$), puis en explorant la liste d'adjacence associée à chaque sommet $i \in V$ afin de remplir la liste $F_i.S$ (en utilisant notamment le champ couleur précédemment calculé).

Nous pouvons à présent établir la complexité de l'algorithme de raffinement. Après avoir identifié en temps $O(\chi(G))$ deux stables S_u et S_v tel que $|S_u| > k$ et $|S_v| < k$, l'algorithme fait appel à la procédure Composantes-Connexes, que nous allons décrire dans le détail cette fois-ci. Tout d'abord, les composantes connexes ne sont pas toutes stockées : ne sont retenues que $t = \min\{|S_u| - k, k - |S_v|\}$ chaînes paires dont les deux extrémités sont dans S_u. Les sommets de ces composantes connexes sont ensuite insérés dans les listes $\mathcal{B}^u$ ou $\mathcal{B}^v$ selon qu'ils proviennent de S_u ou S_v. La détermination des t composantes connexes utiles se fait comme suit. Le stable S_u est examiné afin d'y trouver des sommets n'ayant qu'un seul voisin dans S_v (ces sommets correspondent aux extrémités des chaînes du sous-graphe biparti induit par S_u et S_v). Lorsqu'un tel sommet est trouvé, il est marqué puis la chaîne dont il est l'extrémité est parcourue. Si la longueur de cette chaîne est paire, alors ces sommets sont stockés dans la structure $\mathcal{B} = \mathcal{B}^u \cup \mathcal{B}^v$. Tout ce travail peut être effectué en temps et espace $O(|S_u| + |S_v|)$ grâce à la structure F qui permet d'obtenir le voisinage d'un sommet du graphe biparti induit par $S_u \cup S_v$ en temps $O(1)$. Le procédure complète est détaillée ci-dessous. Par commodité, nous utilisons la notation $\bar{c}$ avec $\bar{c} = u$ si $c = v$ et $\bar{c} = u$ si $c = v$.

Algorithme COMPOSANTES-CONNEXES ;
Entrée : deux stables S_u et S_v tel que $|S_u| > k$ et $|S_v| < k$;
Sortie : l'ensemble $\mathcal{B} = \mathcal{B}^u \cup \mathcal{B}^v$ des composantes connexes utiles ;
Début ;
    $\mathcal{B}^u \leftarrow \emptyset$, $\mathcal{B}^v \leftarrow \emptyset$;
    $r \leftarrow 1$, $j \leftarrow 1$, $t \leftarrow \min\{|S_u| - k, k - |S_v|\}$;
    tant que $r \leq t$ **faire**
        soit i le $j^{ème}$ sommet du stable S_u ;
        si i n'est pas marqué et $F_i.|S_v| \leq 1$ **alors**
            $B^u \leftarrow \{i\}$, $B^v \leftarrow \emptyset$, $c \leftarrow u$, $i_{prec} \leftarrow \emptyset$;
            tant que $F_i.S_{\bar{c}} \setminus \{i_{prec}\} \neq \emptyset$ **faire**
                choisir dans $F_i.S_{\bar{c}}$ le sommet $i_{cour} \neq i_{prec}$;
                $i_{prec} \leftarrow i$, $i \leftarrow i_{cour}$, $c \leftarrow \bar{c}$, $B^c \leftarrow B^c \cup \{i\}$;
            si $|B^u| > |B^v|$ **alors**
                copier les sommets de B^u et B^v dans $\mathcal{B}^u$ et $\mathcal{B}^v$;
                marquer le sommet i dans S_u ;
                $r \leftarrow r + 1$;
        $j \leftarrow j + 1$;
    retourner $\mathcal{B} \leftarrow \mathcal{B}^u \cup \mathcal{B}^v$;
Fin ;

L'échange des sommets entre stables est moins évident que dans le cas des graphes d'arcs propres. Tout d'abord, les sommets à échanger sont marqués dans les tableaux S_u et S_v à partir des ensembles de sommets $\mathcal{B}^u \cup \mathcal{B}^v$ (cela prend un temps $O(|S_u| + |S_v|)$ en utilisant le champ rang de la structure F). Pour chaque sommet $i \in V$, nous pouvons maintenant procéder à l'échange des sommets dans les listes $F_i.S_u$ et $F_i.S_v$. L'inspection de ces listes ne prenant qu'un temps constant, l'échange peut être réalisé en temps $O(1)$ (en utilisant le champ rang de la structure F et le marquage des sommets à échanger dans S_u et S_v). Par conséquent, cette première mise à jour de la structure F peut se faire en temps $O(n)$. Suite à cela, les tableaux S_u et S_v peuvent être mis à jour en temps $O(|S_u| + |S_v|)$, ainsi que les champs F_i.couleur et F_i.rang des sommets $i \in V$ concernés par l'échange.

En résumé, l'utilisation de la structure F, dont la construction nécessite $O(\chi(G)n)$ temps et espace, permet d'implanter la boucle principale de l'algorithme RAFFINER-COLORATION de façon à ne consommer qu'un temps et espace $O(n)$ à chaque tour de boucle. Comme $\chi(G)$ tours de boucle suffisent au raffinement d'une coloration, l'algorithme RAFFINER-COLORATION s'exécute donc en $O(\chi(G)n)$ temps et espace.

Proposition 4.3 *Le problème* ORDO *peut être résolu en temps et espace $O(\chi(G)n)$ pour les graphes sans $K_{1,3}$, étant donnée en entrée une coloration minimum du graphe.*

Preuve. La preuve de l'implication (2) $\Rightarrow$ (3) du théorème 4.1 fournit l'algorithme. Après le remplissage de la structure F, la procédure de raffinement est exécutée une première fois. Si tous les stables de la coloration raffinée sont de taille au plus k, alors cette coloration forme une solution optimale au problème Ordo (puisque de cardinal $\chi(G)$). Si tous les stables de la coloration sont de taille supérieure k, alors la coloration peut être nettoyée en temps linéaire avant d'exécuter à nouveau la procédure de raffinement. La solution ainsi obtenue est aussi optimale (puisque de cardinal $\lceil n/k \rceil$). D'après la discussion précédente, la complexité de la méthode tout entière ne prend qu'un temps et espace $O(\chi(G)n)$. □

Remarque. Nous rappelons que le problème de la coloration minimum, et par conséquent le problème Ordo, sont $\mathcal{NP}$-difficiles pour les graphes sans $K_{1,3}$ [90].

En accord avec la preuve de l'implication (2) ⇒ (3) du théorème 4.1, nous avons aussi la proposition suivante.

Proposition 4.4 *Le problème de la coloration équitable peut être résolu en temps et espace $O(qn)$ pour les graphes sans $K_{1,3}$, étant donnée en entrée une q-coloration du graphe.*

Remarque. Comme l'ont noté Hansen *et al.* [83], toute q-coloration équitable avec $q = \max\{\chi(G), \lceil n/k \rceil\}$ induit aussi une solution optimale au problème Ordo. Toutefois, le calcul de cette coloration prend $O(\chi(G,k)n)$ temps et espace, et non plus $O(\chi(G)n)$ temps et espace.

Voici maintenant deux corollaires concernant la propriété de redécoupage. Le premier est une conséquence directe de la discussion précédente et le second, qui porte sur les graphes d'arcs propres (ou d'intervalles propres), découle de la preuve de validité de l'algorithme Ordo-Arcs-Propres.

Corollaire 4.4 (Complexité du redécoupage) *Le problème* Ordo *peut être résolu en temps et espace $O(n^2/k)$ pour les graphes sans $K_{1,3}$, étant donnée en entrée une coloration du graphe où chaque couleur apparaît au moins k fois.*

Corollaire 4.5 (Complexité du redécoupage) *Le problème* Ordo *peut être résolu en temps et espace $O(n)$ pour les graphes d'arcs propres, étant données en entrée une coloration du graphe où chaque couleur apparaît au moins k fois et une représentation par arcs propres ordonnée.*

Le problème de la coloration minimum pouvant être résolu en temps $O(n^4)$ pour les graphes parfaits sans $K_{1,3}$ [91], nous avons aussi le corollaire suivant.

Corollaire 4.6 (Graphes parfaits sans $\mathrm{K}_{1,3}$) *Le problème* Ordo *peut être résolu en temps $O(n^4)$ et espace $O(n^2)$ pour les graphes parfaits sans $K_{1,3}$.*

Nous concluons cette discussion par deux corollaires concernant le problème Ordo pour les graphes duaux, qui forment probablement la sous-classe la plus connue des graphes sans $K_{1,3}$. Ce problème peut aussi être vu comme le *problème de la coloration minimum des arêtes d'un graphe tel que chaque couleur n'apparaisse pas plus de k fois*. Rappelons que les problèmes de coloration d'arêtes ont des applications importantes dans la construction d'emplois du temps ainsi qu'en ordonnancement [41, 42, 107, 43, 45, 44]. Le dual d'un graphe G est noté $L(G)$, tandis que n, m et $\Delta(G)$ dénotent respectivement le nombre de sommets, le nombre d'arêtes et le degré maximum de G.

Corollaire 4.7 (Graphes duaux – dichotomie générale) *Soit G un graphe et k un entier positif :*

(1) si $\lceil m/k \rceil \leq \Delta(G)$, alors la double inégalité $\Delta(G) \leq \chi(L(G), k) \leq \Delta(G) + 1$ est satisfaite et le problème Ordo *est $\mathcal{NP}$-difficile pour le graphe $L(G)$;*

(2) si $\lceil m/k \rceil \geq \Delta(G) + 1$, alors l'égalité $\chi(L(G), k) = \lceil m/k \rceil$ est satisfaite et le problème Ordo *peut être résolu en temps et espace polynomial pour le graphe $L(G)$.*

Preuve. La transposition de l'assertion (3) du théorème 4.1 au dual de G donne $\chi(L(G), k) = \max\{\chi(L(G)), \lceil m/k \rceil\}$, tandis que d'un autre côté, le célèbre théorème de Vizing (cf. [46, p. 103–105]) nous fournit la double inégalité $\Delta(G) \leq \chi(L(G)) \leq \Delta(G) + 1$. Ainsi, lorsque $\lceil m/k \rceil \geq \Delta(G) + 1$, nous avons $\chi(L(G), k) = \lceil m/k \rceil$ et la preuve constructive du théorème de Vizing couplée à la proposition 4.3 fournit un algorithme en temps et espace polynomial pour déterminer une partition en $\lceil m/k \rceil$ stables de taille au plus k. Dans le cas contraire, nous avons $\chi(L(G), k) = \chi(L(G))$ (qui implique aussitôt $\Delta(G) \leq \chi(L(G), k) \leq \Delta(G) + 1$) et le problème devient $\mathcal{NP}$-difficile d'après le théorème de Holyer [90]. □

Remarque. Lorsque k est un paramètre fixé, Alon [1] a montré que le problème Ordo est polynomial pour les graphes duaux. D'un autre côté, Cohen et Tarsi [26] ont montré que le problème est $\mathcal{NP}$-difficile pour les compléments des graphes duaux, même lorsque k est un paramètre fixé supérieur ou égal à trois.

Corollaire 4.8 (Graphes duaux – cas polynomiaux) *Si un graphe G ne contient aucun cycle induit de longueur supérieure ou égale à cinq, alors le problème* Ordo *peut être résolu en temps et espace polynomial pour le graphe $L(G)$. De plus, l'égalité $\chi(L(G), k) = \max\{\Delta(G), \lceil m/k \rceil\}$ est satisfaite.*

Preuve. Trotter [150] a montré que le dual $L(G)$ d'un graphe G est parfait si et seulement si G ne contient pas de cycle induit impair de longueur supérieure ou égale à cinq (voir aussi [40, 116]). Puisque le problème de la coloration peut être résolu en temps et espace polynomial pour les graphes parfaits [79], nous obtenons le résultat *via* la proposition 4.3 (lorsque $L(G)$ est parfait, nous avons aussi $\Delta(G) = \chi(L(G))$). □

Corollaire 4.9 (Graphes duaux – cas polynomiaux) *Le problème* Ordo *peut être résolu en temps et espace polynomial pour les duaux de graphes faiblement triangulés.*

Remarque. Les graphes bipartis sont des exemples de graphes qui satisfont les conditions des corollaires 4.8 et 4.9. Notons que pour ces derniers, le résultat découle aussi d'un célèbre théorème de König et de sa preuve constructive (cf. [46, p. 103] ou [137, p. 321–336] pour plus de détails).

4.2 Suffisance pour les graphes d'intervalles ou d'arcs

Dans cette section, nous démontrons que les graphes d'intervalles partagent la propriété de redécoupage, quel que soit k, et un algorithme linéaire est même donné pour résoudre le problème Ordo, étant donnée en entrée une coloration où chaque couleur apparaît au moins k fois. Nous verrons ensuite comment étendre le résultat aux graphes d'arcs. Nous rappelons que d'un autre côté, le problème Ordo est $\mathcal{NP}$-difficile pour ces deux classes de graphes, même avec $k \geq 4$ fixé [14]. Nous terminerons la section en discutant de l'extension de la propriété aux graphes de tolérances propres ou unitaires.

Le cas des graphes d'intervalles

Tout comme les graphes sans $K_{1,3}$, les graphes d'intervalles ne possèdent pas le graphe $K_{k+1,k+1}$ comme sous-graphe induit pour $k \geq 2$. En effet, les graphes d'intervalles sont triangulés et par conséquent ne peuvent induire aucun cycle de longueur égale à quatre sans corde. En nous appuyant sur le lemme suivant, nous allons montrer que les graphes d'intervalles partagent eux aussi la propriété de redécoupage.

Lemme 4.2 (Lemme de découpage) *Soit r stables disjoints $S_1, \ldots, S_r$ contenant chacun au moins t intervalles ($1 \leq r \leq t$) et r entiers positifs $\beta_1, \ldots, \beta_r$ tel que $\sum_{u=1}^{r} \beta_u = t$. Alors il existe un stable S^* de taille t tel que pour tout $u = 1, \ldots, r$, exactement β_u intervalles de S^* appartiennent à S_u. De plus, ce stable S^* peut être extrait en temps $O(rt)$ si les intervalles de chaque stable sont ordonnés.*

Preuve. La preuve est constructive : nous donnons un algorithme pour déterminer le stable S^* en temps $O(rt)$. Celui-ci prend en entrée les r stables disjoints $S_1, \ldots, S_r$, chacun de taille au moins t, ainsi que les r entiers $\beta_1, \ldots, \beta_r$. Nous supposons que les intervalles (ouverts) de chaque stable sont ordonnés selon les extrémités gauches croissantes (l'intervalle de rang j dans S_u sera noté $I_{u,j}$). L'algorithme procède de la manière suivante : l'intervalle de rang j dans S^* est celui de plus petite extrémité droite parmi les intervalles de rang j dans les stables S_u (où les β_u intervalles n'ont pas encore été sélectionnés).

Algorithme Découper-Intervalles ;
Entrée : les stables $S_1, \ldots, S_r$ de taille au moins t, les entiers $\beta_1, \ldots, \beta_r$;
Sortie : le stable S^* ;
Début ;
  $S^* \leftarrow \emptyset$;
  pour j de 1 à t **faire**
    $u^* \leftarrow 0$;
    pour u de 1 à r **faire**
      si $\beta_u > 0$ et ($u^* = 0$ ou $d(I_{u,j}) < d(I_{u^*,j})$) **alors** $u^* \leftarrow u$;
    $S^* \leftarrow S^* \cup \{I_{u^*,j}\}$, $\beta_{u^*} \leftarrow \beta_{u^*} - 1$;
  retourner S^* ;
Fin ;

La figure 4.2 illustre une exécution de l'algorithme sur trois stables S_1, S_2, S_3 de taille trois ($r = 3$, $t = 3$ et $\beta_1 = \beta_2 = \beta_3 = 1$). Les intervalles foncés correspondent aux intervalles inclus dans S^* à chaque rang $j = 1, 2, 3$ (les intervalles hachurés n'étant plus candidats à la sélection à un rang donné).

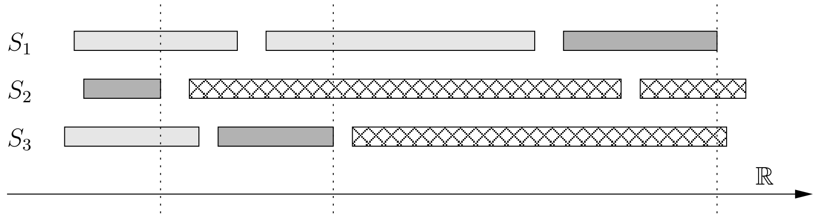


FIG. 4.2 – L'algorithme DÉCOUPER-INTERVALLES.

Établissons la validité de l'algorithme. L'ensemble S^* retourné par l'algorithme contient bien t intervalles (un intervalle est extrait à chaque rang $j = 1, \ldots, t$ et les stables en entrée sont tous de taille au moins t). D'autre part, nous montrons que l'intervalle de rang j dans S^* ne peut pas chevaucher celui de rang $j+1$ pour tout $j = 1, \ldots, t-1$. Pour cela, considérons que ces deux intervalles correspondent respectivement à $I_{u,j} \in S_u$ et $I_{v,j+1} \in S_v$ ($1 \leq u, v \leq t$). Si $u = v$, alors l'assertion est démontrée ($I_{u,j}$ et $I_{v,j+1}$ appartenant au même stable, ils ne peuvent se chevaucher). Dans le cas contraire, supposons que $I_{u,j}$ et $I_{v,j+1}$ se chevauchent, c'est-à-dire que $g(I_{v,j+1}) < d(I_{u,j})$. Comme $d(I_{v,j}) \leq g(I_{v,j+1})$, nous avons aussitôt que $d(I_{v,j}) < d(I_{u,j})$. Or, $I_{v,j+1}$ ayant été sélectionné au rang $j+1$, l'intervalle $I_{v,j}$ était nécessairement un candidat à la sélection au rang j. Par conséquent, $I_{u,j}$ n'était pas, parmi les intervalles candidats au rang j, celui ayant la plus petite extrémité droite, ce qui est une contradiction. □

Proposition 4.5 (Propriété de redécoupage) *Les graphes d'intervalles partagent la propriété de redécoupage, quel que soit k.*

Preuve. Soit G un graphe d'intervalles et $S_1, \ldots, S_q$ une partition de G en stables de taille au moins k. Admettons que les intervalles de chacun des stables de la partition soient ordonnés et posons $|S_u| = \alpha_u k + \beta_u$ comme étant la taille du stable S_u avec α_u un entier positif non nul et $0 \leq \beta_u \leq k-1$. Pour commencer, de chaque stable S_u sont extraits $\alpha_u - 1$ stables de taille k, plus un si $\beta_u = 0$. Après cet opération de nettoyage, chaque stable S_u ne contient plus que $k + \beta_u < 2k$ intervalles avec $\beta_u > 0$. Ensuite, tant qu'il existe r stables dont la somme des indices β_u est supérieure à k, nous appliquons le lemme 4.2 de découpage pour extraire des stables de taille exactement k. Les conditions du lemme de découpage sont respectées puisque chaque stable contient au moins k intervalles. Enfin, lorsque cela n'est plus possible, il ne reste plus qu'à extraire un stable de taille $n \bmod k$ (si n n'est pas un multiple de k) dans les $r < k$ stables dont la somme des $\beta_u > 0$ doit justement être égale à $n \bmod k$. □

La preuve de la proposition précédente fournit un algorithme efficace pour résoudre le problème ORDO pour les graphes d'intervalles, lorsque l'on dispose d'une coloration du graphe où chaque couleur apparaît au moins k fois. En effet, les intervalles de chacun des stables de la partition $S_1, \ldots, S_q$ peuvent être ordonnés en temps et espace $O(n)$ si l'on possède une représentation par intervalles ordonnée du graphe G. De là, le procédure de nettoyage ne prend qu'un temps linéaire. Enfin, en prenant soin de vider les stables de leurs β_u intervalles les uns après les autres, les exécutions répétées de la procédure DÉCOUPER-INTERVALLES ne prennent au total qu'un temps $O(qk)$, soit un temps $O(n)$ puisque $n > qk$.

Corollaire 4.10 (Complexité du redécoupage) *Le problème* ORDO *peut être résolu en temps et espace $O(n)$ pour les graphes d'intervalles, étant données en entrée une coloration du graphe où chaque couleur apparaît au moins k fois et une représentation par intervalles ordonnée.*

Le cas des graphes d'arcs

Nous allons voir que la proposition 4.5 peut être étendue aux graphes d'arcs circulaires à l'aide d'une version plus faible du lemme de découpage. En effet, il n'est malheureusement pas possible d'étendre en l'état le lemme 4.2 de découpage aux arcs circulaires. L'exemple suivant montre que, même en se restreignant à des arcs unitaires, il existe une infinité de cas pour lesquels le lemme 4.2 de découpage n'est pas valide.

Soit un ensemble $S_1, \ldots, S_r$ de stables disjoints contenant chacun t arcs unitaires ouverts avec $\beta_u = 1$ pour tout $u = 1, \ldots, r$ $(r, t \geq 1)$. Afin de définir la position de ces arcs sur le cercle, divisons celui-ci en t sections $\theta_0, \ldots, \theta_{t-1}$, chacune de longueur ℓ. Nous dirons qu'un arc est de rang j si son extrémité *scm* appartient à la section de cercle θ_j. Les r arcs de rang j sont disposés dans chaque section j, chacun de longueur ℓ et décalés de $\epsilon < \ell/t$ par rapport au précédent de façon à ce que l'arc du stable S_u chevauche d'un côté les arcs de rang $(j+1) \bmod t$ appartenant aux stables d'indice inférieur à u et de l'autre les arcs de rang $(j-1) \bmod t$ appartenant aux stables d'indice supérieur à u dans le sens contraire des aiguilles d'une montre (voir la figure 4.3).

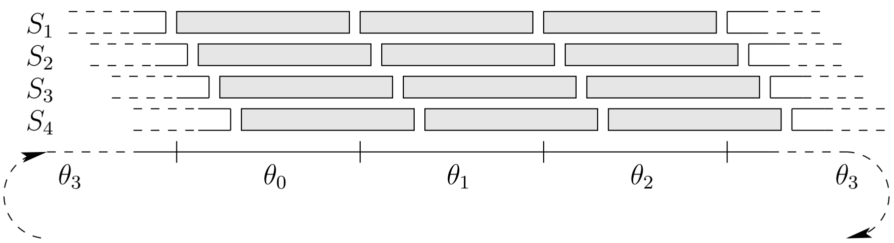


FIG. 4.3 – Une illustration de la construction avec $r = 4$ et $t = 4$.

Maintenant, supposons l'existence d'un stable S^* de taille t possédant un et un seul arc dans chaque stable S_u pour tout $u = 1, \ldots, r$. Aucun de ces arcs ne pouvant être de même rang, S^* ne contient donc qu'un et un seul arc de rang j pour tout $j = 1, \ldots, t$. Admettons que l'arc $A_1^* \in S^*$ provenant du stable S_1 soit de rang j. D'après la remarque précédente, S^* possède nécessairement un arc de rang $(j-1) \bmod t$ appartenant à un des stables $S_2, \ldots, S_r$. Or, par définition, tous les arcs de rang $(j-1) \bmod t$ et différents de A_1^* chevauchent ce même arc A_1^*. Par conséquent, nous aboutissons à une contradiction et il n'existe aucun stable de taille t dans $S_1, \ldots, S_r$ ayant la propriété désirée. En prolongeant la preuve, on peut même montrer qu'il n'existe pas d'autre stable de taille t que les stables $S_1, \ldots, S_r$ eux-mêmes.

Bien que le lemme 4.2 de découpage ne puisse pas s'étendre aux arcs circulaires, nous remarquons que la démonstration de la proposition 4.5 autorise l'emploi d'une version plus faible du lemme en question. En effet, les stables sur lesquels le lemme de découpage est appliqué sont tous de taille *strictement* supérieure à k. Or le lemme 4.2 tient toujours lorsque les stables sont de taille k. Peut-on utiliser cette remarque afin d'étendre la proposition 4.5 aux graphes d'arcs ? La réponse est oui, en utilisant la version suivante du lemme de découpage où chaque stable n'est plus de taille t, mais de taille $t+1$.

Lemme 4.3 (Lemme faible de découpage) *Soit r stables disjoints $S_1, \ldots, S_r$ contenant chacun au moins $t+1$ arcs circulaires ($1 \leq r \leq t$), ainsi que r entiers positifs $\beta_1, \ldots, \beta_r$ tel que $\sum_{u=1}^{r} \beta_u = t$. Alors il existe un stable S^* de taille t tel que pour tout $u = 1, \ldots, r$, exactement β_u arcs de S^* appartiennent à S_u. De plus, ce stable S^* peut être extrait en temps $O(rt)$ si les arcs de chaque stable sont ordonnés.*

Preuve. La démonstration repose sur le lemme 4.2 de découpage. Soit p un point du cercle. Pour tout $u = 1, \ldots, r$, retirons de chaque stable S_u l'arc qui contient p (il en existe au plus un). Suite au retrait de ces arcs, le point p n'est couvert par aucun arc de $S_1, \ldots, S_r$. Par conséquent, le graphe induit par ces stables est maintenant un graphe d'intervalles, sur lequel le lemme de découpage peut s'appliquer (puisque chaque stable est de taille au moins t). Clairement, le retrait des arcs ne demande qu'un temps $O(rt)$. Ensuite, la procédure Découper-Intervalles peut directement être utilisée sur le nouvel ensemble d'arcs, à condition que les arcs de chaque stable soient renumérotés à partir du point p et dans l'ordre circulaire. Le temps total nécessaire à l'extraction du stable S^* reste donc en $O(rt)$. $\square$

En accord avec la discussion précédente, la proposition suivante et son corollaire sont établies.

Proposition 4.6 (Propriété de redécoupage) *Les graphes d'arcs partagent la propriété de redécoupage, quel que soit k.*

Corollaire 4.11 (Complexité du redécoupage) *Le problème* Ordo *peut être résolu en temps et espace $O(n)$ pour les graphes d'arcs, étant données en entrée une coloration du graphe où chaque couleur apparaît au moins k fois et une représentation par arcs ordonnée.*

Remarque. Un corollaire du lemme 4.3 est que les graphes d'arcs ne contiennent pas le graphe $K_{k+1,k+1}$ comme sous-graphe induit pour $k \geq 2$. À la différence des graphes d'intervalles, ils admettent comme sous-graphe induit le graphe $K_{2,2}$, c'est-à-dire le cycle C_4 (tout comme les graphes sans $K_{1,3}$).

Le cas des graphes de tolérances propres

Comme nous l'avons évoqué en introduisant ce chapitre, le graphe $K_{k+1,k+1}$ appartient à la classe des graphes de tolérances bornées, quel que soit k (voir la figure 4.4). Toutefois, une question reste en suspens : le graphe $K_{k+1,k+1}$ possède-t-il une représentation par tolérances propres ? Dans cette dernière partie, nous apportons une réponse à cette question pour $k = 2$, en démontrant que les graphes de tolérances propres ne possèdent aucune copie induite du graphe $K_{3,3}$. D'un autre côté, nous montrerons que le lemme de découpage ne peut pas s'étendre aux graphes de tolérances unitaires pour $k \geq 3$, même dans sa version faible. Nous rappelons que la tolérance d'un intervalle (ou sommet) i est notée $t(i)$.

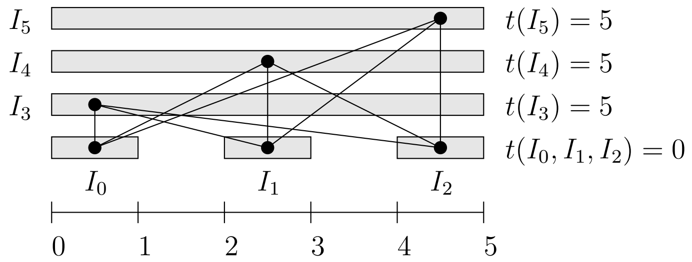


Fig. 4.4 – Une représentation du graphe $K_{3,3}$ par tolérances bornées.

Lemme 4.4 *Soit G un graphe de tolérances propres et A, B deux stables disjoints de G contenant chacun trois sommets. Alors il existe un sommet dans A et un sommet dans B qui ne sont reliés par aucune arête.*

Preuve. Considérons une représentation par tolérances propres du graphe G où tous les intervalles sont ouverts et notons $A = \{a_1, a_2, a_3\}$ et $B = \{b_1, b_2, b_3\}$ avec (i) $g(a_1) \leq g(a_2) \leq g(a_3)$ et $g(b_1) \leq g(b_2) \leq g(b_3)$ (comme les intervalles sont propres, les extrémités droites satisfont les mêmes inégalités). Sans perdre de généralité, nous pouvons admettre que $g(a_2) \leq g(b_2)$. Nous affirmons alors que les sommets a_1 et b_3 ne sont reliés par aucune arête dans le graphe G. Pour cela, montrons que $d(a_1) - g(b_3) \leq \min\{t(a_1), t(b_3)\}$. Les couples de sommets a_1, a_2 et b_2, b_3 appartenant chacun à un même stable, nous avons que (ii) $d(a_1) - g(a_2) \leq \min\{t(a_1), t(a_2)\}$ et $d(b_2) - g(b_3) \leq \min\{t(b_2), t(b_3)\}$. Comme les intervalles sont propres et que $g(a_2) \leq g(b_2)$, nous avons aussi que (iii) $d(a_1) - g(a_2) \geq d(a_1) - g(b_2)$ et $d(b_2) - g(b_3) \geq d(a_2) - g(b_3)$. En combinant les inégalités (i), (ii) et (iii), nous obtenons alors que

$$d(a_1) - g(b_3) \leq d(a_1) - g(b_2) \leq d(a_1) - g(a_2) \leq \min\{t(a_1), t(a_2)\} \leq t(a_1)$$

$$d(a_1) - g(b_3) \leq d(a_2) - g(b_3) \leq d(b_2) - g(b_3) \leq \min\{t(b_2), t(b_3)\} \leq t(b_3)$$

Ainsi, l'inégalité $d(a_1) - g(b_3) \leq \min\{t(a_1), t(b_3)\}$ est valide et notre affirmation justifiée. □

Proposition 4.7 (Propriété de redécoupage) *Les graphes de tolérances propres partagent la propriété de redécoupage lorsque $k = 2$.*

Corollaire 4.12 (Complexité du redécoupage) *Le problème 2-Ordo peut être résolu en temps et espace linéaire pour les graphes de tolérances propres, étant donnée en entrée une coloration du graphe où chaque couleur apparaît au moins deux fois.*

Remarque. Le problème peut même être résolu en temps et espace $O(n)$ si une représentation par tolérances propres est fournie en entrée. Notons que ce type de résultat peut s'avérer utile à la conception d'un algorithme dédié à la résolution du problème 2-Ordo pour les graphes de tolérances propres, comme cela a été fait pour les graphes d'intervalles (voir l'algorithme 2-Ordo-Intervalles, p. 27).

L'extension de la proposition précédente pour $k \geq 3$ semble être un problème difficile. En effet, nous allons voir au travers de l'exemple suivant que toute tentative d'extension du lemme de découpage aux graphes de tolérances unitaires et bornées est vouée à l'échec (y compris dans sa version faible).

Soit M_k le graphe défini comme l'union de deux ensembles de sommets numérotés de 1 à k et dont les seuls couples de sommets non connectés par une arête sont ceux qui portent des numéros identiques. Définissons maintenant le graphe N_k, pour $k \geq 3$, comme l'union de deux graphes M_k et d'un stable S de taille $k^2 - 3k$ (voir la figure 4.5). Le graphe N_k possède une partition $S_1, \ldots, S_k$ en k stables de taille $k+1$, où chaque stable S_k est composé des quatre sommets portant le numéro k dans $M_k \cup M_k$ et de $k-3$ sommets de S. Nous affirmons que de ce graphe M_k ne peut être extrait aucun stable S^* de taille k et possédant un sommet dans chaque stable S_u pour tout $u = 1, \ldots, k$. En effet, si un sommet i de S^* appartient au sous-graphe M_k, alors aucun autre sommet de M_k ne peut appartenir à S^* (puisque le seul sommet avec lequel i n'est pas connecté dans M_k appartient au même stable que i). Par conséquent, le stable S^* doit contenir au moins $k-2$ sommets provenant du sous-graphe $N_k \setminus (M_k \cup M_k)$ et appartenant chacun à des stables différents, ce qui est impossible puisque chaque stable S_u ($1 \leq u \leq k$) ne contient pas plus de $k-3$ sommets dans ce sous-graphe.

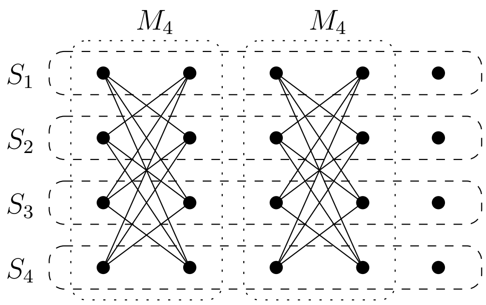


FIG. 4.5 – Le graphe N_4.

À présent, voici une représentation de M_k par tolérances unitaires et bornées. Soit p un point entier sur l'axe des réels et ℓ un entier supérieur à $4k$. Pour tout $i = 1, \ldots, k$, les deux sommets portant le numéro i sont respectivement représentés par les intervalles $I_i^g =]p+i-\ell, p+i[$ et $I_i^d =]p-i, p-i+\ell[$ munis des tolérances $t(I_i^g) = t(I_i^d) = 2i$ (voir la figure 4.6). L'ensemble composé des intervalles I_i^g, respectivement I_i^d, induit une clique, car ceux-ci partagent une portion de l'axe de longueur supérieure à $2k$. Ensuite, pour toute paire d'intervalles I_i^g et $I_{i'}^d$, nous avons $|I_i^g \cap I_{i'}^d| = i + i'$; leurs tolérances respectives étant de $2i$ et $2i'$, ceux-ci ne sont donc reliés par une arête que si $i + i' \leq 2i$ et $i + i' \leq 2i'$, soit $i = i'$. Par conséquent, cette représentation est correcte. De plus, une représentation du graphe N_k par tolérances unitaires et bornées se déduit aisément de celle-ci. Ainsi, la version faible du lemme de découpage tient pour les graphes de tolérances propres lorsque $t = 2$ (d'après le lemme 4.4), mais pas pour les graphes de tolérances unitaires et bornées lorsque $t \geq 3$. Puisque le graphe M_2 est isomorphe au cycle C_4, nous pouvons remarquer que la version forte du lemme ne tient pas non plus lorsque $t = 2$.

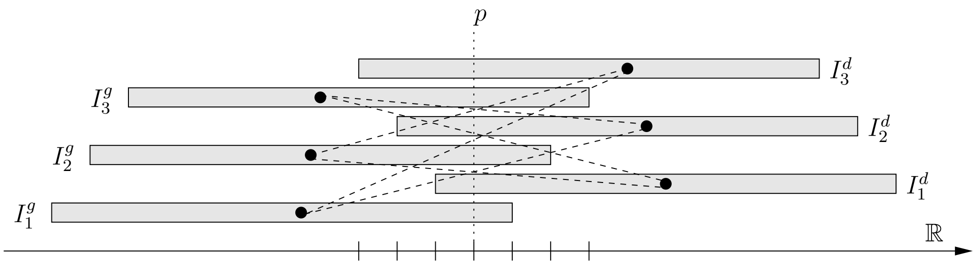


FIG. 4.6 – Une représentation du graphe M_3 par tolérances unitaires.

Remarque. La version faible du lemme de découpage tient pour les graphes planaires lorsque $t = 2$, puisqu'un graphe planaire ne peut contenir le graphe $K_{3,3}$ comme sous-graphe induit (d'après le théorème de Kuratowski, cf. [46, p. 80–84]). Par conséquent, les graphes planaires partagent la propriété de redécoupage lorsque $k = 2$. D'un autre côté, le graphe N_3 admet une représentation planaire, ce qui condamne toute tentative d'extension pour $t = 3$. Comme pour les graphes de tolérances propres, la question suivante se pose alors : les graphes planaires partagent-ils la propriété de redécoupage lorsque $k \geq 3$?

4.3 Suffisance pour les graphes triangulés

Dans cette dernière section, nous discutons de la propriété de redécoupage chez les graphes triangulés. En dépit de nombreux efforts, nous ne sommes parvenus à étendre la proposition 4.5 que pour $k \leq 4$.

Proposition 4.8 (Propriété de redécoupage) *Les graphes triangulés partagent la propriété de redécoupage lorsque $k \leq 4$.*

Comme pour les graphes d'intervalles ou d'arcs, la preuve de cette proposition repose sur un lemme de découpage. Mais avant de détailler ce lemme, voici la propriété cruciale sur laquelle repose toutes les preuves de cette section.

Lemme 4.5 (Propriété locale de faible densité) *Soit A, B deux stables disjoints dans un graphe triangulé. Alors le sous-graphe induit par A et B est une forêt contenant au plus $|A| + |B| - 1$ arêtes.*

Preuve. Le sous-graphe induit par A et B est triangulé, donc il ne contient pas de cycle induit de longueur supérieure ou égale à quatre. Étant biparti, il ne contient pas non plus de cycle induit de longueur trois. Par conséquent, celui-ci est acyclique et ne contient donc pas plus de $|A| + |B| - 1$ arêtes. □

Le lemme de découpage, ici dans sa version faible et *ad hoc* pour $k \leq 4$, est établi en deux parties. La première partie, décrite ci-dessous, permet d'extraire des stables de taille deux ou trois lors de l'application de la méthode de découpage utilisée pour la preuve des proposition 4.5 et 4.6, pour $k \leq 4$. Nous dirons qu'un sommet est d-connecté (resp. exactement d-connecté) à un stable lorsqu'il est connecté à au moins (resp. exactement) d sommets de ce stable.

Lemme 4.6 (Lemme *ad hoc* de découpage) *Soit A, B, C trois stables disjoints dans un graphe triangulé :*

(1) si A et B sont chacun de taille deux, alors il existe un stable de taille deux avec un sommet dans A et un sommet dans B ;

(2) si A et B sont chacun de taille trois, alors il existe un stable de taille trois avec deux sommets dans A et un sommet dans B ;

(3) si A, B et C sont chacun de taille quatre, alors il existe un stable de taille trois avec un sommet dans chacun des stables A, B et C.

Preuve. La preuve de l'assertion (1) est immédiate d'après la propriété locale de faible densité. En effet, prenons deux sommets de A et deux sommets de B. Si (1) n'est pas vérifiée, alors les deux sommets de A doivent être connectés aux deux sommets de B. Cela fait quatre arêtes au total, alors que la propriété locale de faible densité en impose moins de trois. Plus directement, l'assertion (1) découle du fait que le graphe biparti induit par A et B ne contient pas de copie induite du graphe $K_{2,2}$, *alias* le cycle C_4.

La preuve de l'assertion (2) est tout aussi directe. Si (2) n'est pas vérifiée, les trois sommets de B sont connectés à au moins deux des trois sommets de A. Le sous-graphe induit par ces six sommets doit donc contenir six arêtes. Or, la propriété locale de faible densité en impose moins de cinq, ce qui est une contradiction.

Supposons maintenant qu'il n'existe pas de stable comme décrit dans l'assertion (3). Tout d'abord, nous affirmons que tout sommet $i \in A \cup B \cup C$ est 3-connecté à au moins un stable parmi A, B, C. Sans perdre de généralité, admettons que $i \in A$ contredise cette affirmation. Il existe donc au moins deux sommets de B et deux sommets de C qui ne sont pas connectés à i et d'après la propriété locale de faible densité, un des deux sommets de B n'est pas connecté à un des deux sommets de C. Comme ces derniers ne sont pas connectés au sommet i, ils induisent avec celui-ci un stable de taille trois ayant la propriété désirée, ce qui est une contradiction. Notre première affirmation est donc démontrée. Prenons maintenant trois sommets de A. D'après l'affirmation précédente, ces trois sommets doivent chacun être 3-connectés aux stables B ou C. Par conséquent, au moins deux de ces trois sommets sont 3-connectés au même stable, ce qui est impossible sans violer la propriété locale de faible densité. □

Remarque. Nous verrons par la suite que l'assertion (3) peut être renforcée, au prix de quelques efforts supplémentaires (voir le lemme 4.12, p. 87).

La seconde partie du lemme de découpage, décrite ci-après, statue sur l'extraction des stables de taille quatre. Hormis la preuve de l'assertion (4), la démonstration de cette seconde partie demande plus efforts.

Lemme 4.7 (Lemme *ad hoc* de découpage) *Soit A, B, C, D quatre stables disjoints dans un graphe triangulé :*

(4) si A et B sont chacun de taille quatre, alors il existe un stable de taille quatre avec trois sommets dans A et un sommet dans B ;

(5) si A et B sont chacun de taille quatre, alors il existe un stable de taille quatre avec deux sommets dans A et deux sommets dans B ;

(6) si A, B et C sont respectivement de taille six, quatre et quatre, alors il existe un stable de taille quatre avec deux sommets dans A, un sommet dans B et un sommet dans C ;

(7) si A, B, C et D sont chacun de taille cinq, alors il existe un stable de taille quatre avec un sommet dans chacun des stables A, B, C et D.

L'assertion (4) s'obtient directement par application de la propriété locale de faible densité. En effet, si le stable de taille quatre que nous recherchons dans A et B n'existe pas, alors chaque sommet de B doit être connecté à au moins deux sommets de A. Ainsi, le sous-graphe induit par A et B doit contenir au moins huit arêtes, alors que la propriété locale de faible densité en impose moins de sept.

Pour faciliter l'écriture des preuves qui vont suivre, nous aurons besoin de quelques définitions spécifiques. Soit A, B, C, D quatre stables disjoints dans un graphe triangulé, chacun de taille au moins deux. Nous appelons *doublet* dans (A, B) un stable de taille deux avec un sommet dans A et un sommet dans B ; par analogie, un *triplet* dans (A, B, C) (resp. un *quadruplet* dans (A, B, C, D)) est est un stable de taille trois (resp. quatre) avec un sommet dans chacun des stables A, B et C (resp. A, B, C et D). Un *carré* dans (A, B) est un stable de taille quatre avec deux sommets dans A et deux sommets dans B ; par analogie, un *quasi-carré* $\{a, b, a', b'\}$ dans (A, B) correspond à la succession des trois doublets $\{a, b\}$, $\{b, a'\}$ et $\{a', b'\}$ et devient un carré si les sommets a et b' ne sont pas reliés (voir la figure 4.7). Nous rappelons que le degré d'un sommet i est noté $d(i)$.

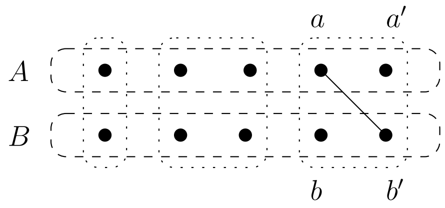


FIG. 4.7 – Un doublet, un carré et un quasi-carré.

Lemme 4.8 (Lemme des doublets) *Soit A, B deux stables disjoints dans un graphe triangulé. Si A et B sont de taille au moins $t \geq 2$, alors il existe $t-1$ doublets disjoints dans (A, B).*

Preuve. Lorsque A et B sont de taille $t = 2$, l'assertion est immédiatement vérifiée (d'après la propriété locale de faible densité). Pour $t > 2$, nous procédons de la façon suivante. Tant que $t \geq 2$, nous appliquons le lemme avec $t = 2$ sur deux stables $A' \subseteq A$ et $B' \subseteq B$ afin d'en retirer un doublet. Clairement, le nombre de doublets ainsi extraits sera de $t - 1$. □

Preuve des assertions (5) et (6) du lemme de découpage

Voici le lemme sur lequel repose la preuve des assertions (5) et (6). Nous verrons qu'il permet aussi d'obtenir simplement l'assertion (3) du lemme 4.6.

Lemme 4.9 (Lemme du carré) *Soit A, B deux stables disjoints dans un graphe triangulé. Si A et B sont chacun de taille au moins quatre, alors il existe un carré dans (A, B).*

Preuve. Nous écrirons $A = \{a_1, a_2, a_3, a_4\}$ et $B = \{b_1, b_2, b_3, b_4\}$. Supposons qu'il n'existe aucun carré dans (A, B). Cela implique immédiatement que pour toute paire de sommets $a_i, a_j \in A$, l'inégalité $d(a_i) + d(a_j) \geq 3$ est satisfaite (dans le cas contraire, au moins deux sommets de B ne sont pas connectés à a_i et a_j, et un carré existe dans (A, B)). Nous pouvons en déduire que trois sommets de A ont un degré au moins deux. D'un autre côté, la propriété locale de faible densité impose que la somme des degrés des sommets appartenant à A soit inférieure ou égale à sept. Par conséquent, nous pouvons sans perdre de généralité poser $d(a_1) \leq 1$, $d(a_2) \geq 2$, $d(a_3) \geq 2$ et $d(a_4) \geq 2$. En appliquant la propriété locale de faible densité au sous-graphe induit par $A \setminus \{a_1\}$ et B, nous obtenons aussi l'inégalité $d(a_2) + d(a_3) + d(a_4) \leq 6$ qui, combinée aux précédentes, nous donne (i) $d(a_1) \leq 1$ et $d(a_2) = d(a_3) = d(a_4) = 2$.

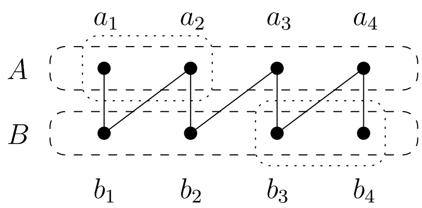


FIG. 4.8 – La chaîne et le carré.

Par des arguments symétriques, nous obtenons que (ii) $d(b_1) = d(b_2) = d(b_3) = 2$ et $d(b_4) \leq 1$ pour les sommets de l'ensemble B. Une fois réunies, les conditions (i) et (ii) forcent le graphe biparti induit par A et B à être isomorphe à une chaîne où $\{a_1, a_2, b_3, b_4\}$ forme un carré (voir la figure 4.8), ce qui contredit notre hypothèse première. □

L'assertion (5) du lemme 4.7 découle immédiatement du lemme du carré. Voici maintenant la preuve de l'assertion (6). Soit A, B, C trois stables disjoints dans un graphe triangulé, respectivement de taille six, quatre et quatre. D'après le lemme du carré, il existe un carré $\{b_1, c_1, b_2, c_2\}$ dans (B, C). Considérons alors le graphe biparti induit par ce carré et le stable A et supposons qu'il n'existe aucun stable de taille quatre avec un sommet dans $\{b_1, b_2\}$, un sommet dans $\{c_1, c_2\}$ et deux sommets dans A. Clairement, cela implique que pour toute paire b_i, c_j de sommets $(1 \leq i, j \leq 2)$, l'inégalité $d(b_i) + d(c_j) \geq 5$ soit satisfaite. En sommant les quatre inégalités en question, nous obtenons que le nombre d'arêtes dans le graphe biparti considéré est supérieur à dix. Or, la propriété locale de faible densité en impose un nombre inférieur ou égal à neuf, ce qui est une contradiction.

Remarque. Une preuve de l'assertion (3) du lemme 4.6 s'obtient par des arguments similaires. Cette technique de preuve peut d'ailleurs être généralisée en le lemme suivant.

Lemme 4.10 *Soit $X = X_1 \cup \cdots \cup X_t$ et Y deux stables disjoints de taille $2t$ dans un graphe triangulé, où $X_1, \ldots, X_t$ forment t couples disjoints de sommets ($t \geq 1$). Alors il existe un stable de taille $t+1$ composé d'un sommet provenant de chaque ensemble X_i ($1 \leq i \leq t$) et d'un sommet de Y.*

Preuve. Supposons qu'un tel stable n'existe pas. L'inégalité $d(x_1) + \cdots + d(x_t) \geq 2t$ est alors satisfaite pour tout t-uplet $x_1 \in X_1, \ldots, x_t \in X_t$. En sommant ces inégalités sur tous les t-uplets possibles, nous obtenons que $d(X_1) + \cdots + d(X_t) \geq 4t$ où $d(X_i)$ représente la somme des degrés des sommets appartenant à l'ensemble X_i. D'un autre côté, la propriété locale de faible densité impose que $d(X_1) + \cdots + d(X_t) \leq 4t - 1$, ce qui contredit l'inégalité précédente. □

La preuve de l'assertion (3) correspond alors à la combinaison du lemme du carré et du lemme 4.10 où $t = 2$: il suffit de poser $X_1 = \{b_1, b_2\}$, $X_2 = \{c_1, c_2\}$ et $Y = A$ avec $\{b_1, b_2, c_1, c_2\}$ un carré dans (B, C).

Remarque. Cette seconde remarque a été émise par F. Maffray : l'assertion (5) peut être renforcée, il suffit en effet que le stable A soit de taille cinq, les stables B et C restant de taille quatre. En utilisant les mêmes arguments, nous obtenons que $A \cup \{b_1, c_1, b_2, c_2\}$ induit un arbre. Par recherche exhaustive en raisonnant sur le diamètre (ou sur le degré maximum), nous ne trouvons alors que quelques arbres de ce type, et tous ont la propriété voulue.

Preuve de l'assertion (7) du lemme de découpage

La preuve de l'assertion (7) reste dans le même esprit que celle que nous avons donnée pour l'assertion (3) du lemme 4.6. Celle-ci repose essentiellement sur le fait qu'un triplet peut être extrait de trois stables disjoints de taille seulement trois, affirmation qui repose essentiellement sur le lemme suivant.

Lemme 4.11 (Lemme du quasi-carré) *Soit A, B deux stables disjoints dans un graphe triangulé. Si A et B sont chacun de taille au moins trois, alors il existe un quasi-carré dans (A, B).*

Preuve. Posons $A = \{a_1, a_2, a_3\}$ et $B = \{b_1, b_2, b_3\}$ et distinguons les deux cas suivants.

Cas (a) : le degré maximum dans le sous-graphe induit par A et B est au plus deux. Immédiatement, nous en déduisons que ce sous-graphe est une union de chaînes disjointes. Dans le pire des cas et sans perdre de généralité, admettons que ce sous-graphe soit isomorphe à la chaîne $\{a_1, b_1, a_2, b_2, a_3, b_3\}$. Nous observons alors que l'ensemble $\{a_1, a_2, b_2, b_3\}$ induit un quasi-carré (voir la figure 4.9).

Cas (b) : le degré maximum dans le sous-graphe induit par A et B est trois. Sans perdre de généralité, admettons que le sommet a_1 soit de degré trois. D'après la propriété locale de faible densité, nous obtenons immédiatement que les sommets a_2 et a_3 sont tous deux de degré inférieur à un. Après élimination des isomorphismes, les deux seules configurations possibles sont illustrées sur la figure 4.10. Après avoir numéroté les sommets de ces deux configurations comme sur celle-ci, nous remarquons que l'ensemble $\{a_2, a_3, b_1, b_2\}$ forme un quasi-carré.

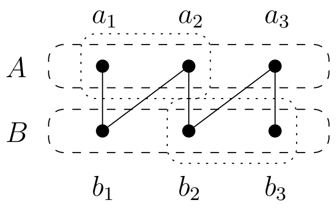


FIG. 4.9 – Lemme du quasi-carré : le cas (a).

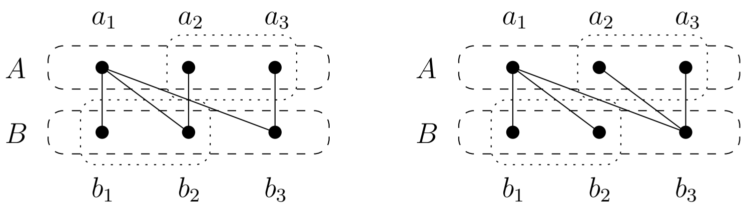


FIG. 4.10 – Lemme du quasi-carré : les deux configurations du cas (b).

□

Lemme 4.12 (Lemme du triplet) *Soit A, B, C trois stables disjoints dans un graphe triangulé. Si A, B et C sont chacun de taille au moins trois, alors il existe un triplet dans (A, B, C).*

Preuve. Posons $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$ et $C = \{c_1, c_2, c_3\}$. D'après le lemme précédent, il existe un quasi-carré dans (A, B). Sans perdre de généralité, notons $\{a_1, b_1, a_2, b_2\}$ ce quasi-carré (avec $\{a_1, b_2\}$ le couple de sommets pouvant être relié par une arête) et supposons qu'il n'existe aucun triplet dans (A, B, C). Nous avons alors nécessairement que (i) $d(a_1) + d(b_1) \geq 3$, $d(b_1) + d(a_2) \geq 3$ et $d(a_2) + d(b_2) \geq 3$. D'un autre côté, la propriété locale de faible densité impose que (ii) $d(a_1) + d(b_1) + d(a_2) \leq 5$ et $d(b_1) + d(a_2) + d(b_2) \leq 5$ (puisque les ensembles $\{a_1, b_1, a_2\}$ et $\{b_1, a_2, b_2\}$ induisent chacun un stable). Des inégalités (i) et (ii) nous pouvons déduire que $1 \leq d(a_i) \leq 2$ et $1 \leq d(b_i) \leq 2$ pour $i = 1, 2$. Après propagation des contraintes et élimination des symétries, les trois seules configurations possibles sont :

	$d(a_1)$	$d(b_1)$	$d(a_2)$	$d(b_2)$
Cas (a)	1	2	1	2
Cas (b)	1	2	2	1
Cas (c)	2	1	2	2

Cas (a) : sans perdre de généralité, admettons que le sommet a_1 soit connecté au sommet c_1. En appliquant successivement les inégalités (i), nous obtenons que les couples de sommets suivants doivent être reliés par une arête : $\{b_1, c_2\}$, $\{b_1, c_3\}$, $\{a_2, c_1\}$, $\{b_2, c_2\}$, $\{b_2, c_3\}$ (voir la figure 4.11). Le sous-graphe induit par $\{b_1, b_2, c_2, c_3\}$ ne respecte alors pas la propriété locale de faible densité, ce qui est une contradiction.

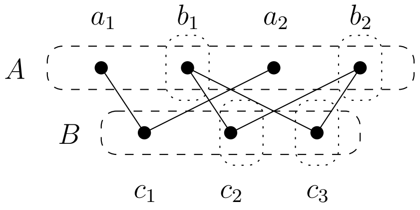


FIG. 4.11 – Lemme du triplet : le cas (a).

Cas (b) : sans perdre de généralité, admettons que le sommet a_1 soit connecté au sommet c_1. En appliquant successivement les inégalités (i) tout en respectant la propriété locale de faible densité, nous obtenons sans perdre de généralité que les couples de sommets suivants doivent être reliés par une arête : $\{b_1, c_2\}$, $\{b_1, c_3\}$, $\{a_2, c_1\}$, $\{a_2, c_2\}$, $\{b_2, c_3\}$ (voir la figure 4.12). Si le sommet a_1 est relié au sommet b_2, alors le sous-graphe induit par $\{a_1, b_1, a_2, b_2\}$ et C est isomorphe à un cycle de longueur sept sans corde, ce qui est une contradiction. Dans le cas contraire, l'ensemble $\{a_1, b_2, c_2\}$ induit le triplé désiré, ce qui contredit notre hypothèse première.

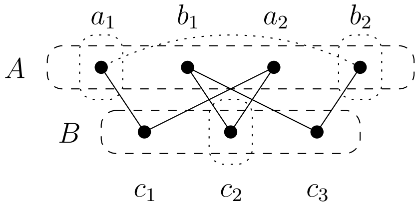


FIG. 4.12 – Lemme du triplet : le cas (b).

Cas (c) : sans perdre de généralité, admettons que le sommet a_1 soit connecté aux sommets c_1 et c_2. En appliquant successivement les inégalités (i), nous obtenons que les couples de sommets suivants doivent être reliés par une arête : $\{b_1, c_3\}$, $\{a_2, c_1\}$, $\{a_2, c_2\}$ (voir la figure 4.13). Le sous-graphe induit par $\{a_1, a_2, c_1, c_2\}$ ne respecte alors pas la propriété locale de faible densité, ce qui est une contradiction.

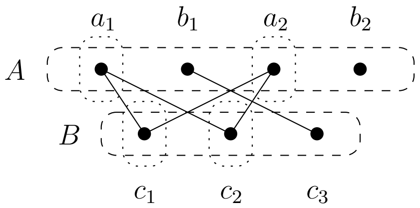


FIG. 4.13 – Lemme du triplet : le cas (c).

$\square$

À présent, l'assertion (7) du lemme de découpage peut être démontrée. Soit A, B, C, D quatre stables disjoints dans un graphe triangulé, chacun de taille cinq. Tout d'abord, nous montrons que s'il n'existe pas de quadruplet dans (A, B, C, D), alors tout sommet du sous-graphe induit par ces quatre stables doit être exactement 3-connecté à un des trois stables auxquels il n'appartient pas.

Montrons que tout sommet doit être 3-connecté à un des trois stables auxquels il n'appartient pas. Sans perdre de généralité, supposons que $a_1 \in A$ ne soit 3-connecté à aucun des trois stables B, C ou D. Dans ce cas, il existe un sous-graphe composé de trois sommets de B, trois sommets de C et trois sommets de D tel que chacun de ces neuf sommets ne soit pas connecté à a_1. D'après le lemme 4.12, il existe un triplet dans ce sous-graphe et donc un quadruplet dans (A, B, C, D) (puisque chaque sommet de ce triplet n'est pas connecté à a_1), ce qui est une contradiction.

Sans perdre de généralité, supposons maintenant que le sommet $a_1 \in A$ soit 4-connecté au stable B. Clairement, aucun autre sommet de A ne peut être 3-connecté à B sans violer la propriété locale de faible densité. Sans perdre de généralité, admettons que les sommets $a_2, a_3 \in A$ (resp. $a_4, a_5 \in A$) soient 3-connectés à C (resp. D). D'après la discussion précédente, les sommets de D doivent être chacun 3-connectés à A, B ou C. Ainsi, au moins un sommet de D est 3-connecté au stable B. Sans perdre de généralité, admettons que ce soit le sommet $d_1 \in D$. Comme a_1 est 4-connecté à B, il existe deux sommets de B qui sont à la fois connectés à a_1 et à d_1. D'après la propriété locale de faible densité, ces deux derniers sommets doivent donc être reliés par une arête. Par des arguments similaires, nous pouvons montrer qu'un autre sommet d_2 est nécessairement connecté à a_1. En effet, il existe un sommet de D qui est soit 3-connecté à B comme d_1, ou bien 3-connecté à A et relié à a_1 (voir la figure 4.14 en considérant $d_4, d_5 \in D$ comme 3-connectés au stable C : soit deux sommets de D dont d_1 sont 3-connectés à B, soit deux sommets de D différents de d_1 sont 3-connectés à A). En résumé, le sommet $a_1 \in A$ est 2-connecté au stable D. Les sommets $a_4, a_5 \in A$ étant 3-connectés au stable D, nous obtenons alors la contradiction suivante : le sous-graphe induit par $\{a_1, a_4, a_5\}$ et D ne respecte pas la propriété locale de faible densité.

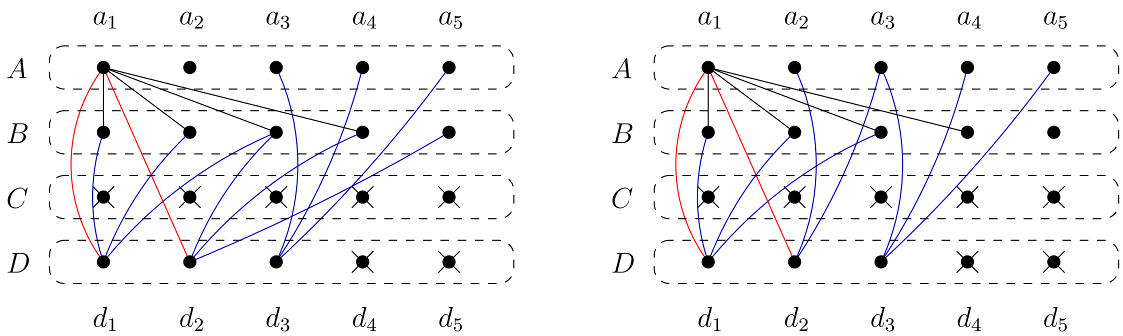


FIG. 4.14 – La preuve de l'assertion (7) : a_1 est 4-connecté à B.

Nous venons de démontrer que tout sommet du sous-graphe induit doit être exactement 3-connecté à un des trois stables auxquels il n'appartient pas. En développant le même type d'arguments, nous pouvons montrer que s'il n'existe pas de quadruplet dans (A, B, C, D), alors tout sommet de ce même sous-graphe ne peut pas être à la fois 3-connecté à un stable et 2-connecté à un autre. Sans perdre de généralité, supposons que le sommet $a_1 \in A$ soit relié aux sommets $b_1, b_2, b_3 \in B$ et relié aux sommets $c_1, c_2 \in C$. Clairement, un second sommet de A doit être 3-connecté au stable B, un autre au stable C et les deux derniers au stable D. Sans perdre de généralité, admettons que $a_2 \in A$ soit 3-connecté à B, que $a_3 \in A$ soit 3-connecté à C et que $a_4, a_5 \in A$ soient 3-connectés à D. D'après la discussion précédente, chaque sommet de D doit être 3-connecté à A, B ou C. Nous affirmons qu'au moins deux sommets de D satisfont une des trois conditions suivantes : (i) le sommet est 3-connecté au stable A et relié à a_1, (ii) le sommet est 3-connecté à B et relié à au moins deux sommets parmi $\{b_1, b_2, b_3\}$, ou bien (iii) le sommet est 3-connecté au stable C et relié aux deux sommets $\{c_1, c_2\}$. Afin d'éviter de trop lourds détails techniques, nous laissons au lecteur le soin de vérifier cette affirmation (voir la figure 4.15).

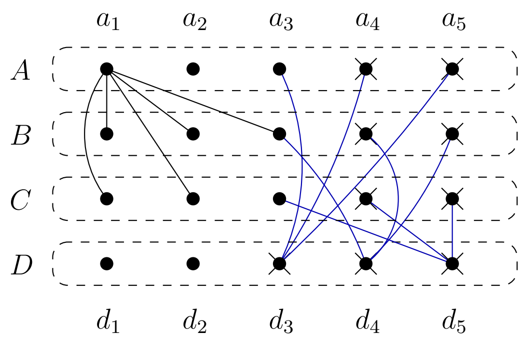


FIG. 4.15 – La preuve de l'assertion (7) : a_1 est 3-connecté à B et 2-connecté à C.

Sans perdre de généralité, notons d_1 et d_2 les deux sommets de D qui satisfont à une de ces trois conditions. De là, nous obtenons que d_1 et d_2 sont tous deux connectés au sommet a_1 (dans le cas (i), cela est immédiat et dans les cas (ii) et (iii), la propriété locale de faible densité l'impose). Les sommets $a_4, a_5 \in A$ étant 3-connectés au stable D, nous obtenons à nouveau la contradiction suivante : le sous-graphe induit par $\{a_1, a_4, a_5\}$ et D ne respecte pas la propriété locale de faible densité.

Pour conclure, nous établissons enfin l'assertion (7) du lemme 4.7 de découpage. Supposons pour cela qu'il n'existe pas de quadruplet dans (A, B, C, D). D'après la discussion précédente, tout sommet du sous-graphe induit par ces quatre stables est exactement 3-connecté à un stable, mais ne peut pas en même temps être 2-connecté à un autre stable. Dans chaque stable A, B, C ou D, deux paires de sommets sont 3-connectées au même stable. Sans perdre de généralité, considérons que les deux sommets $a_1, a_5 \in A$ soient 3-connectés au stable B avec b_1, b_2, b_3 reliés à a_1 et b_3, b_4, b_5 reliés à a_5. Comme a_1 et a_5 ne peuvent pas être 2-connectés à C ou D, au moins trois sommets de C et trois sommets de D ne sont reliés ni à a_1, ni à a_5. Sans perdre de généralité, admettons que ces sommets soient respectivement c_2, c_3, c_4 et d_2, d_3, d_4 (voir la figure 4.16). D'après le lemme du triplet, il existe un triplet avec un sommet dans $\{b_1, b_2, b_4, b_5\}$, un sommet dans $\{c_2, c_3, c_4\}$ et un sommet dans $\{d_2, d_3, d_4\}$. Si le triplet en question comporte un des deux sommets b_1 ou b_2 (resp. b_4 ou b_5), alors il induit aussi un quadruplet avec le sommet a_5 (resp. a_1). Dans un cas comme dans l'autre, (A, B, C, D) contient un quadruplet, ce qui contredit notre hypothèse première et complète ainsi la preuve de l'assertion (7) du lemme de découpage.

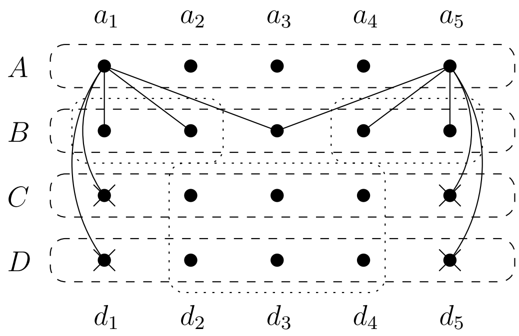


FIG. 4.16 – La preuve de l'assertion (7) : l'épilogue.

Bien qu'existentielles, les preuves des lemmes 4.6 et 4.7 de découpage peuvent servir de base à une méthode de découpage par recherche exhaustive, dont la complexité reste linéaire lorsque $k \leq 4$. Par conséquent, nous avons aussi le corollaire suivant.

Corollaire 4.13 (Complexité du redécoupage) *Le problème* ORDO *peut être résolu en temps et espace linéaire pour les graphes triangulés lorsque $k \leq 4$, étant donnée en entrée une coloration du graphe où chaque couleur apparaît au moins k fois.*

Discussion et conjectures

Une question majeure reste ouverte : *les graphes triangulés possèdent-ils la propriété de redécoupage lorsque $k \geq 5$?* N'ayant trouvé aucun contre-exemple allant à l'encontre de cette proposition, nous émettons la conjecture suivante.

Conjecture. Les graphes triangulés partagent la propriété de redécoupage, quel que soit k.

Nous avons pu vérifier la validité de cette assertion pour deux sous-classes des graphes triangulés : les forêts et les graphes scindés. Le résultat concernant les forêts est dû à Baker et Coffman [6] ; l'algorithme polynomial que proposent les deux auteurs pour résoudre le problème ORDO restreint aux forêts et aux arbres s'appuie sur ce résultat. La preuve que nous proposons ici, plus directe que l'originale, fournit un algorithme simple pour le redécoupage.

Proposition 4.9 (Baker et Coffman, 1996) *Les forêts partagent la propriété de redécoupage, quel que soit k.*

Preuve. Soit $F = (X, Y, E)$ une forêt sous la forme d'un graphe biparti avec $|X| \geq k$ et $|Y| \geq k$. Nous pouvons considérer que $|X| = k + \beta_x$ et $|Y| = k + \beta_y$ avec $1 \leq \beta_x, \beta_y \leq k - 1$ (si tel n'est pas le cas, il suffit d'extraire des stables de taille exactement k dans X et Y afin s'y ramener). Si $\beta_x + \beta_y > k$, alors découper F de façon triviale en quatre stables de taille au plus k est le mieux que l'on puisse faire. Dans le cas contraire, c'est-à-dire $\beta_x + \beta_y \leq k$, nous montrons que l'on peut toujours extraire de F un stable de taille exactement $\beta_x + \beta_y$ et obtenir ainsi une partition en trois stables de taille au plus k.

Lorsque $\beta_x+\beta_y \leq k$, nous avons $\min\{\beta_x, \beta_y\} \leq k/2$. Sans perdre de généralité, admettons que $\beta_x \leq k/2$ et considérons trois ensembles disjoints $X_1 \cup X_2 \cup X_3 \subseteq X$, chacun de taille β_x. S'il n'existe pas dans F de stable avec β_x sommets dans X_i ($1 \leq i \leq 3$) et β_y sommets dans Y, alors la somme des degrés des sommets de X_i doit être supérieure à $k+1$. En effet, si tel n'est pas le cas, β_y sommets de Y ne sont connectés à aucun sommet de X_i (puisque $Y = k + \beta_y$) et le stable recherché existe. Ainsi, nous obtenons la contradiction suivante : le nombre d'arêtes dans F doit être supérieur à $3k+3$, alors que la propriété locale de faible densité en impose pas plus de $2k + \beta_x + \beta_y - 1 \leq 3k - 1$. □

Proposition 4.10 *Les graphes scindés partagent la propriété de redécoupage, quel que soit k.*

Preuve. Considérons une partition d'un graphe scindé tel que tous les stables soient de taille au moins k. Comme le graphe est scindé, tous les sommets n'appartenant pas à une clique maximum induisent nécessairement un stable. Après avoir extrait autant de stables de taille k que ce qu'il y a de sommets dans la clique maximum, l'ensemble des sommets restants peut donc être découpé de façon optimale. □

Corollaire 4.14 *Le problème* Ordo *peut être résolu en temps et espace linéaire pour les forêts et les graphes scindés, étant donnée en entrée une coloration du graphe où chaque couleur apparaît au moins k fois.*

Valider la conjecture précédente nécessite l'extension des assertions (3) et (7) du lemme de découpage au cas $k = 5$, c'est-à-dire un réponse à la question : *peut-on extraire de cinq stables disjoints de taille six un stable avec un sommet dans chacun de ces cinq stables ?* La démonstration que nous donnons de l'assertion (7) est longue et fastidieuse et nous ne pensons pas pouvoir l'étendre. En revanche, voici une piste de travail que nous avons commencé à explorer et dont l'issue nous semble plus favorable.

Nous montrons qu'un lemme moins fort que le lemme du carré suffit à l'obtention de l'assertion (3) du lemme 4.6 de découpage. Bien que les lemmes du carré et du quasi-carré impliquent celui-ci, nous donnons la preuve suivante, plus directe.

Lemme 4.13 (Lemme affaibli du quasi-carré) *Soit A, B deux stables disjoints dans un graphe triangulé. Si A et B sont chacun de taille au moins quatre, alors il existe un quasi-carré dans (A, B).*

Preuve. D'après le lemme 4.8 des doublets, les deux stables A et B induisent trois doublets notés respectivement $\{a_1, b_1\}$, $\{a_2, b_2\}$ et $\{a_3, b_3\}$. S'il n'existe aucun quasi-carré dans (A, B), alors chaque paire a_i, b_j de sommets ($1 \leq i, j \leq 3$) avec $i \neq j$ doit être reliée par une arête. Ainsi, nous obtenons la contradiction suivante : le sous-graphe induit par les trois doublets doit contenir six arêtes, alors que la propriété locale de faible densité en impose moins de cinq. □

L'assertion (3) du lemme de découpage s'obtient alors de la manière suivante. Soit A, B, C trois stables disjoints dans un graphe triangulé, chacun de taille au moins quatre.

D'après le lemme précédent, il existe dans (A, B) un quasi-carré $\{a_1, b_1, a_2, b_2\}$ avec $a_1, a_2 \in A$, $b_1, b_2 \in B$ et a_1 non connecté à b_2.

Supposons qu'il n'existe aucun stable de taille trois avec un sommet dans $\{a_1, a_2\}$, un sommet dans $\{b_1, b_2\}$ et un sommet dans $C = \{c_1, c_2, c_3, c_4\}$. Clairement, chaque sommet de C doit être touché par au moins une arête provenant de chacun des trois doublets du quasi-carré (si tel n'est pas le cas, il existe immédiatement un triplet dans (A, B, C)). Ainsi, nous pouvons affirmer que (i) $d(a_1)+d(b_1) \geq 4$, $d(b_1)+d(a_2) \geq 4$ et $d(a_2)+d(b_2) \geq 4$. D'un autre côté, l'application de la propriété locale de faible densité dans les deux sous-graphes induits respectivement par les stables $\{a_1, b_1, a_2\}$ et C et par les stables $\{b_1, a_2, b_2\}$ et C nous fournit les inégalités (ii) $d(a_1)+d(b_1)+d(a_2) \leq 6$ et $d(b_1)+d(a_1)+d(b_2) \leq 6$ (par abus de langage, d dénote ici le degré du sommet dans le sous-graphe en question). En joignant les inégalités (i) et (ii), nous obtenons que chaque sommet du quasi-carré $\{a_1, b_1, a_2, b_2\}$ doit être exactement 2-connecté au stable C. À présent, considérons le doublet $\{a_1, b_1\}$ du quasi-carré et admettons sans perdre de généralité que a_1 soit connecté à c_1, c_2 et b_1 soit connecté à c_3, c_4. En appliquant les mêmes arguments au doublet $\{b_1, a_2\}$, nous obtenons alors que le sommet a_2 doit être connecté à c_1, c_2, ce qui impossible sans violer la propriété locale de faible densité.

Cette technique de preuve semble pouvoir s'étendre au cas $k = 4$, même si cela est moins clair qu'avec $k = 3$. Soit A, B, C, D quatre stables disjoints dans un graphe triangulé, chacun de taille au moins cinq. Admettons qu'il existe un ensemble $R = \{a_1, b_1, c_1, a_2, b_2, c_2\}$ dans (A, B, C, D) tel que les sous-ensembles $\{a_1, b_1, c_1\}$, $\{b_1, c_1, a_2\}$, $\{c_1, a_2, b_2\}$ et $\{a_2, b_2, c_2\}$ induisent chacun un stable. En appliquant la technique utilisée précédemment, nous obtenons le système d'inégalités suivant pour les sommets de l'ensemble R :

$$\left\{\begin{array}{ccccccccccccl}
d(a_1) & + & d(b_1) & + & d(c_1) & & & & & & & & \geq 5 \\
 & & d(b_1) & + & d(c_1) & + & d(a_2) & & & & & & \geq 5 \\
 & & & & d(c_1) & + & d(a_2) & + & d(b_2) & & & & \geq 5 \\
 & & & & & & d(a_2) & + & d(b_2) & + & d(c_2) & & \geq 5 \\
d(a_1) & + & d(b_1) & + & d(c_1) & + & d(a_2) & & & & & & \leq 8 \\
 & & d(b_1) & + & d(c_1) & + & d(a_2) & + & d(b_2) & & & & \leq 8 \\
 & & & & d(c_1) & + & d(a_2) & + & d(b_2) & + & d(c_2) & & \leq 8
\end{array}\right.$$

De ce système nous pouvons déduire que chaque sommet de l'ensemble R doit être 3-connecté au stable D. Une étude de cas (dont le détail est sans intérêt ici) nous permet alors de conclure qu'il existe un quadruplet dans (A, B, C, D). Par analogie aux quasi-carrés, l'ensemble R sera appelé *quasi-rectangle*. Suite à cette discussion, nous avons donc l'assertion suivante.

Lemme 4.14 *Soit A, B, C, D quatre stables dans un graphe triangulé, chacun de taille au moins cinq. S'il existe un quasi-rectangle dans (A, B, C, D), alors il existe un quadruplet dans (A, B, C, D).*

Remarque. Nous pensons que l'utilisation raisonnée d'un ordinateur peut être utile à la validation ou au rejet des deux conjectures que nous avons émises. Nous avons par exemple fait correspondre aux assertions (3) et (7) un programme linéaire en nombres entiers dont une solution induirait un contre-exemple. Les variables 0/1 du programme correspondent aux arêtes du sous-graphe induit par les quatre stables et ses contraintes décrivent la propriété locale de faible densité ainsi que la non existence du stable que l'on recherche. Dans le cas de l'assertion (7), la résolution du programme (dont nous savons à présent qu'il n'a pas de solution) devient toutefois très difficile et nécessite l'ajout de coupes correspondant aux différents lemmes que nous avons pu établir (lemme du carré, lemme du triplet, etc.), ainsi qu'à l'élimination des symétries.

Pour conclure, nous proposons une conjecture plus forte encore que la précédente, tendant à unifier tous les résultats de cette section. En effet, la propriété énoncée dans cette seconde conjecture, dérivée de la propriété locale de faible densité, est partagée par les graphes sans $K_{1,3}$, les graphes d'arcs et les graphes triangulés.

Conjecture. Les graphes dont tout sous-graphe induit par deux stables disjoints A et B ne comporte pas plus de $|A| + |B|$ arêtes, partagent la propriété de redécoupage, quel que soit k.

4.4 Synthèse des résultats

Pour clore le chapitre, voici une brève synthèse des résultats obtenus concernant la propriété de redécoupage et sa complexité.

	Graphes sans $K_{1,3}$	Graphes d'intervalles	Graphes d'arcs
$k \geq 2$	$O(n^2/k)$	$O(n+m)$	$O(n+m)$

FIG. 4.17 – La complexité du redécoupage pour les graphes sans $K_{1,3}$ et les graphes d'arcs.

	Graphes de tolérances propres	Graphes de tolérances
$k = 2$	$O(n+m)$	contre-exemple
$k \geq 3$	*ouvert* (même si unitaires et bornées)	contre-exemple

FIG. 4.18 – La complexité du redécoupage pour les graphes de tolérances.

	Forêts	Graphes scindés	Graphes triangulés
$k \leq 4$	$O(n+m)$	$O(n+m)$	$O(n+m)$
$k \geq 5$	$O(n+m)$	$O(n+m)$	*ouvert*

FIG. 4.19 – La complexité du redécoupage pour les graphes triangulés.

Contrairement au cas des graphes triangulés, nous pensons qu'il existe des graphes de tolérances unitaires et bornées ne partageant pas la propriété de redécoupage.

Chapitre 5

Sur certains problèmes de partition relatifs aux graphes d'intervalles

Dans ce chapitre, nous abordons certains problèmes de partition touchant de près ou de loin la classe des graphes d'intervalles [1]. Nous étudions tout particulièrement le problème de la *partition d'un graphe d'intervalles, ou d'arcs, en sous-graphes (induits) d'intervalles propres*. Immédiatement, deux questions peuvent être posées concernant ce problème. La première, plutôt soulevée par le mathématicien, est : peut-on trouver de bonnes bornes inférieure et supérieure sur la taille de la partition minimum d'un graphe d'intervalles (ou d'arcs) en sous-graphes d'intervalles propres ? La seconde, plutôt soulevée par l'informaticien, est : existe-t-il un algorithme efficace pour déterminer une telle partition ?

Nous apportons une réponse à la première question au travers du théorème présenté ci-dessous. Bien que la preuve de ce résultat fournisse quelques éléments de réponse à la seconde question, celle-ci reste ouverte.

Théorème 5.1 *Tout graphe d'intervalles, ou même d'arcs, à n sommets admet une partition en moins de $O(\log n)$ sous-graphes d'intervalles propres. De plus, cette borne est atteinte pour une famille infinie de graphes d'intervalles.*

La preuve constructive de ce résultat fournit un algorithme en temps et espace linéaire pour calculer une telle partition. De fait, ce résultat peut avoir des applications dans la mise au point d'algorithmes d'approximation pour certains problèmes difficiles relatifs aux graphes d'intervalles ou d'arcs. En effet, beaucoup de problèmes difficiles pour ces graphes deviennent plus faciles à traiter dès lors qu'il s'agit de graphes d'intervalles propres. Dans la deuxième section, nous présenterons une application de ce type à la croisée des domaines de l'ordonnancement de tâches et de la planification de personnel, à l'origine des travaux présentés dans ce chapitre. Enfin, nous discuterons dans la troisième section de quelques problèmes et résultats relatifs à la partition d'un graphe d'intervalles en sous-graphe d'intervalles propres. Mais tout d'abord, voici la démonstration du théorème 5.1.

[1] Les résultats présentés dans ce chapitre apparaissent en partie dans [64] et [67] (en anglais).

5.1 Partitions en sous-graphes d'intervalles propres

Pour commencer, nous donnons une première preuve du théorème, s'appuyant notamment sur une construction en temps et espace linéaire. Par la suite, d'autres types de construction seront exploitées afin d'affiner la borne supérieure du théorème.

Une première borne supérieure

Le lemme suivant nous permet d'obtenir une borne supérieure en fonction de t lorsque le graphe d'intervalles est sans $K_{1,t}$. La plus petite valeur t pour laquelle un graphe d'intervalles est sans $K_{1,t}$ peut être déterminée en temps $O(n^2)$ comme suit. Pour chaque intervalle I_i $(1 \leq i \leq n)$, calculer la taille t_i du plus grand stable parmi tous les intervalles intersectés par I_i : il est alors assez facile de voir que $t = \max\{t_i \mid 1 \leq i \leq n\}$. Les intervalles de la représentation étant ordonnés, le calcul d'un stable maximum ne prend qu'un temps $O(n)$ [81].

Lemme 5.1 (Lemme de partition linéaire) *Soit G un graphe d'intervalles sans $K_{1,t}$, avec $t > 1$. Alors G admet une partition en moins de $\lfloor t/2 \rfloor$ sous-graphes d'intervalles propres. De plus, cette partition s'obtient en temps et espace linéaire.*

Preuve. Un algorithme est proposé pour la construction de cette partition. De façon synthétique, l'algorithme extrait et colore linéairement des cliques de G avec l'ensemble de couleurs $\{0, \ldots, \lfloor t/2 \rfloor - 1\}$; la sortie de l'algorithme sera la partition de G induite par ces $\lfloor t/2 \rfloor$ couleurs.

Algorithme COLORER-CLIQUES ;
Entrée : un graphe d'intervalles G sans $K_{1,t}$ $(t > 1)$;
Sortie : une partition de G en $\lfloor t/2 \rfloor$ sous-graphes d'intervalles propres ;
Début ;
  calculer une représentation par intervalles $I_1, \ldots, I_n$ of G, ordonnée selon $<_g$;
  $\mathcal{C}_0 \leftarrow \cdots \leftarrow \mathcal{C}_{\lfloor t/2 \rfloor - 1} \leftarrow \emptyset$, $i \leftarrow 1$, $j \leftarrow 0$;
  tant que $i \leq n$ **faire**
    $C_j \leftarrow \{I_i\}$, $d_{clique} \leftarrow d(I_i)$, $i \leftarrow i + 1$;
    tant $i \leq n$ et $g(I_i) \leq d_{clique}$ **faire**
      $C_j \leftarrow C_j \cup \{I_i\}$;
      si $d(I_i) < d_{clique}$ **alors** $d_{clique} \leftarrow d(I_i)$;
      $i \leftarrow i + 1$;
    $\mathcal{C}_{j \bmod \lfloor t/2 \rfloor} \leftarrow \mathcal{C}_{j \bmod \lfloor t/2 \rfloor} \cup \{C_j\}$, $j \leftarrow j + 1$;
  retourner $\mathcal{C}_0, \ldots, \mathcal{C}_{\lfloor t/2 \rfloor - 1}$;
Fin ;

Puisque le calcul d'une représentation par intervalles ordonnée se fait en temps et espace $O(n+m)$ [32, 82], l'algorithme tout entier peut s'exécuter en temps et espace linéaire. Nous établissons maintenant sa validité en montrant que chaque classe de couleur $\mathcal{C}_r$ $(0 \leq r \leq$

$\lfloor t/2 \rfloor - 1$) induit un graphe d'intervalles propres. Soit $\mathcal{C}_r = \{C_1^r, \ldots, C_q^r\}$ l'ensemble des cliques affectées à $\mathcal{C}_r$ par l'algorithme, dans l'ordre de leur extraction. Si $q \leq 2$, alors $\mathcal{C}_r$ est immédiatement sans $K_{1,3}$. Dans le cas contraire, supposons que $\mathcal{C}_c$ contienne une copie induite de $K_{1,3}$ avec I_a, représentant son sommet central, et $I_b \prec I_c \prec I_d$ ses trois feuilles. Clairement, ces feuilles appartiennent à des cliques disjointes et nous pouvons poser $I_b \in C_u^r$, $I_c \in C_v^r$ et $I_d \in C_w^r$ avec $1 \leq u < v < w \leq q$. Par l'algorithme, I_a appartient nécessairement à C_u^r.

À présent, sélectionnons dans chaque clique C_j coloriée par l'algorithme entre C_u^r et C_w^r l'intervalle de plus petite extrémité droite et ajoutons-le à un ensemble S, initialement vide. Nous affirmons que S induit un stable de taille au moins $2\lfloor t/2 \rfloor + 1$. En effet, si deux intervalles de S se chevauchent, alors ils appartiennent à une clique de même couleur, ce qui est une contradiction. Au moins $\lfloor t/2 \rfloor$ cliques sont colorées par l'algorithme entre C_u^r et C_v^r (non compris) et encore $\lfloor t/2 \rfloor$ entre C_v^r et C_w^r (non compris). Par conséquent, S contient au moins $2\lfloor t/2 \rfloor + 1$ éléments et l'assertion est vérifiée.

Comme l'intervalle $I_a \in C_u^r$ chevauche l'intervalle $I_d \in C_w^r$, celui-ci chevauche aussi tous les intervalles de S, excepté peut-être le dernier intervalle (en partant de la gauche), qui appartient à C_w^r. Ce dernier intervalle est alors remplacé par I_d, qui ne chevauche pas l'avant-dernier intervalle de S (en effet, si ce n'était pas le cas, I_d n'appartiendrait pas à C_w^r). Le stable S ainsi formé est de taille $2\lfloor t/2 \rfloor + 1 \geq t$, pour tout $t > 1$. Nous obtenons ainsi qu'au moins t intervalles disjoints sont intersectés par I_a, ce qui contredit le fait que G est sans $K_{1,t}$. Par conséquent, l'ensemble $\mathcal{C}_r$ induit bien un graphe d'intervalles sans $K_{1,3}$, *i.e.* un graphe d'intervalles propres d'après le théorème de Roberts [131] (voir aussi le théorème 1.3), et la correction de l'algorithme est démontrée. □

Remarque. L'algorithme COLORER-CLIQUES affecte les couleurs dans l'ordre $0, \ldots, \lfloor t/2 \rfloor - 1$. En fait, l'algorithme reste correct en prenant comme motif n'importe quelle permutation de l'ensemble $\{0, \ldots, \lfloor t/2 \rfloor - 1, 0, \ldots, \lfloor t/2 \rfloor - 1\}$, répétée autant de fois que cela est nécessaire pour compléter l'affectation (la preuve reste la même que celle que nous avons exposée ci-dessus). Cela implique notamment qu'il existe au moins $(2t')!/(2^{t'} t'!)$ partitions non isomorphes d'un graphe d'intervalles sans $K_{1,t}$ en $t' = \lfloor t/2 \rfloor$ sous-graphes d'intervalles propres. Par exemple, si le graphe admet une partition en deux sous-graphes d'intervalles propres, les cliques peuvent être colorées à l'aide d'une des trois séquences de couleurs suivantes : $\{0, 1, 0, 1\}$, $\{0, 1, 1, 0\}$ ou $\{0, 0, 1, 1\}$. Tout autre séquence valide sera nécessairement isomorphe à une de ces trois.

L'ensemble des cliques $C_1, \ldots, C_j$ extraites de G forme clairement une partition de G en cliques. En fait, cette partition, que nous appellerons par la suite *partition en cliques canonique*, induit une partition minimum de G en cliques. En effet, en prenant l'intervalle de plus petite extrémité droite dans chaque clique C_j, nous obtenons un stable maximum de G. Par conséquent, le cardinal de cette partition en cliques est nécessairement minimum. Comme nous l'avions déjà précisé (voir la remarque p. 47), le calcul d'un stable maximum ou d'une partition minimum en cliques ne prend donc qu'un temps $O(n)$, si une représentation par intervalles ordonnée selon $<_g$ (ou $<_d$) est donnée en entrée.

Lemme 5.2 *Tout graphe d'intervalles admet une partition en moins de $\lceil \log_3((n+1)/2) \rceil$ sous-graphes d'intervalles sans $K_{1,5}$. De plus, cette partition s'obtient en temps et espace* *linéaire.*

Preuve. D'après la proposition 1.1, une représentation $I_1, \ldots, I_n$ de G par intervalles ouverts, dont les extrémités sont dans $\{1, \ldots, n\}$, s'obtient en temps et espace linéaire. Notons ℓ la longueur maximum d'un intervalle de cette représentation ($\ell \leq n-1$). Maintenant, effectuons une partition des intervalles selon leur taille, comme suit : l'ensemble $\mathcal{I}_1$ contient les intervalles de longueur $\{1, 2, 3, 4\}$, l'ensemble $\mathcal{I}_2$ contient les intervalles de longueur $\{5, \ldots, 16\}$, et plus généralement, l'ensemble $\mathcal{I}_i$ contient les intervalles de longueur $\{2 \cdot 3^{i-1} - 1, \ldots, 2 \cdot 3^i - 2\}$. De cette façon, nous obtenons $\lceil \log_3((\ell + 2)/2) \rceil$ ensembles, dont nous pouvons montrer que chacun induit un graphe d'intervalles sans $K_{1,5}$.

Supposons qu'un ensemble $\mathcal{I}_i$ ne satisfasse pas cette condition. Cela implique qu'un intervalle de $\mathcal{I}_i$ contient proprement au moins trois intervalles disjoints. L'intervalle le plus grand de $\mathcal{I}_i$ étant de longueur $2 \cdot 3^i - 2$, la somme des longueurs de ces trois intervalles doit donc être inférieure à $2 \cdot 3^i - 2$. D'un autre côté, l'intervalle le plus petit de $\mathcal{I}_i$ étant de longueur $2 \cdot 3^{i-1} - 1$, la somme des longueurs de ces trois intervalles ne peut être supérieure à $2 \cdot 3^i - 3$, ce qui contredit l'affirmation précédente. $\square$

Remarque. La démonstration peut être réalisée à l'aide d'une représentation par intervalles fermés, dont les extrémités sont dans $\{1, \ldots, 2n\}$. La partition doit alors se faire de manière à ce que l'ensemble $\mathcal{I}_i$ contienne les intervalles de longueur $\{4 \cdot 3^{i-1} - 3, \ldots, 4 \cdot 3^i - 4\}$ pour $i = 1, \ldots, \lceil \log_3((\ell + 4)/4) \rceil$ (ici $\ell \leq 2n - 1$). En fait, une généralisation de la preuve permet de montrer que tout graphe d'intervalles admet une partition en $O(\log_t n)$ sous-graphes d'intervalles sans $K_{1,t+2}$, pour tout entier $t > 1$.

Proposition 5.1 (Borne supérieure) *Tout graphe d'intervalles (resp. d'arcs) admet une partition en moins de* $2\lceil \log_3((n + 1)/2) \rceil$ *(resp.* $2\lceil \log_3((n + 1)/2) \rceil + 1$*) sous-graphes d'intervalles propres, pour* $n > 1$*. De plus, cette partition s'obtient en temps et espace linéaire.*

Preuve. La preuve de la borne supérieure pour les graphes d'intervalles découle immédiatement de la combinaison du lemme 5.2 et du lemme 5.1 de partition linéaire pour $t = 5$. Quant aux graphes d'arcs, nous procédons de la façon suivante. Tout d'abord, calculons en temps et espace linéaire une représentation par arcs du graphe [120]. Ensuite, choisissons un point p sur le cercle et déterminons l'ensemble V' des sommets correspondant aux arcs qui contiennent p. Il suffit alors d'observer que V' induit une clique et que le sous-graphe induit par $V \setminus V'$ est un graphe d'intervalles pour obtenir la borne recherchée (toute clique induit de façon triviale un graphe d'intervalles propres). $\square$

Voici un graphe d'intervalles pour lequel la construction proposée ci-dessus retourne une partition en $f(n) = 2\lfloor \log_3((n + 1)/2) \rfloor$ sous-graphes d'intervalles propres. Pour simplifier la construction, nous considérerons que n est un multiple de $f(n)$ et nous poserons $g(n) = n/f(n) - 1$. Le graphe d'intervalles en question est représenté par l'ensemble d'intervalles ouverts suivant : pour tout $i = 1, \ldots, f(n)/2$, soit un intervalle $]1, 2 \cdot 3^i - 1[$, un intervalle $]1, 2 \cdot 3^{i-1}[$, $g(n)$ intervalles $]2 \cdot 3^{i-1}, 4 \cdot 3^{i-1} - 1[$ et $g(n)$ intervalles $]4 \cdot 3^{i-1} - 1, 2 \cdot 3^i - 2[$. Le lecteur notera que les extrémités de tous ces intervalles appartiennent à l'ensemble $\{1, \ldots, n\}$ et que le graphe n'est pas un graphe d'intervalles propres (voir la figure 5.1 pour un exemple de construction).

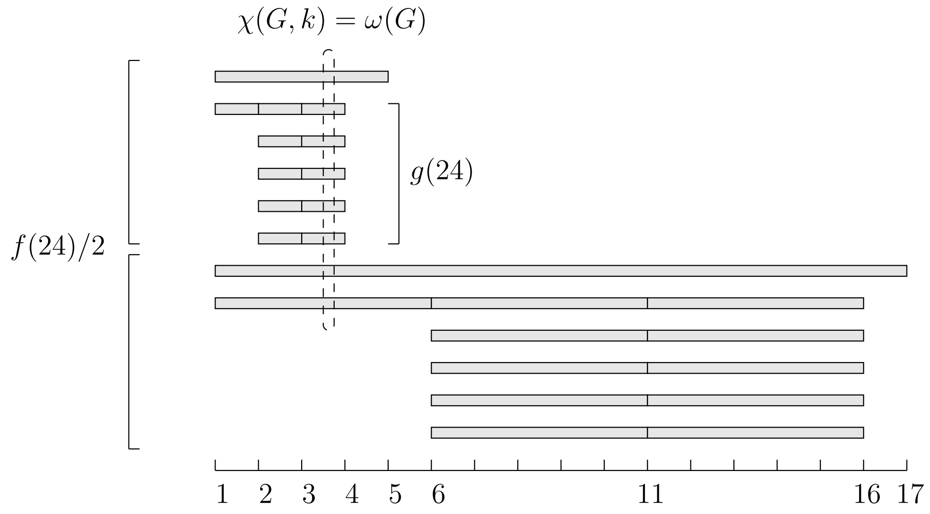


FIG. 5.1 – Un exemple de construction avec $n = 24$ ($f(24) = 4$, $g(24) = 5$).

L'application du lemme 5.2 nous donne une partition en $f(n)/2$ ensembles contenant chacun les intervalles d'indice i dans la définition précédente. Ensuite, chacun de ces ensembles admet une partition en deux sous-graphes d'intervalles propres par application du lemme de partition linéaire avec $t = 5$. Cette seconde partition est de surcroît la meilleure possible puisque ces ensembles contiennent chacun une copie induite de $K_{1,3}$. Ainsi, la partition obtenue après l'application des deux lemmes est exactement de cardinal $f(n)$. D'un autre côté, la meilleure partition possible est de cardinal deux, composée d'une part des intervalles $]1, 2 \cdot 3^i - 1[$ et $]1, 2 \cdot 3^{i-1}[$ pour $i = 1, \dots, f(n)/2$ et d'autre part des intervalles $]2 \cdot 3^{i-1}, 4 \cdot 3^{i-1} - 1[$ et $]4 \cdot 3^{i-1} - 1, 2 \cdot 3^i - 2[$ pour $i = 1, \dots, f(n)/2$. En effet, le premier ensemble induit clairement une clique (tous ses intervalles contiennent le point d'abscisse $1 + \epsilon$, avec $\epsilon > 0$) et le second est une union de cliques disjointes (les intervalles d'indice i ne peuvent en aucun cas chevaucher ceux d'indice $i - 1$).

La borne inférieure

Comme en témoigne la proposition suivante, la borne supérieure que nous venons d'établir n'est pas loin d'être la meilleure possible.

Proposition 5.2 (Borne inférieure) *Pour tout entier $k \geq 1$, il existe un graphe k-parti H_k à $n = (3^k - 1)/2$ sommets qui n'admet aucune partition en moins de $k = \log_3(2n + 1)$ sous-graphes d'intervalles propres.*

Preuve. Une représentation par intervalles de ce graphe H_k s'obtient en définissant récursivement k stables $S_1, \dots, S_k$ comme suit. Le stable S_1 consiste en un unique intervalle ouvert. Pour $i = 2, \dots, k$, le stable S_i est obtenu en copiant le stable S_{i-1}, puis en subdivisant chacun de ses intervalles en trois intervalles ouverts de taille égale (voir la figure 5.2 pour un exemple de construction). L'ensemble $S_1, \dots, S_k$ des stables résultant de cette construction induit un graphe k-parti dont le nombre de sommets est $n = \sum_{i=1}^{k} 3^{i-1} = (3^k - 1)/2$.

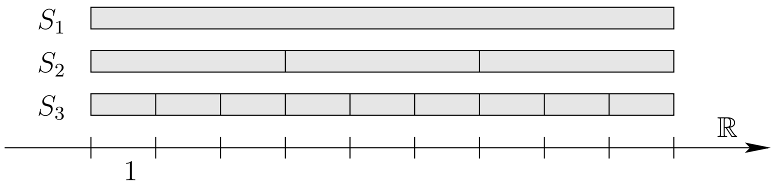


FIG. 5.2 – Une représentation du graphe H_3 par intervalles.

Comme chaque stable induit de façon triviale un sous-graphe d'intervalles propres, H_k admet immédiatement une partition en k sous-graphes d'intervalles propres. Nous allons montrer par induction que la plus petite partition de H_k en sous-graphes d'intervalles propres est en fait de cardinal $\xi(H_k) = k$. Nous prendrons donc $\xi(H_{i-1}) = i - 1$ comme base de l'induction pour $i > 2$; le lecteur vérifiera aisément que $\xi(H_1) = 1$ et $\xi(H_2) = 2$. À présent, supposons que $\xi(H_i) < i$ et considérons une partition de H_i en $i - 1$ ensembles $\mathcal{I}_1, \ldots, \mathcal{I}_{i-1}$ d'intervalles, induisant chacun un graphe d'intervalles propres.

Considérons, sans perte de généralité, que l'unique intervalle $I^* \in S_1$ appartienne à l'ensemble $\mathcal{I}_1$. Nous affirmons alors que les intervalles de l'ensemble $\mathcal{I}_1 \backslash \{I^*\}$ induisent au plus deux cliques disjointes. En effet, le contraire impliquerait l'existence d'une copie induite de $K_{1,3}$ dans $\mathcal{I}$ (avec I^* comme sommet central et un intervalle de chacune des cliques disjointes comme feuille). Cette affirmation implique qu'au moins un intervalle de S_2 ainsi que tous les intervalles issus de sa subdivision dans les stables $S_3, \ldots, S_i$ n'appartiennent pas à l'ensemble $\mathcal{I}_1$. Or, ces intervalles induisent un exemplaire du graphe H_{i-1}, dont la partition en sous-graphes d'intervalles propres nécessite, d'après l'hypothèse d'induction, $i-1$ ensembles. Nous obtenons en conséquence une contradiction, puisque seuls les $i - 2$ ensembles $\mathcal{I}_2, \ldots, \mathcal{I}_{i-1}$ sont disponibles pour réaliser cette partition. Ainsi, nous avons $\xi(H_i) = i$ pour $i > 2$ et la preuve par induction est complétée. Enfin, l'égalité $n = (3^k - 1)/2$ nous permet de conclure que $\xi(H_k) = \log_3(2n + 1)$. □

Remarque. La construction du graphe H_k peut être réalisée à l'aide d'intervalles ayant tous des extrémités entières entre 0 et $(2n + 1)/3$ (qui ici est entier puisque $n - 1$ est multiple de trois). Il suffit pour cela de définir la longueur des intervalles de S_k comme égale à un (l'unique intervalle de S_1, le plus long de la représentation, sera alors de longueur 3^{k-1}).

Corollaire 5.1 (Borne inférieure) *Pour tout entier $n \geq 1$, il existe un graphe d'intervalles qui n'admet aucune partition en moins de $\lfloor \log_3(2n + 1) \rfloor$ sous-graphes d'intervalles propres.*

Preuve. Définissons le graphe H_k avec k le plus grand entier tel que $n \geq (3^k - 1)/2$, puis ajoutons à celui-ci exactement $n - (3^k - 1)/2$ sommets isolés. Par la proposition précédente, ce graphe n'admet aucune partition en moins de $\lfloor \log_3(2n + 1) \rfloor$ sous-graphes d'intervalles propres. □

Corollaire 5.2 *Pour tout entier $t \geq 2$, il existe un graphe d'intervalles sans $K_{1,t}$, possédant au plus $\lfloor (3t - 4)/2 \rfloor$ sommets, qui n'admet aucune partition en moins de $\lfloor \log_3(t - 1) \rfloor + 1$ sous-graphes d'intervalles propres.*

Preuve. Il suffit de remarquer que le graphe H_k défini dans la proposition précédente est sans $K_{1,t}$ pour $t \in \{3^{k-1}+1, \ldots, 3^k\}$. En exprimant n et $\xi(H_k)$ en fonction de t, nous obtenons que H_k contient au plus $\lfloor (3t-4)/2 \rfloor$ sommets et n'admet aucune partition en moins de $\lfloor \log_3(t-1) \rfloor + 1$ sous-graphes d'intervalles propres pour tout $t \in \{3^{k-1}+1, \ldots, 3^k\}$, quelque soit $k \geq 1$. □

Raffinement de la borne supérieure

Voici une nouvelle construction, légèrement plus coûteuse en temps, permettant d'abaisser significativement la borne du lemme 5.1 de partition linéaire et de là, la borne supérieure de la proposition 5.1.

Lemme 5.3 (Lemme de partition logarithmique) *Soit G un graphe d'intervalles sans $K_{1,t}$ avec $t > 1$. Alors G admet une partition en moins de $f_0(t) = \lceil \log_3((t-1)/2) \rceil + 1$ sous-graphes d'intervalles propres. De plus, cette partition s'obtient en temps $O(n \log t + m)$ et espace linéaire.*

Preuve. Voici l'algorithme permettant d'obtenir une telle partition. Par abus de langage, nous définirons ici l'extrémité gauche $g(C_j)$ (resp. extrémité droite $d(C_j)$) d'une clique C_j comme la plus grande (resp. petite) extrémité gauche (resp. extrémité droite) d'un intervalle dans C_j. Ainsi, nous dirons qu'un intervalle traverse la clique C_j quand il chevauche la portion de droite $[g(C_j), d(C_j)]$.

Algorithme Colorer-Cliques++ ;
Entrée : un graphe d'intervalles G sans $K_{1,t}$ $(t > 1)$;
Sortie : une partition de G en $f_0(t) = \lceil \log_3((t-1)/2) \rceil$ sous-graphes d'intervalles propres ;
Début ;
  calculer une représentation par intervalles $I_1, \ldots, I_n$ de G, ordonnée selon $<_g$;
  calculer une partition canonique $C_0, \ldots, C_{q-1}$ de G en cliques ;
  $i^* \leftarrow \lceil \log_3((t-1)/2) \rceil$, $\mathcal{C}_0 \leftarrow \cdots \leftarrow \mathcal{C}_{i^*} \leftarrow \emptyset$, $i \leftarrow i^*$;
  tant que $i \geq 0$ **faire**
    pour j de 0 à $q-1$ par pas de 3^i **faire**
      inclure dans $\mathcal{C}_i$ tous les intervalles non marqués traversant la clique C_j ;
      marquer les intervalles qui viennent d'être inclus dans $\mathcal{C}_i$;
    $i \leftarrow i - 1$;
  retourner $\mathcal{C}_0, \ldots, \mathcal{C}_{i^*}$;
Fin ;

Tout d'abord, analysons la complexité de l'algorithme. Comme nous l'avons vu précédemment, calculer une représentation par intervalles ordonnée se fait en temps et espace linéaire [32, 82]. De là, la partition canonique de G en cliques s'obtient en temps et espace $O(n)$. Correctement implantée, la recherche des intervalles non marqués qui traversent les cliques sélectionnées par la boucle **pour** ne prend qu'un temps $O(n)$. En effet, les intervalles non marqués qui traversent une clique C_j ont nécessairement leur extrémité gauche dans la portion de droite $]d(C_{j'}), d(C_j)]$ où $C_{j'}$ est la clique précédemment sélectionnée. Par conséquent, un balayage des intervalles dans l'ordre $<_g$ permettra d'effectuer cette recherche en une seule passe. La boucle **tant que** étant répétée $i^* + 1$ fois, soit $\lceil \log_3((t-1)/2) \rceil + 1$ fois, l'algorithme tout entier s'exécutera en temps $O(n \log t + m)$ et espace linéaire.

La validité de l'algorithme est maintenant démontrée. Les intervalles non marqués qui sont inclus dans un ensemble $\mathcal{C}_i$, à chaque pas de la boucle **pour**, forment une clique (ces intervalles traversent tous la clique C_j). Ainsi, chaque ensemble $\mathcal{C}_i$ $(0 \leq i \leq i^* - 1)$ est une union de cliques disjointes. De plus, nous observons que trois cliques ne peuvent avoir été incluses dans l'ensemble $\mathcal{C}_i$ sans qu'aucune clique de $\mathcal{C}_{i+1}$ ne soit intercalée entre deux de ces trois. En effet, les cliques incluses dans $\mathcal{C}_i$ correspondent aux ensembles d'intervalles non marqués qui traversent C_j pour $j = \beta \cdot 3^i$ avec $\beta \geq 0$. Or, les intervalles traversant ces cliques C_j se trouvent déjà inclus dans $\mathcal{C}_{i+1}$, pour tout β multiple de trois. De cette observation, nous pouvons déduire que les ensembles $\mathcal{C}_0, \ldots, \mathcal{C}_{i^*-1}$ d'intervalles induisent chacun un graphe d'intervalles sans $K_{1,3}$. Admettons qu'un intervalle I_a et trois intervalles $I_b \prec I_c \prec I_d$ induisent une copie de $K_{1,3}$ dans un ensemble $\mathcal{C}_i$ $(0 \leq i \leq i^* - 1)$. Clairement, les trois intervalles I_b, I_c, I_d appartiennent à trois cliques différentes dans $\mathcal{C}_i$ et l'intervalle I_a doit traverser chacune de ces trois cliques. Or, d'après la remarque précédente, I_a traverse aussi une clique de l'ensemble $\mathcal{C}_{i+1}$. Ce dernier ayant été constitué avant l'ensemble $\mathcal{C}_i$, il devrait donc contenir l'intervalle I_a, ce qui est une contradiction.

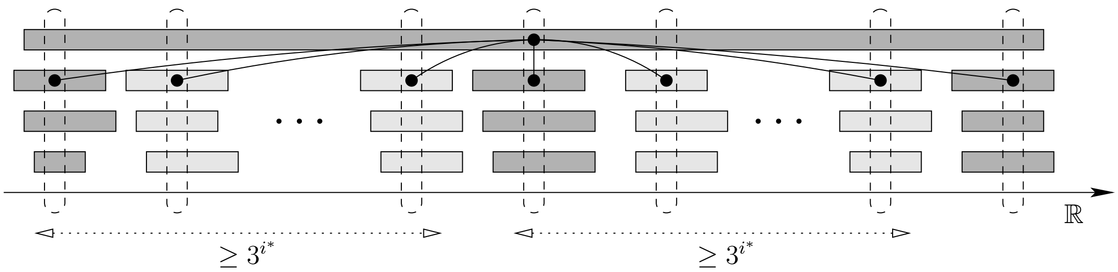


FIG. 5.3 – Une illustration de la preuve $K_{1,3} \Rightarrow K_{1,t}$.

Enfin, à l'aide des mêmes arguments que ceux employés dans la preuve du lemme 5.1 de partition linéaire, nous pouvons montrer que l'existence d'un sous-graphe induit $K_{1,3}$ dans $\mathcal{C}_{i^*}$ implique aussitôt l'existence d'un sous-graphe induit $K_{1,t'}$ dans G avec $t' = 2 \cdot 3^{i^*} + 1 \geq t$ (lorsque $t > 1$), ce qui est une contradiction. Par conséquent, l'ensemble $\mathcal{C}_{i^*}$ induit lui aussi un graphe sans $K_{1,3}$. La preuve de cette affirmation est illustrée par la figure 5.3, les détails de celle-ci étant laissés au lecteur. □

Cette fois, le lemme 5.2 n'est pas nécessaire à l'obtention d'une borne générale logarithmique. Un graphe à n sommets étant trivialement sans $K_{1,n}$, nous obtenons immédiatement la proposition suivante.

Proposition 5.3 (Nouvelle borne supérieure) *Tout graphe d'intervalles (resp. d'arcs) admet une partition en moins de $f_0(n) = \lceil \log_3((3n-3)/2) \rceil$ (resp. $f_0(n) + 1$) sous-graphes d'intervalles propres, pour $n > 1$. De plus, cette partition s'obtient en temps $O(n \log n) + m$ et espace linéaire.*

Le lecteur notera que les bornes supérieures offertes par le lemme 5.3 et la proposition 5.3 sont les meilleures possibles *à un près* dans le cas des graphes d'intervalles, et *à deux près* dans le cas des graphes d'arcs. Toutefois, une utilisation brutale du lemme de partition logarithmique conduit à des résultats peu satisfaisants sur certaines instances, comme par exemple le simple graphe $K_{1,n-1}$. Alors que la meilleure partition possible est de cardinal deux, l'algorithme Colorer-Cliques++ retourne une partition de cardinal $\lceil \log_3((3n-3)/2) \rceil$. En fait, tous les graphes ayant un nombre chromatique en $O(1)$ et possédant une copie induite de $K_{1,t}$ avec $t = \Omega(n)$ sont critiques pour l'algorithme Colorer-Cliques++. Les deux corollaires qui suivent vont nous aider à mieux traiter ce type d'instances et nous permettre ainsi d'affiner encore la borne supérieure dans le cas général. Nous rappelons que $\alpha(G)$ dénote la taille d'un stable maximum dans le graphe G.

Corollaire 5.3 *Soit G un graphe d'intervalles avec $1 < \alpha(G) < t$. Alors G admet une partition en mois de $f_0(t)$ sous-graphes d'intervalles propres. De plus, cette partition s'obtient en temps $O(n \log t + m)$ et espace linéaire.*

Preuve. Tout graphe est de façon triviale sans $K_{1,\alpha(G)+1}$. Comme $1 < \alpha(G) < t$, le graphe G est donc sans $K_{1,t}$ et admet une partition en $f_0(t)$ sous-graphes d'intervalles propres d'après le lemme de partition logarithmique. □

Corollaire 5.4 *Soit G un graphe d'intervalles avec $t \leq \alpha(G) < n-1$. Alors G admet une partition en moins de $f_0(n-t)+1$ sous-graphes d'intervalles propres. De plus, cette partition s'obtient en temps $O(n \log t + m)$ et espace linéaire.*

Preuve. Après avoir calculé une représentation de G par intervalles, ordonnée selon $<_g$, un stable maximum est extrait de G en temps linéaire. Celui-ci étant de taille $\alpha(G) \geq t$, il reste dans G au plus $n-t$ intervalles $(n-t > 1)$, qui admettent une partition en $f_0(n-t)$ sous-ensembles induisant chacun un graphe d'intervalles propres (d'après le lemme de partition logarithmique). Puisque tout stable induit de façon triviale un sous-graphe d'intervalles propres, la preuve du lemme est établie. □

Remarque. Une conséquence intéressante de ces deux corollaires est celle-ci. Soit G un graphe d'intervalles satisfaisant une des deux conditions suivantes, pour $t = O(1)$:

(1) G ne possède pas de copie induite de $K_{1,t}$ (ou de stable de taille supérieure à t) ;
(2) G possède une copie induite de $K_{1,n-t}$ (ou un stable de taille supérieure à $n-t$).

Alors G admet une partition en $O(1)$ sous-graphes d'intervalles propres, calculable en temps et espace linéaire. Notons que cette remarque reste valable même si le lemme de partition linéaire est utilisé pour effectuer la partition.

En joignant les deux corollaires précédents, nous obtenons que tout graphe d'intervalles admet une partition en moins de $f_1(n,t) = \max\{f_0(t), f_0(n-t)+1\}$ pour n'importe quel entier t variant entre 2 et $n-2$. La recherche de la valeur de t pour laquelle $f_1(n,t)$ atteint son minimum nous conduit alors au résultat suivant : tout graphe d'intervalles admet une partition en moins de $f_1(n) = \lceil \log_3((9n-6)/8) \rceil$ sous-graphes d'intervalles propres, pour $n > 1$. De plus, cette partition s'obtient en temps $O(n \log n + m)$ et espace linéaire. Le coefficient multiplicatif de la variable n (à l'intérieur du logarithme) est dorénavant $\frac{9}{8} = 1.125$, et non plus $\frac{3}{2} = 1.5$ comme dans la proposition 5.3. Voici comment nous obtenons le minimum de la fonction $f(n,t) = \max\{f_0(t), f_0(n-t)+1\}$ sur l'ensemble $t \in \{2,\dots,n-2\}$. Les fonctions $f_0(t)$ et $f_0(n-t)+1$ étant respectivement monotone croissante et monotone décroissante sur leur ensemble de définition, une borne inférieure du minimum recherché est donnée par l'équation $\tilde{f}_0(\tilde{t}) = \tilde{f}_0(n-\tilde{t})+1$ pour $\tilde{t} \in [2,\dots,n-2]$ où $\tilde{f}_0(\tilde{t}) = \log_3((3\tilde{t}-3)/2)$. Un simple calcul permet alors de vérifier que cette équation a pour unique solution $\tilde{t} = (3n-2)/4$. De là, un minimum de $f(n,t)$ s'obtient en prenant le minimum parmi les deux valeurs $f_0(\lceil\tilde{t}\rceil)$ et $f_0(n-\lfloor\tilde{t}\rfloor)+1$, que l'on peut borner de la façon suivante :

$$\begin{array}{rcl} f_0(\lceil\tilde{t}\rceil) & \leq & \lceil\log_3((9n-6)/8)\rceil \\ f_0(n-\lfloor\tilde{t}\rfloor)+1 & \leq & \lceil\log_3((9n+18)/8)\rceil \end{array}$$

La meilleure borne possible est donc $f(n) = \lceil\log_3((9n-6)/8)\rceil$ obtenu avec la valeur $t = \lceil(3n-2)/4\rceil$ (pour $n \geq 6$). La borne est ensuite étendue jusqu'à $n > 1$ en vérifiant que $f(2) = f(3) = 1$ et $f(4) = f(5) = 2$.

Cet emploi conjugué des corollaires 5.3 et 5.4 peut être généralisé par l'algorithme récursif suivant. Cet algorithme prend en entrée une représentation ordonnée $\mathcal{I}$ d'un graphe d'intervalles G et retourne en sortie une partition de G en sous-graphes d'intervalles propres.

Algorithme Colorer-Cliques-Récursif ;
Entrée : une représentation $\mathcal{I}$ du graphe G par intervalles (ordonnée selon $<_g$) ;
Sortie : une partition de G en sous-graphes d'intervalles propres ;
Début ;
  si $|\mathcal{I}| \leq 3$ **alors retourner** $\mathcal{I}$;
  sinon
    choisir une valeur de t dans $\{2,\dots,n-2\}$;
    calculer un stable maximum S dans $\mathcal{I}$;
    si $|S| < t$ **alors retourner** Colorer-Cliques++($\mathcal{I}$) ;
    sinon
      $\mathcal{I} \leftarrow \mathcal{I} \setminus S$;
      retourner $S \cup$ Colorer-Cliques-Récursif($\mathcal{I}$) ;
Fin ;

Comme précédemment, nous allons rechercher les valeurs de t pour lesquelles la partition retournée par l'algorithme est de plus petit cardinal. Tout d'abord, imaginons que l'algorithme soit appelé récursivement i fois, puis interrompu lors d'un dernier appel à la procédure Colorer-Cliques++ sur l'ensemble des intervalles restant dans $\mathcal{I}$, quelque soit la valeur de $|S|$. Nous numérotons ces appels récursifs de 0 à i inversement à l'ordre chronologique (l'appel 0 sera donc le dernier appel et l'appel i le premier). À chaque appel $j = 0,\dots,i$ nous associons les entiers t_j et n_j correspondant aux valeurs de t et de n lors de cet appel. Ainsi, le cardinal de la partition retournée par l'algorithme sera

$$f_i(n,t_0,t_1,\dots,t_i) = \max\{f_0(t_i), f_0(t_{i-1})+1,\dots,f_0(t_j)+i-j,\dots,f_0(t_0)+i, f_0(n_0-t_0)+i+1\}$$

Nous devons maintenant déterminer les valeurs de $t_0,t_1,\dots,t_i$ pour lesquelles la fonction $f_i(n,t_0,t_1,\dots,t_i)$ atteint un minimum. Nous montrons par récurrence sur i que ces valeurs sont telles que

$$t_j = \left\lceil \frac{2 \cdot 3^j}{3^{j+1} - 1} n + O(1) \right\rceil$$

pour $j = 0, 1, \ldots, i$ et permettent d'obtenir

$$f_i(n) \leq \left\lceil \log_3 \left(\frac{3^{i+1}}{3^{i+1} - 1} n + O(1) \right) \right\rceil$$

Pour $i = 0$ et $i = 1$ (pas d'appel récursif ou un seul), nous avons démontré précédemment que $f_0(n) = \lceil \log_3((3n-3)/2) \rceil$ et $f_1(n) = \lceil \log_3((9n-6)/8) \rceil$ pour $t_0 = n$ et $t_1 = \lceil (3n-2)/4) \rceil$. Maintenant, supposons que la propriété soit vraie pour $i - 1 \geq 0$. Nous cherchons à évaluer

$$f_i(n) = \min_{t_0, t_1, \ldots, t_i} \max\{f_0(t_i), f_0(t_{i-1}) + 1, \ldots, f_0(t_0) + i, f_0(n_0 - t_0) + i + 1\}$$

qui peut être borné de la façon suivante, en utilisant l'hypothèse de récurrence :

$$f_i(n) \leq \min_{t_i} \max\{f_0(t_i), f_{i-1}(n - t_i) + 1\}$$

La recherche du minimum se fait comme précédemment en résolvant l'équation $\tilde{f}_0(\tilde{t}_i) = \tilde{f}_{i-1}(n - \tilde{t}_i) + 1$ pour $\tilde{t}_i \in [2, \ldots, n-2]$ où

$$\begin{aligned} \tilde{f}_0(n) &= \log_3 \left(\frac{3}{2} n + O(1) \right) \\ \tilde{f}_{i-1}(n) &= \log_3 \left(\frac{3^i}{3^i - 1} n + O(1) \right) \end{aligned}$$

Cette équation a pour unique solution $\tilde{t}_i = \frac{2 \cdot 3^i}{3^{i+1} - 1} n + O(1)$. Un minimum de la fonction $f_i(n)$ s'obtient alors pour $t_i = \lceil \tilde{t}_i \rceil$, ce qui donne

$$f_i(n) \leq f_0(\lceil \tilde{t}_i \rceil) \leq \left\lceil \log_3 \left(\frac{3^{i+1}}{3^{i+1} - 1} n + O(1) \right) \right\rceil$$

et complète la preuve par récurrence.

Nous observons alors que de façon asymptotique, c'est-à-dire pour $i \to n \to \infty$, nous obtenons $t_\infty \sim \lceil 2n/3 \rceil$ et $f_\infty(n) \sim \lceil \log_3 n \rceil$. De cette observation, nous pouvons tirer la conclusion suivante : en laissant à l'algorithme la possibilité de se rappeler récursivement, la valeur de t peut être abaissée jusqu'à $\lceil 2n/3 \rceil$ afin d'obtenir une solution de cardinal $\lceil \log_3 n \rceil$. Cette affirmation peut désormais être vérifiée de la manière suivante. Considérons que $t = \lceil 2n/3 \rceil$ avec n la taille de l'ensemble $\mathcal{I}$ à chaque appel de l'algorithme. Si l'algorithme s'arrête sur un appel de la procédure Colorer-Cliques++ après $i \geq 0$ appels récursifs, alors le cardinal de la partition retournée est borné par

$$\left\lceil \log_3 \frac{3 \left\lceil \frac{2}{3} \left(\frac{1}{3}\right)^i n \right\rceil - 3}{2} \right\rceil \leq \left\lceil \log_3 \left(\frac{3}{2} \frac{2}{3} \left(\frac{1}{3}\right)^i n \right) \right\rceil \leq \left\lceil \log_3 \left(\frac{1}{3^i} \, n \right) \right\rceil \leq \lceil \log_3 n \rceil$$

D'un autre côté, le nombre d'appels récursifs ne peut excéder $\lceil \log_3 n \rceil - 1$ puisque plus de deux tiers des intervalles sont supprimés de $\mathcal{I}$ à chaque appel et que l'algorithme s'arrête lorsque $|\mathcal{I}| \leq 3$. Ainsi, dans le cas où l'algorithme s'arrête après le premier test, la partition retournée comprend les $\lceil \log_3 n \rceil - 1$ stables extraits à chaque appel récursif de l'algorithme, plus l'ensemble formé des trois intervalles (ou moins) restant dans $\mathcal{I}$. Notons que chaque appel ne consommant qu'un temps $O(n)$ lors du calcul du stable maximum, l'algorithme tout entier s'exécute en temps $O(n \log n)$ et espace $O(n)$.

Proposition 5.4 (Borne supérieure raffinée) *Tout graphe d'intervalles (resp. d'arcs) admet une partition en moins de $\lceil \log_3 n \rceil$ (resp. $\lceil \log_3 n \rceil + 1$) sous-graphes d'intervalles propres, pour $n > 1$. De plus, cette partition s'obtient en temps $O(n \log n) + m$ et espace linéaire.*

Remarque. L'algorithme Colorer-Cliques-Récursif peut être modifié de façon à extraire le stable formé par les feuilles du plus grand sous-graphe induit $K_{1,t}$, et non pas un stable maximum (dont la taille est nécessairement supérieure ou égale à ce dernier). Cette modification n'améliore pas la qualité de la partition retournée dans le pire des cas, alors que le temps d'exécution de l'algorithme est de l'ordre de $O(n^2 + m)$. Toutefois, celle-ci peut s'avérer intéressante en pratique.

Pour terminer, nous donnons un graphe d'intervalles pour lequel la borne $\lceil \log_3 n \rceil$ est atteinte de façon asymptotique. Prenons $\lceil 2n/3 \rceil - 1$ intervalles de longueur unitaire induisant un stable et recouvrons-les par un intervalle I. Nous avons là une représentation du graphe $K_{1,\lceil 2n/3 \rceil - 1}$. Maintenant, disposons les intervalles restants, eux aussi de longueur unitaire, de sorte que le plus grand stable dans le graphe reste de taille $\lceil 2n/3 \rceil$. Imaginons par exemple que ces derniers induisent un stable de taille $\lfloor n/3 \rfloor$ (voir la figure 5.4 ci-dessous).

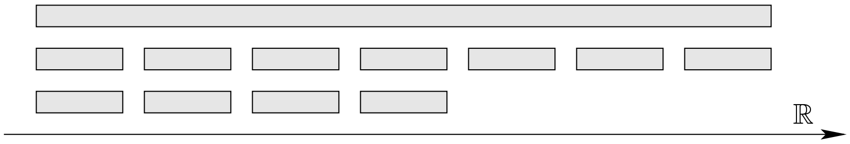


Fig. 5.4 – Un exemple de construction avec $n = 12$.

Alors que ce graphe admet une partition en deux sous-graphes d'intervalles propres, l'algorithme Colorer-Cliques-Récursif retourne une partition de cardinal $\lceil \log_3 \frac{3\lceil 2n/3 \rceil - 3}{2} \rceil \leq \lceil \log_3 n \rceil$, avec l'égalité lorsque n tend vers l'infini.

Questions ouvertes et extensions

Les bornes supérieures du lemme 5.3 et de la proposition 5.4 sont les meilleures possibles *à un près* pour les graphes d'intervalles. Les premières plages de valeurs pour lesquelles les bornes inférieures et supérieures diffèrent sont $t = 8, 9$ et $n = 10, 11, 12$. Un récapitulatif des bornes que nous avons obtenues concernant le problème de la partition en sous-graphes d'intervalles propres est donné sur la figure 5.5.

Resserrer plus encore ces bornes semble être une tâche bien ardue. Pour l'aborder, répondre à la question suivante semble décisif. *Existe-t-il des graphes d'intervalles sans $K_{1,8}$*

graphes d'intervalles	borne inférieure	borne supérieure
sans $K_{1,3}$	1	1
sans $K_{1,4}$	2	2
sans $K_{1,5}$	2	2
sans $K_{1,6}$	2	2
sans $K_{1,7}$	2	2
sans $K_{1,8}$	2	3
sans $K_{1,9}$	2	3
sans $K_{1,10}$	3	3
sans $K_{1,t}$	$\lfloor \log_3(t-1) \rfloor + 1$	$\lceil \log_3((t-1)/2) \rceil + 1$
à n sommets	$\lfloor \log_3(2n+1) \rfloor$	$\lceil \log_3 n \rceil$

FIG. 5.5 – Un récapitulatif des bornes inférieures et supérieures.

ou $K_{1,9}$ n'admettant aucune partition en moins de trois sous-graphes d'intervalles propres ? Ou bien, les graphes d'intervalles sans $K_{1,8}$ ou $K_{1,9}$ admettent-ils tous une partition en deux sous-graphes d'intervalles propres ? Afin de mesurer la difficulté du problème, nous invitons le lecteur à étudier la figure 5.6 où est représenté un graphe d'intervalles sans $K_{1,8}$, mais possédant une copie induite de $K_{1,7}$. Clairement, ce graphe n'est pas un graphe d'intervalles propres. D'autre part, il est assez facile de vérifier que celui-ci admet une partition en trois sous-graphes d'intervalles propres. Toutefois, ce graphe admet-il une partition en deux sous-graphes d'intervalles propres ? La réponse est oui : une partition s'obtient en plaçant les intervalles clairs dans un premier ensemble et les intervalles foncés dans un second.

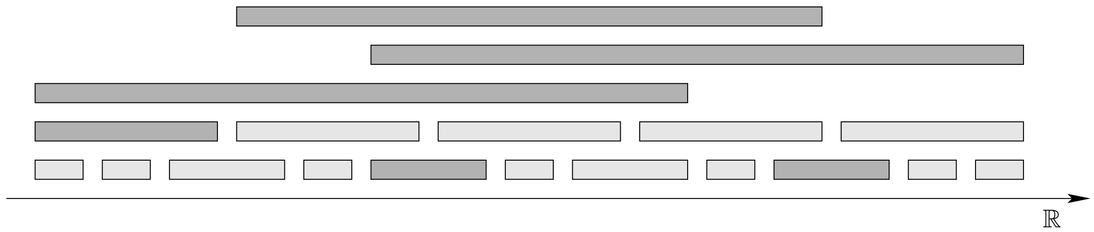


FIG. 5.6 – Une instance sans $K_{1,8}$ problématique.

Plus généralement se pose le problème de la caractérisation des graphes d'intervalles en k sous-graphes d'intervalles propres. Nous savons maintenant que tout graphe d'intervalles qui contient H_k comme sous-graphe induit n'admet pas de partition en moins de k sous-graphes d'intervalles propres. D'un autre côté, *tout graphe d'intervalles ne contenant pas H_k comme sous-graphe induit admet-il une partition en moins de k sous-graphes d'intervalles propres ?* Les questions de complexité sont elles aussi à mettre en relation avec le problème de la caractérisation. En effet, n'ayant aucune caractérisation précise, déterminer une partition *minimum* d'un graphe d'intervalles en sous-graphes d'intervalles propres semble être un problème difficile (l'exemple de la figure 5.6 l'illustre une nouvelle fois). Toutefois, les discussions précédentes nous permettent d'établir quelques résultats partiels concernant ces questions.

Corollaire 5.5 (Complexité) *Une partition minimum d'un graphe d'intervalles sans $K_{1,7}$ en sous-graphes d'intervalles propres peut être déterminée en temps et espace linéaire.*

Preuve. Le lemme de partition logarithmique fournit un algorithme efficace pour résoudre le problème restreint aux graphes d'intervalles sans $K_{1,7}$. Tout d'abord, testons en temps et espace linéaire si le graphe est un graphe d'intervalles propres [31]. Si ce n'est pas le cas, alors l'algorithme COLORER-CLIQUES++ (avec $t = 7$) permet de trouver en temps et espace linéaire une partition en deux sous-graphes d'intervalles propres. □

Corollaire 5.6 (Complexité) *Une partition minimum d'un graphe d'intervalles (resp. d'arcs) en sous-graphes d'intervalles propres peut être approchée en temps $O(n \log n + m)$ et espace linéaire avec un ratio $\lceil \log_3 n \rceil / 2$ (resp. $(\lceil \log_3 n \rceil + 1)/2$) dans le pire des cas.*

Preuve. L'algorithme COLORER-CLIQUES-RÉCURSIF est à la base de l'approximation. Si le graphe d'intervalles en entrée n'est pas un graphe d'intervalles propres [31], alors celui-ci permet de trouver en temps $O(n \log n + m)$ et espace linéaire une partition de cardinal $\lceil \log_3 n \rceil$. Or, dans ce cas, la partition optimale est de cardinal au moins deux. Dans le cas des graphes d'arcs, il nous faut extraire auparavant une clique du graphe, comme cela est fait dans la preuve de la proposition 5.1, pour appliquer la méthode en question (d'où le ratio $(\lceil \log_3 n \rceil + 1)/2$ dans le pire des cas). □

Remarque. L'exemple de la figure 5.4 montre un graphe d'intervalles pour lequel la méthode en question atteint le ratio d'approximation $(\lceil \log_3 n \rceil + 1)/2$.

En résumé, voici les questions de complexité qui demeurent ouvertes à ce jour. *Le problème de la partition d'un graphe d'intervalles (même sans $K_{1,8}$) en un minimum de sous-graphes d'intervalles propres est-il polynomial ? Existe-t-il un algorithme énumératif quasi-polynomial en temps $O(n^{\lceil \log_3 n \rceil})$ pour ce problème ? Existe-t-il un algorithme polynomial d'approximation ayant un ratio constant dans le pire des cas pour ce problème ?*

Comme cela a été fait pour les graphes d'arcs, nous espérons pouvoir étendre le théorème 5.1 aux *graphes triangulés* et aux *graphes de tolérances*. Bien que le graphe d'intersection des cliques forme un arbre pour les graphes triangulés (et non plus une chaîne comme pour les graphes d'intervalles), nous pensons que l'algorithme COLORER-CLIQUES peut être adapté pour colorier des cliques possédant une telle structure. Pour les graphes de tolérances, une idée serait d'effectuer une partition en sous-graphes d'intervalles propres en supprimant les tolérances. Le problème est que, par le jeu des tolérances, toute représentation d'intervalles se chevauchant deux-à-deux n'induit pas forcément une clique. Prenons par exemple quatre intervalles $I_a = I_b = I_c = I_d = [0, 1]$ avec les tolérances $t_a = 0$ et $t_b = t_c = t_d = 1$. Cette représentation induit clairement le graphe $K_{1,3}$, alors que les quatre intervalles se chevauchent deux-à-deux. Nous pensons toutefois que tout ensemble de sommets dont les intervalles se chevauchent deux-à-deux admet une partition en $O(1)$ sous-graphes d'intervalles propres (plus précisément en $O(1)$ stables ou cliques). Enfin, nous conclurons par une question relative aux limites de l'extension du théorème 5.1 : *existe-t-il un quelconque graphe n'admettant aucune partition en moins de $n^{1-\epsilon}$ sous-graphes d'intervalles propres ($\epsilon \geq 0$) ?*

5.2 Une application à la planification de personnel

Le problème de la *planification de personnel* est aujourd'hui au cœur de l'administration des entreprises. Le besoin accru de productivité, couplé à une réglementation du travail toujours plus dense, en fait un des problèmes phares de la recherche opérationnelle depuis plus de quarante ans. Le problème que nous étudions ici consiste de façon schématique en l'*affectation de tâches (fixes dans le temps) à des employés sous la forme de vacations*. L'employé ne pouvant effectuer deux tâches à la fois, les tâches de la vacation affectée à celui-ci ne doivent pas se chevaucher dans le temps. Voici une formulation épurée d'une problématique que nous avons rencontrée lors de notre séjour au sein de la firme Prologia du Groupe Air Liquide, spécialisée dans la résolution de problèmes de planification de personnel [7]. Soit $\{T_i\}_{i=1,\ldots,n}$ un ensemble de tâches ayant chacune une date de début d_i et une date de fin f_i. La *réglementation* impose que chaque employé n'exécute pas plus de k tâches. Les tâches allouées à un employé devant être deux-à-deux *disjointes*, le but est de réaliser une affectation optimale des tâches aux employées vis-à-vis des objectifs suivants : à un *premier niveau*, réduire le nombre d'employés à mobiliser, les employés travaillant souvent en sous-effectif (*productivité*), puis à un *second niveau*, équilibrer l'affectation (*social*) et prévenir au mieux les modifications futures du planning (*robustesse*).

Les tâches étant de simples intervalles sur l'axe du temps, ce problème peut être formulé comme un problème de *coloration d'intervalles tel que chaque couleur marque au plus k sommets*. Lorsque le planning est *cyclique*, nous obtenons le même problème de coloration mais pour des *arcs circulaires*. Dans ce modèle, les critères d'optimisation deviennent respectivement : (P) minimiser le nombre de couleurs utilisées, (S) équilibrer le nombre d'intervalles (ou d'arcs) de chaque couleur, et (R) maximiser la taille du plus petit espace libre entre deux intervalles (ou arcs) consécutifs d'une même couleur. En recherchant à maximiser le plus petit temps d'enchaînement entre deux tâches consécutives d'une même vacation, le critère R permet de prévenir les chevauchements entre les tâches d'une vacation lorsque celles-ci doivent être retardées ou avancées dans le temps. Si par exemple, la valeur de ce plus petit temps d'enchaînement est de vingt minutes, alors une réorganisation du planning ne sera nécessaire qu'en cas d'un retard cumulé supérieur à vingt minutes, et ce quelles que soient les tâches affectées par ce retard. Nous dirons alors qu'une solution au problème est *P-optimale* si elle est optimale pour le critère P, et *SR-optimale* si elle est optimale pour les critères *S et R*. Une solution sera dite *P/SR-optimale* si celle-ci est SR-optimale parmi toutes les solutions P-optimales.

La formulation de ce problème rappelle bien entendu celle du problème d'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles ou d'arcs, les critères S et R en moins. Par conséquent, même en ne considérant que le critère P à optimiser, le problème reste difficile à résoudre pour $k \geq 4$ fixé [14]. À moins que $\mathcal{P} = \mathcal{NP}$, la difficulté inhérente au problème nous condamne donc à l'élaboration d'heuristiques efficaces, permettant de trouver seulement de "bonnes" solutions. Nous présenterons dans cette section deux algorithmes linéaires d'approximation pour ce problème fondamental de planification de personnel, noté Planif. L'entrée de nos algorithmes d'approximation sera composée d'un ensemble $\mathcal{I}$ de n intervalles et de l'entier k. La qualité de nos algorithmes, en relation avec le critère P, sera mesurée par leur *ratio dans le pire des cas* défini comme $\sup_G\{\,|\mathcal{S}|/\chi(\mathcal{I},k)\,\}$, où $\mathcal{S}$ est une partition de l'ensemble $\mathcal{I}$ d'intervalles en stables de taille au plus k retournée par l'algorithme. Le premier algorithme, classique, atteint un ratio constant dans le pire des cas pour le seul critère P. Malheureusement, une telle approche n'offre aucune garantie sur la qualité de la solution obtenue vis-à-vis des critères S et R. D'un autre côté, le problème Planif s'avère être soluble par un simple algorithme glouton lorsque les intervalles sont propres. Une idée peut donc être d'effectuer une première partition de l'ensemble des intervalles en sous-ensembles d'intervalles propres – ou du moins tel que chacun de ces sous-ensembles induise un graphe d'intervalles propres – puis de résoudre de manière optimale le problème sur chacun de ces sous-ensembles à l'aide de l'algorithme glouton. Bien entendu, la qualité d'une telle optimisation “locale” dépend fortement du résultat de la partition initiale de l'ensemble d'intervalles. En nous appuyant sur les résultats de la section précédente, nous montrons que ce type d'approche permet d'obtenir un ratio d'approximation *constant en situation réelle* (*i.e.* sous certaines conditions), tout en offrant des garanties sur la qualité des solutions obtenues vis-à-vis des critères S et R.

Une approximation classique

Avant de donner une brève description de l'algorithme d'approximation, nous rappelons deux propositions démontrées dans les chapitres précédents sur lesquelles il repose :

(1) une coloration minimum de $\mathcal{I}$ où le nombre $\vartheta(\mathcal{I})$ de stable de taille un est le plus petit que possible peut être déterminée en temps $O(n \log n)$ et espace $O(n)$;

(2) si $\mathcal{I}$ admet une partition en stables où chaque stable est de taille au moins k, alors une partition optimale de $\mathcal{I}$ en $\lceil n/k \rceil$ stables de taille au plus k peut être déterminée en temps $O(n \log n)$ et espace $O(n)$.

La première assertion découle de l'analyse de l'algorithme 2-Ordo-Intervalles (voir p. 27) et la seconde est une réécriture du corollaire 4.10 (voir p. 78).

Algorithme 2-Approx-Planif ;
Entrée : un ensemble $\mathcal{I}$ de n intervalles, un entier k ;
Sortie : une solution $\mathcal{S}$ au problème Planif pour $\mathcal{I}$;
Début ;
  calculer un coloration minimum $\mathcal{C} = \{S_1, \ldots, S_{\chi(G)}\}$ de $\mathcal{I}$ avec $\vartheta(\mathcal{I})$ minimum ;
  $\mathcal{S} \leftarrow \emptyset$;
  pour chaque $S_j \in \mathcal{C}$ **faire**
    si $|S_j| < k$ **alors** $\mathcal{C} \leftarrow \mathcal{C} \setminus \{S_j\}$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_j\}$;
  calculer une partition optimale $\mathcal{S}_k$ de $\mathcal{C}$ en stables de taille au plus k ;
  retourner $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_k$;
Fin ;

Proposition 5.5 *L'algorithme* 2-Approx-Planif *atteint en temps $O(n \log n)$ et espace $O(n)$ le ratio asymptotique $(2k-2)/k$ dans le pire des cas pour le critère P. Ce ratio est, en outre, le meilleur possible.*

Preuve. La validité et la complexité de l'algorithme découlent des assertions (1) et (2) mentionnées plus haut. Quant au ratio dans le pire des cas, nous l'établissons de la façon suivante. Notons $\mathcal{C}_2$ l'ensemble des stables de taille $2, \ldots, k-1$ dans $\mathcal{C}$ et $\mathcal{C}_k$ l'ensemble des stables de taille au moins k. La solution $\mathcal{S}$ retournée par l'algorithme est de cardinal

$$|\mathcal{S}| = |\mathcal{C} \setminus \mathcal{C}_k| + \left\lceil \frac{\sum_{S_j \in \mathcal{C}_k} |S_j|}{k} \right\rceil \tag{5.1}$$

Si $\mathcal{C}_k = \emptyset$ (*i.e.* $\chi(\mathcal{I}, k) = \chi(\mathcal{I})$) ou $\mathcal{C} \setminus \mathcal{C}_k = \emptyset$ (*i.e.* $\chi(\mathcal{I}, k) = \lceil n/k \rceil$), alors $\mathcal{S}$ est P-optimale. Considérons maintenant que $\mathcal{C}_k \neq \emptyset$ et $\mathcal{C} \setminus \mathcal{C}_k \neq \emptyset$. Soit $S_j = \{I_i\}$ un des $\vartheta(\mathcal{I})$ stables de taille un. Puisque $\omega(\mathcal{I}) = \chi(\mathcal{I})$, I_i appartient à toute clique maximum de $\mathcal{I}$. Par conséquent, toute solution optimale comprend au moins $\vartheta(\mathcal{I})$ intervalles non couplés, ce qui implique l'inégalité

$$\chi(\mathcal{I}, k) \geq \left\lceil \frac{n - \vartheta(\mathcal{I})}{k} \right\rceil + \vartheta(\mathcal{I}) \tag{5.2}$$

Comme $\sum_{S_j \in \mathcal{C} \setminus \mathcal{C}_k} |S_j| = \sum_{S_j \in \mathcal{C}_2} |S_j| + \vartheta(\mathcal{I})$ et que les stables de $\mathcal{C}_2$ sont tous de taille au moins deux, nous pouvons réécrire l'inégalité (5.2) comme suit :

$$\frac{\sum_{S_j \in \mathcal{C}_k} |S_j|}{k} \leq \chi(\mathcal{I}, k) - \frac{2}{k} |\mathcal{C}_2| - \vartheta(\mathcal{I}) \tag{5.3}$$

En employant (5.3) puis l'égalité $|\mathcal{C} \setminus \mathcal{C}_k| = |\mathcal{C}_2| + \vartheta(\mathcal{I})$, l'expression (5.1) devient

$$|\mathcal{S}| \leq \left\lceil \chi(\mathcal{I}, k) + \frac{k-2}{k} |\mathcal{C}_2| \right\rceil \tag{5.4}$$

Puisque $|\mathcal{C}_2| \leq \chi(\mathcal{I}) - 1 \leq \chi(\mathcal{I}, k) - 1$, nous obtenons enfin que $|\mathcal{S}| \leq \lceil \frac{2k-2}{k} \chi(\mathcal{I}, k) - \frac{k-2}{k} \rceil$.

Pour conclure, nous montrons que ce ratio est le meilleur possible. Posons $n = kq$ ($q \geq 2$) et définissons l'ensemble $\mathcal{I}$ comme l'union de q intervalles formant une clique et de $n - q$ intervalles disjoints. Voici une façon de colorier $\mathcal{I}$ où $\vartheta(\mathcal{I})$ est minimisé : répartissons les intervalles de la clique dans q stables différents $S_1, \ldots, S_q$, puis plaçons les intervalles disjoints de sorte que $|S_1| = kq - 2(q-1)$ et $|S_2| = \cdots = |S_q| = 2$ (le lecteur notera que $\chi(\mathcal{I}) = n/k = q$). La solution retournée par l'algorithme sera de cardinal

$$|\mathcal{S}| = \left\lceil \frac{(k-2)\chi(\mathcal{I}) + 2}{k} \right\rceil + \chi(\mathcal{I}) - 1 \tag{5.5}$$

alors qu'une solution P-optimale est simplement de cardinal $\chi(\mathcal{I}, k) = \chi(\mathcal{I}) = n/k$. Par substitution dans (5.5), nous obtenons alors que $|\mathcal{S}| = \lceil \frac{2k-2}{k} \chi(\mathcal{I}, k) - \frac{k-2}{k} \rceil$ et le ratio $(2k-2)/k$ est atteint de façon asymptotique. □

Remarque. L'algorithme 2-Approx-Planif retourne une solution P-optimale lorsque $k = 2$ et fournit une $4/3$-approximation dans le pire des cas lorsque $k = 3$.

Pour les graphes d'arcs circulaires, voici une méthode simple qui atteint en temps $O(n \log n)$ et espace $O(n)$ le ratio asymptotique $(3k-2)/k$ dans le pire des cas pour le critère P. L'entrée est composée d'un ensemble $\mathcal{A}$ de n arcs circulaires et de l'entier k. Après avoir choisi un point p du cercle, nous retirons de $\mathcal{A}$ l'ensemble C des arcs qui contiennent le point p. Clairement, les arcs de C induisent une clique et $\mathcal{A}$ peut être considéré comme un ensemble d'intervalles, noté $\mathcal{I}$. Une solution au problème Planif pour $\mathcal{A}$ est alors $\mathcal{S} = \{C\} \cup$ 2-APPROX-PLANIF$(\mathcal{I}, k)$. Comme $|C| \leq \chi(\mathcal{A}, k)$ et $\chi(\mathcal{I}, k) \leq \chi(\mathcal{A}, k)$, nous avons $|\mathcal{S}| \leq \lceil \frac{3k-2}{k} \chi(\mathcal{A}, k) - \frac{k-2}{k} \rceil$.

À présent, prenons deux ensembles d'arcs formant chacun une clique de taille q, et $n-2q$ arcs disjoints. Afin que la représentation induise un graphe d'arcs et non pas seulement un graphe d'intervalles, il faut que les deux cliques ne soient pas disjointes et qu'un arc appartenant à une des deux viennent chevaucher tous les autres arcs de la représentation (en effectuant le tour du cercle). Sur une telle instance, l'algorithme atteint de façon asymptotique le ratio $(3k-3)/k$ si le point p est choisi de manière à ce qu'une des deux cliques soit extraite de la représentation et que la coloration de l'ensemble $\mathcal{I}$ des intervalles restants soit faite comme dans la preuve de la proposition 5.5.

Cette méthode fournit une 2-approximation dans le pire des cas lorsque $k = 2$ et une 7/3-approximation lorsque $k = 3$.

Un algorithme glouton pour le cas des intervalles propres

L'algorithme en question est le même que celui qui nous a permis de résoudre le problème ORDO pour les graphes d'intervalles propres.

Algorithme GLOUTON-PLANIF ;
Entrée : un ensemble $\mathcal{I}$ de n intervalles propres, un entier k ;
Sortie : une solution $\mathcal{S}$ au problème PLANIF pour $\mathcal{I}$;
Début ;
  soit $I_0, \ldots, I_{n-1}$ les intervalles de $\mathcal{I}$ ordonnés selon $<_g$;
  calculer $\omega(\mathcal{I})$;
  $\chi(\mathcal{I}, k) \leftarrow \max\{\omega(\mathcal{I}), \lceil n/k \rceil\}$;
  $S_0 \leftarrow \cdots \leftarrow S_{\chi(\mathcal{I},k)-1} \leftarrow \emptyset$;
  pour i de 0 à $n-1$ **faire**
    $S_{i \bmod \chi(\mathcal{I},k)} \leftarrow S_{i \bmod \chi(\mathcal{I},k)} \cup \{I_i\}$;
  retourner $\mathcal{S} \leftarrow \{S_0, \ldots, S_{\chi(\mathcal{I},k)-1}\}$;
Fin ;

Proposition 5.6 *L'algorithme* GLOUTON-PLANIF *retourne en temps $O(n \log n)$ et espace $O(n)$ une solution P/SR-optimale au problème* PLANIF *pour un ensemble $\mathcal{I}$ de n intervalles propres.*

Preuve. Ordonner les intervalles de $\mathcal{I}$ prend un temps $O(n \log n)$. De là, le calcul d'une clique maximum peut s'effectuer en $O(n)$ temps et espace à l'aide de l'algorithme CLIQUE-INTERVALLES (voir p. 45). Le reste de l'algorithme s'exécute en temps linéaire.

Le caractère P/S-optimal de la solution découle immédiatement de la preuve de la proposition 2.11. À présent, supposons que l'ensemble $S_0, \ldots, S_{\chi(\mathcal{I},k)-1}$ ne soit pas P/R-optimal. Soit $S_0^*, \ldots, S_{\chi(\mathcal{I},k)-1}^*$ une solution P/R-optimale et ℓ^* la longueur du plus petit espace entre deux intervalles consécutifs dans cette solution. Nous rappelons que les intervalles $I_0, \ldots, I_{n-1}$ sont triés selon l'ordre défini par les extrémités gauches croissantes ; l'intervalle de rang t dans le stable S_u^* sera noté $I_{u,t}$. Nous affirmons que pour tout $i = 1, \ldots, n$, l'intervalle $I_i \in S_u^*$ peut être placé au rang $t = \lfloor i/\chi(\mathcal{I}, k) \rfloor$ du stable S_v^* avec $v = i \bmod \chi(\mathcal{I}, k)$, et ce, sans abaisser la longueur ℓ^*. Suite à cela, l'ensemble $S_0^*, \ldots, S_{\chi(\mathcal{I},k)-1}^*$ coïncidera exactement avec l'ensemble $S_0, \ldots, S_{\chi(\mathcal{I},k)-1}$ produit par l'algorithme glouton. Nous pourrons donc en conclure que $\mathcal{S}$ est P/R-optimal.

Afin de prouver cette affirmation, nous utilisons un procédé de construction inductif dont l'étape initiale est réalisée comme suit. Si $I_0 \in S_u^*$ avec $u \neq 0$, alors nous échangeons l'ensemble des intervalles de S_u^* avec ceux de S_0^*. Clairement, ℓ^* reste inchangée (l'espacement entre les intervalles n'est pas modifié) et I_0 est maintenant bien placé. À présent, admettons que les intervalles $I_0, \ldots, I_{i-1}$ soient correctement placés. L'intervalle $I_i \in S_u^*$ doit être transporté dans le stable S_v^*, si $u \neq v$. Deux cas sont alors à distinguer.

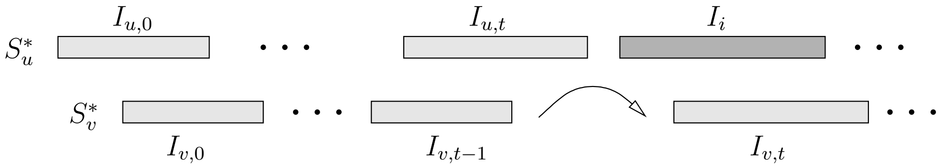


FIG. 5.7 – Le cas (a) : $u < v$.

Cas (a) : $u < v$ (voir la figure 5.7). Dans ce cas, nous avons nécessairement

$$\begin{aligned} S_u^* &= \{I_{u,0}, \ldots, I_{u,t}, I_i, \ldots, I_{u,j}, \ldots\} \\ S_v^* &= \{I_{v,0}, \ldots, I_{v,t-1}, I_{v,t}, \ldots, I_{v,j}, \ldots\} \end{aligned}$$

D'après l'hypothèse d'induction, nous avons que $d(I_{v,t-1}) \leq d(I_{u,t})$ and $g(I_i) \leq g(I_{v,t})$. Puisque $d(I_{u,t}) < g(I_i)$, nous obtenons les inégalités (i) $d(I_{v,t-1}) \leq d(I_{u,t}) < g(I_i) \leq g(I_{v,t})$ qui nous autorisent à redéfinir les stables S_u^* et S_v^* comme suit :

$$\begin{aligned} S_u^* &= \{I_{u,0}, \ldots, I_{u,t}, I_{v,t}, \ldots, I_{v,j}, \ldots\} \\ S_v^* &= \{I_{v,0}, \ldots, I_{v,t-1}, I_i, \ldots, I_{u,j}, \ldots\} \end{aligned}$$

L'espacement entre les intervalles a été modifié : $g(I_i) - d(I_{u,t})$ est devenu $g(I_{v,t}) - d(I_{u,t})$ dans S_u^* et $g(I_{v,t}) - d(I_{v,t-1})$ est devenu $g(I_i) - d(I_{v,t-1})$ dans S_v^*. D'après (i), ces deux nouveaux espaces sont plus grands que le plus petit des deux anciens.

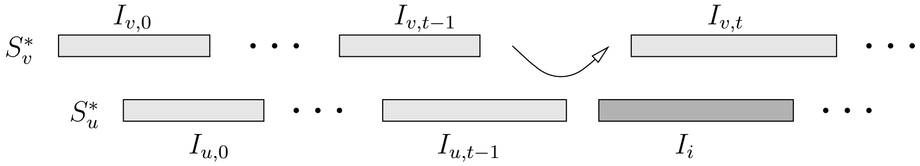


FIG. 5.8 – Le cas (b) : $u > v$.

Cas (b) : $u > v$ (voir la figure 5.8). Ici, nous avons nécessairement

$$\begin{aligned} S_u^* &= \{I_{u,0}, \ldots, I_{u,t-1}, I_i, \ldots, I_{u,j}, \ldots\} \\ S_v^* &= \{I_{v,0}, \ldots, I_{v,t-1}, I_{v,t}, \ldots, I_{v,j}, \ldots\} \end{aligned}$$

Ici l'hypothèse d'induction nous donne les inégalités (ii) $d(I_{v,t-1}) \leq d(I_{u,t-1}) < g(I_i) \leq g(I_{v,t})$, qui nous permettent de redéfinir les stables S_u^* et S_v^* comme suit :

$$\begin{array}{rcl} S_u^* & = & \{I_{u,0}, \ldots, I_{u,t-1}, I_{v,t}, \ldots, I_{v,j}, \ldots\} \\ S_v^* & = & \{I_{v,0}, \ldots, I_{v,t-1}, I_i, \ldots, I_{u,j}, \ldots\} \end{array}$$

D'après (ii), les espaces dans S_u^* et S_v^* n'ont pu que s'agrandir après cette opération.

Suite à cette analyse de cas, la preuve de notre affirmation est complète et la proposition démontrée. □

Une approximation logarithmique

Voici un nouvel algorithme d'approximation pour le problème PLANIF, basé sur les résultats de la section précédente. Par abus de langage, nous appellerons ici *sous-ensemble propre*, tout sous-ensemble d'intervalles qui *induit* un graphe d'intervalles propres.

Algorithme log-APPROX-PLANIF
Entrée : un ensemble $\mathcal{I}$ de n intervalles, un entier k ;
Sortie : une solution $\mathcal{S}$ au problème PLANIF pour $\mathcal{I}$;
Début ;
  ordonner $\mathcal{I}$ selon $<_g$;
  $\mathcal{C} \leftarrow$ COLORER-CLIQUES-RÉCURSIF($\mathcal{I}$), $\mathcal{S} \leftarrow \emptyset$;
  pour chaque sous-ensemble propre $\mathcal{C}_r \in \mathcal{C}$ **faire**
    ordonner les intervalles de $\mathcal{C}_r$ selon un ordre d'intervalles propres ;
    $\mathcal{S} \leftarrow \mathcal{S} \cup$ GLOUTON-PLANIF($\mathcal{C}_r, k$) ;
  retourner $\mathcal{S}$;
Fin ;

Proposition 5.7 *L'algorithme* log-APPROX-PLANIF *atteint en temps $O(f(n) \cdot n \log n)$ et espace $O(n)$ le ratio absolu* $\min\{k, f(n)\}$ *dans le pire des cas pour le critère P, avec $f(n) = \lceil \log_3 n \rceil$. Ce ratio est, en outre, le meilleur possible.*

Preuve. La validité et la complexité de l'algorithme découle des propositions 5.4 et 5.6. Une fois $\mathcal{I}$ ordonné selon $<_g$, l'algorithme COLORER-CLIQUES-RÉCURSIF s'exécute en temps $O(n \log n)$ et espace $O(n)$. De même, une fois les intervalles de $\mathcal{C}_r$ ordonnés selon un ordre d'intervalles propres, l'algorithme GLOUTON-PLANIF s'exécute en temps et espace $O(|\mathcal{C}_r|)$.

Malheureusement, déterminer un des ordres d'intervalles propres induit par les intervalles de $\mathcal{C}_r$ n'est pas si simple. Nous montrons comment réaliser cette opération en temps $O(|\mathcal{C}_r| \log |\mathcal{C}_r|)$ en utilisant le fait que l'ensemble $\mathcal{C}_r$ est soit un stable, soit un ensemble de cliques presque disjointes. Quand $\mathcal{C}_r$ est un stable, il suffit de conserver les intervalles dans l'ordre $<_g$, lorsque celui-ci est extrait par l'algorithme COLORER-CLIQUES-RÉCURSIF. Quand $\mathcal{C}_r$ est un ensemble de cliques, il faut tout d'abord conserver les cliques dans l'ordre dans lequel elles ont été extraites par l'algorithme COLORER-CLIQUES++. Soit $\mathcal{C}_r = \{C_1, \ldots, C_q\}$ l'ensemble en question avec $C_1 < \cdots < C_q$. D'après l'algorithme COLORER-CLIQUES++, nous avons que $\bigcap_{\forall j \ impair} C_j \cup C_{j+1} = \emptyset$ (c'est précisément pour cette raison que $\mathcal{C}_r$ induit un graphe d'intervalles sans $K_{1,3}$). Pour chacun de ces ensembles $C_j \cup C_{j+1}$, notons respectivement g^* la plus petite extrémité gauche et d^* la plus grande extrémité droite d'un intervalle de $C_j \cup C_{j+1}$. Puisque les couples $C_j \cup C_{j+1}$ sont disjoints, il est possible d'étendre les extrémités gauches des intervalles de C_j jusqu'au point g^* et les extrémités droites des intervalles de C_{j+1} jusqu'au point d^*. La représentation que l'on obtient est une représentation de $\mathcal{C}_r$ par intervalles propres et l'ordre défini par les nouvelles extrémités des intervalles un ordre d'intervalles. Nous constatons que cet ordre s'obtient simplement en ordonnant les intervalles de chaque clique C_j selon $<_d$ et les intervalles de chaque clique C_{j+1} selon $<_g$ (voir la figure 5.9).

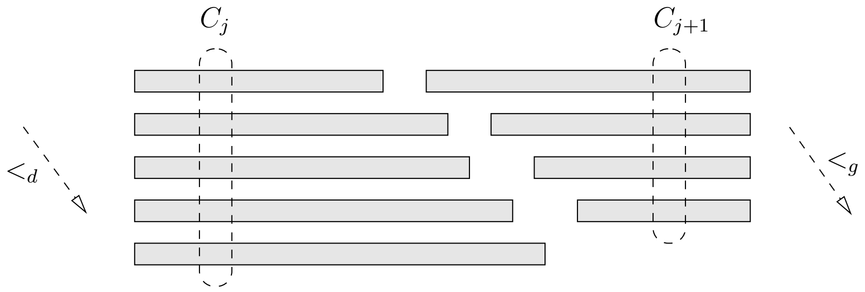


FIG. 5.9 – Un ordre d'intervalles propres sur $C_j \cup C_{j+1}$.

Par conséquent, ordonner les intervalles de $\mathcal{C}_r = \{C_1, \ldots, C_q\}$ (avec $C_1 < \cdots < C_q$) selon un ordre d'intervalles propres peut se faire en temps $O(|\mathcal{C}_r| \log |\mathcal{C}_r|)$ comme suit : trier les intervalles de $\bigcup_{\forall j \ impair} C_j$ selon $<_d$ et les intervalles de $\bigcup_{\forall j \ impair} C_{j+1}$ selon $<_g$, puis fusionner les deux ensembles. En définitive, le temps total consommé par l'algorithme est borné par

$$O(n \log n) + \sum_{r=1}^{f(n)} O(|\mathcal{C}_r| \log |\mathcal{C}_r|) = O(n \log n) + \sum_{r=1}^{f(n)} O(n \log n) = O(f(n) \cdot n \log n)$$

Pour conclure, le ratio de l'algorithme dans le pire des cas découle des deux inégalités suivantes :

$$\begin{aligned} |\mathcal{S}| &= \sum_{r=1}^{f(n)} \chi(\mathcal{C}_r, k) \leq n \leq k \cdot \chi(\mathcal{I}, k) \\ |\mathcal{S}| &= \sum_{r=1}^{f(n)} \chi(\mathcal{C}_r, k) \leq \sum_{r=1}^{f(n)} \chi(\mathcal{I}, k) \leq f(n) \cdot \chi(\mathcal{I}, k) \end{aligned}$$

Lorsque $k \geq \lceil 2n/3 \rceil + 1$, ce ratio est atteint pour la même instance que celle pour laquelle l'algorithme COLORER-CLIQUES-RÉCURSIF se comporte mal (voir la figure 5.4). En effet, alors qu'une partition optimale est de cardinal trois, l'algorithme log-APPROX-PLANIF retourne une solution de cardinal $3\lceil \log_3 n \rceil$, lorsque n tend vers l'infini (la partition $\mathcal{C}$ retournée par COLORER-CLIQUES-RÉCURSIF est de cardinal $\lceil \log_3 n \rceil$ et chaque sous-ensemble propre contient une clique de taille trois). □

Remarque. Ici la partition de $\mathcal{I}$ en sous-ensemble propres est effectuée à l'aide de l'algorithme COLORER-CLIQUES-RÉCURSIF. Il est bien entendu possible d'utiliser à la place la première méthode de partition décrite dans la preuve de la proposition 5.1. Dans ce cas, le ratio dans le pire des cas devient $\min\{k, f(n)\}$ avec $f(n) = 2\lceil \log_3((n+1)/2)\rceil + 1$. Ce ratio est, en outre, le meilleur possible de façon asymptotique. En effet, lorsque $k = f(n)$, le ratio $2\lfloor \log_3((n+1)/2)\rfloor + 1$ est atteint pour la même instance que celle avec laquelle cette même méthode de partition se comporte mal (voir la figure 5.1). En effet, après avoir effectué une partition en $f(n)/2$ sous-ensembles propres (par application successive des lemmes 5.2 et 5.1), la procédure GLOUTON-PLANIF retourne une partition de chaque sous-ensemble propre en $2g(n) + 1$ stables (chacun de taille au plus deux). D'un autre côté, une solution bien meilleure peut être obtenue si l'on commence par retirer de $\mathcal{I}$ la clique de taille $f(n)$ induit par les intervalles contenant le point d'abscisse $1 + \epsilon$ (avec $\epsilon > 0$). En définitive, nous avons pour cette instance le ratio asymptotique

$$\frac{|\mathcal{S}|}{\chi(\mathcal{I}, k)} \geq \frac{(f(n)/2)(2g(n)+1)}{f(n)+g(n)-1} = f(n) \cdot \frac{n - f(n)/2}{n + f^2(n) - 2f(n)} \sim f(n) = k$$

Comme pour l'algorithme 2-APPROX-PLANIF, le cas des arcs circulaires peut être traité par l'algorithme log-APPROX-PLANIF après avoir pris soin de retirer les arcs contenant un point p du cercle. La complexité de l'algorithme reste alors inchangée et son ratio d'approximation dans le pire des cas pour le critère P devient $\min\{k, f(n) + 1\}$.

Performances en situation réelle

Au vu de ce résultat, l'algorithme log-APPROX-PLANIF ne semble pas pouvoir rivaliser avec l'algorithme 2-APPROX-PLANIF. Toutefois, la proposition 5.7 nous donne une garantie sur la qualité de la solution vis-à-vis du critère P dans les cas les plus extrêmes. Or, le type d'instances pour lesquelles l'algorithme atteint le ratio $\lceil \log_3 n\rceil$ n'est jamais rencontré en pratique. C'est pourquoi nous allons maintenant discuter de la performance des algorithmes log-APPROX-PLANIF et 2-APPROX-PLANIF *en situation réelle*.

Dans une journée de travail telle que nous en avons rencontrées lors de la planification de personnel d'aérogares [7], la durée des tâches varie du quart d'heure pour les plus courtes à onze heures pour les plus longues (qui est le maximum autorisé par la loi). Dans le pire des cas, nous pourrions donc imaginer une tâche de onze heures recouvrant 45 tâches d'un quart d'heure : d'après le lemme 5.3 de partition logarithmique, le graphe d'intervalles induit par l'ensemble des tâches admettrait d'ores et déjà une partition en moins de quatre sous-ensembles propres. Ce type d'exemple est encore loin de la réalité, puisque nous avons pu constater que la plus petite valeur t pour laquelle le graphe d'intervalles induit par les tâches est sans $K_{1,t}$ est petite, *généralement inférieure à sept*. Cela est en partie dû au fait que les tâches longues (de huit à onze heures) sont directement affectées à des employés sous la forme de vacation particulière ne comportant aucune autre tâche. Ce traitement particulier est demandé afin d'empêcher un trop grand déséquilibre dans les plannings (ces tâches longues empêchent la prise de pause et leurs durées sont, à elles seules, supérieures à la durée moyenne de travail journalier d'un employé). Suite à cette remarque, nous pouvons conclure qu'en situation réelle $f(n) = 2$ (d'après le lemme de partition logarithmique).

Un autre intérêt de l'algorithme log-Approx-Planif réside dans son comportement vis-à-vis des critères S et R. Après avoir effectué une partition des intervalles en sous-ensembles propres (*i.e.* qui induisent des sous-graphes d'intervalles propres), l'utilisation de l'algorithme Glouton-Planif nous permet d'obtenir une solution *P/S-optimale* pour chaque sous-ensemble propre, et même *P/R-optimale* si les intervalles du sous-ensemble sont effectivement propres. Lorsqu'un sous-ensemble propre contient des intervalles qui ne sont pas propres, nous dirons que la solution obtenue par le glouton est *quasi P/SR-optimale*. En effet, une telle solution offre une borne inférieure pour le critère R qui n'est pas sans intérêt. De plus, lorsque des retards se produisent et forcent à revoir le planning en temps réel, il n'est pas toujours nécessaire de relancer une planification complète sur les tâches restant à effectuer. Admettons qu'à la suite d'un retard une tâche en chevauche une autre dans une vacation provenant de l'ensemble $\mathcal{C}_r$, et que ce dernier induise toujours un graphe d'intervalles sans $K_{1,3}$. Afin de rendre le planning à nouveau valide, nous n'avons qu'à effectuer une nouvelle partition de l'ensemble $\mathcal{C}_r$ à l'aide de l'algorithme Glouton-Planif, très efficace en temps. Il est même possible de ne toucher que les tâches ayant une date de début supérieure à l'ancienne date de début de l'intervalle retardé, puisque l'ordre des intervalles ayant une date de début inférieure à celle-ci reste inchangé.

Corollaire 5.7 *En situation réelle (i.e. le graphe d'intervalles induit par les tâches est sans $K_{1,7}$), l'algorithme* log-Approx-Planif *atteint en temps $O(n \log n)$ et espace $O(n)$ le ratio absolu* 2 *dans le pire des cas pour le critère P et garantit l'obtention de solutions quasi P/SR-optimales dans deux sous-problèmes distincts.*

D'un autre côté, les valeurs de k utilisées en pratique sont petites, *généralement inférieures à cinq*. Par conséquent, le ratio d'approximation de l'algorithme 2-Approx-Planif restera inférieur à $\frac{8}{5} = 1.6$ en situation réelle.

Corollaire 5.8 *En situation réelle (i.e. le paramètre k est inférieur ou égal à cinq), l'algorithme* 2-Approx-Planif *atteint en temps $O(n \log n)$ et espace $O(n)$ le ratio asymptotique* $8/5$ *dans le pire des cas pour le critère P.*

En conclusion, les solutions retournées par l'algorithme log-Approx-Planif semblent proposer un bon compromis entre productivité, social et robustesse. Toutefois, l'algorithme 2-Approx-Planif sera préféré lorsque le critère de productivité se veut particulièrement privilégié.

5.3 Quelques problèmes connexes

Dans cette dernière section, nous abordons quelques problèmes en relation avec le problème de la partition d'un graphe d'intervalles en sous-graphes d'intervalles propres. À notre connaissance, ces problèmes n'ont été que très partiellement étudiés et plusieurs questions demeurent ouvertes à leur sujet.

5.3.1 La plus petite partition en stables ou cliques

L'étude que nous avons menée dans la première section du chapitre a fait apparaître l'importance des structures de clique et de stable dans les algorithmes de partition en sous-graphes d'intervalles propres. Par ailleurs, nous savons que toute partition en stables ou cliques induit trivialement une partition en sous-graphes d'intervalles propres. Cette remarque a, par exemple, son importance dans le cas des graphes scindés ou à seuil.

Proposition 5.8 *Le problème de la partition minimum en sous-graphes d'intervalles propres peut être résolu en temps et espace linéaire pour les graphes scindés (ou à seuil).*

Preuve. Lorsqu'un graphe scindé n'est pas un graphe d'intervalles propres, il admet une partition en deux sous-graphes d'intervalles propres (puisque celui-ci est composé d'un stable et d'une clique). Tester si un graphe est un graphe d'intervalles propres ne prend qu'un temps et espace linéaire [31], de même que déterminer la partition d'un graphe scindé en un stable et une clique (cf. [75, p. 154]). □

Remarque. Nous verrons par la suite (proposition 5.20, p. 127) que tester si un graphe à seuil est un graphe d'intervalles propres peut se faire directement, sans avoir à utiliser un algorithme de reconnaissance pour les graphes d'intervalles propres.

Comme nous l'avons fait pour le problème de la partition en sous-graphes d'intervalles propres, nous donnons des bornes sur le cardinal d'une partition minimum d'un graphe d'intervalles en stables ou cliques. Nous rappelons que $\chi(G)$ (resp. $\kappa(G)$) dénote le cardinal d'une partition minimum du graphe G en stables (resp. cliques). De même, $\omega(G)$ (resp. $\alpha(G)$) dénote la taille de la plus grande clique (resp. du plus grand stable) de G.

Proposition 5.9 (Borne supérieure) *Tout graphe parfait admet une partition en moins de $\lceil n/2 \rceil$ stables, ou bien une partition en moins de $\lceil n/2 \rceil$ cliques. De plus, cette partition peut être calculée en temps et espace polynomial.*

Preuve. Nous rappelons que si un graphe G est parfait, alors $\chi(G) = \omega(G)$ et $\kappa(G) = \alpha(G)$. Supposons que $\chi(G) > \lceil n/2 \rceil$. Puisque $\chi(G) = \omega(G)$, cela implique aussitôt l'existence d'une clique de taille strictement supérieure à $\lceil n/2 \rceil$. Le nombre de sommets restants étant strictement inférieur à $\lfloor n/2 \rfloor$, le plus grand stable ne peut être que de taille inférieure ou égale à $\lfloor n/2 \rfloor$ (au mieux un sommet de la clique et les $\lfloor n/2 \rfloor$ sommets). Ainsi, nous avons $\kappa(G) = \alpha(G) \leq \lfloor n/2 \rfloor$. Par conséquent, une partition d'un graphe parfait en moins de $\lceil n/2 \rceil$ stables ou cliques s'obtient en temps et espace polynomial de la façon suivante : calculer une partition minimum en stables et une partition minimum en cliques [79], puis garder la plus petite des deux. □

Remarque. De manière symétrique, la preuve peut être établie en montrant que si $\kappa(G) > \lceil n/2 \rceil$, alors $\chi(G) \leq \lfloor n/2 \rfloor$.

Corollaire 5.9 (Borne supérieure) *Tout graphe d'intervalles possède une partition en moins de $\lceil n/2 \rceil$ stables ou cliques. De plus, cette partition peut être calculée en temps et espace linéaire.*

Voici maintenant une classe de graphes pour lesquels la borne $\min\{\chi(G), \kappa(G)\}$ est la meilleure possible : les graphes composés de ℓ cliques disjointes $C_1, \ldots, C_\ell$ de taille k. Nous noterons ce graphe ℓK_k (avec $k, \ell \geq 1$). Après avoir numéroté de 1 à k les sommets de chaque clique, nous définissons les k stables $S_1, \ldots, S_k$ (chacun de taille ℓ) tel que le stable S_i contienne le sommet portant le numéro i dans chacune des ℓ cliques (voir la figure 5.10). Il est facile de voir que $\omega(\ell K_k) = \chi(\ell K_k) = k$ et $\alpha(\ell K_k) = \kappa(\ell K_k) = \ell$. D'un autre côté, nous pouvons montrer qu'une partition optimale de ℓK_k en stables ou cliques est nécessairement une partition minimum en stables, ou bien une partition minimum en cliques.

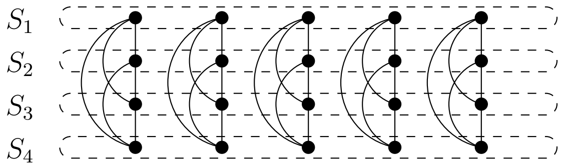


FIG. 5.10 – Le graphe $5\,K_4$.

Supposons que ce ne soit pas le cas : il existe une partition optimale de ℓK_k en stables ou cliques comprenant au moins un stable et au moins une clique. Tout d'abord, nous pouvons considérer que chaque stable de cette partition est induit par un des k stables $S_1, \ldots, S_k$ (il suffit de numéroter à nouveau les sommets de chaque clique C_i de façon à ce que la condition soit satisfaite). À présent, nous affirmons que cette partition doit comporter exactement k stables $S'_1, \ldots, S'_k$, chacun induit par un stable S_j différent ($1 \leq j \leq k$). En effet, si aucun stable de la partition n'est induit par le stable S_j, alors les sommets de S_j doivent être couverts par ℓ cliques disjointes. La partition comprenant au moins un stable par hypothèse, celle-ci est donc de cardinal au moins $\ell+1$, ce qui est une contradiction (toute partition optimale en stables ou cliques est de cardinal au plus $\min\{k, \ell\}$). D'un autre côté, si deux stables de cette partition sont induits par un même stable S_j, alors celle-ci ne peut pas non plus être optimale (l'union de ces deux stables permettrait de diminuer son cardinal de un). Pour conclure, nous faisons apparaître une nouvelle contradiction : les sommets de chaque clique de la partition (il en existe au moins une par hypothèse) peuvent être ajoutés aux stables $S'_1, \ldots, S'_k$, permettant ainsi de diminuer d'au moins un son cardinal.

Ainsi, une partition optimale en stables ou cliques est donnée par une partition minimum en stables si $k \leq \ell$, ou bien par une partition minimum en cliques si $k \geq \ell$. Dans le même temps, nous obtenons que le graphe ℓK_k n'admet aucune partition en moins de $\min\{k, \ell\}$ stables ou cliques. Le nombre n de sommets d'un graphe ℓK_k étant exactement $k\ell$, nous obtenons la proposition suivante lorsque $k = \ell = \sqrt{n}$.

Proposition 5.10 (Borne inférieure) *Pour tout entier $k = \ell \geq 1$, le graphe ℓK_k n'admet aucune partition en moins de $\sqrt{n}$ stables ou cliques, où n dénote le nombre de sommets du graphe.*

Corollaire 5.10 (Borne inférieure) *Pour tout entier $n \geq 1$, il existe un graphe d'intervalles propres n'admettant aucune partition en moins de $\lfloor \sqrt{n} \rfloor$ stables ou cliques.*

Preuve. Définissons le graphe ℓK_k avec $\ell = k = \lfloor\sqrt{n}\rfloor$, puis ajoutons à celui-ci exactement $n - \lfloor\sqrt{n}\rfloor^2$ sommets isolés. Ce graphe possède une représentation par intervalles unitaires et, d'après la proposition précédente, n'admet aucune partition en moins de $\lfloor\sqrt{n}\rfloor$ stables ou cliques. □

Abordons à présent les questions de complexité, avec pour commencer, deux résultats négatifs quelque peu attendus.

Proposition 5.11 (Complexité générale) *Le problème de la partition en trois stables ou cliques est $\mathcal{NP}$-complet pour les graphes planaires.*

Preuve. Nous effectuons une réduction depuis le problème de la 3-coloration, qui est $\mathcal{NP}$-complet pour les graphes planaires [59]. Soit G un graphe non vide et $G' = G \cup 4K_3$. Il est facile de voir que si G admet une 3-coloration, alors G' admet une partition en trois stables. D'un autre côté, toute partition de G' en trois stables ou cliques ne peut contenir aucune clique. Supposons le contraire : la partition de $G' = G \cup 4K_3$ comprend au moins une clique. Cette clique appartient soit au sous-graphe G, soit au sous-graphe $4K_3$. Dans les deux cas nous obtenons une contradiction puisque le graphe $3K_3$ n'admet aucune partition en moins de trois stables ou cliques d'après la proposition 5.10. Par conséquent, si G' admet une partition en trois stables ou cliques, alors G admet nécessairement une 3-coloration. Comme G' est planaire si et seulement si G l'est aussi, la réduction reste valide pour les graphes planaires. □

Remarque. Il est possible de montrer à l'aide de quelques détails techniques supplémentaires que le graphe $G' = G \cup 3K_3$ admet une coloration en trois stables ou cliques si et seulement si le graphe G admet une 3-coloration.

Proposition 5.12 (Complexité générale) *Déterminer une partition minimum en stables ou cliques est un problème $\mathcal{NP}$-difficile pour les graphes duaux.*

Preuve. Pour les graphes duaux, le problème de la q-coloration est $\mathcal{NP}$-complet [90]. Or, la preuve précédente peut être aisément étendue de façon à montrer que le graphe $G' = G \cup (q+1)K_q$ admet une coloration en q stables ou cliques si et seulement si le graphe G admet une q-coloration. Comme G' est un graphe dual si et seulement si G l'est aussi (toute clique K_q est le dual d'une étoile $K_{1,q}$), la proposition est démontrée. □

D'un autre côté, déterminer si un graphe admet une partition en deux stables ou cliques revient à tester si celui-ci est un graphe biparti, le complément d'un graphe biparti, ou bien un graphe scindé. Les graphes bipartis ou scindés étant reconnus en temps et espace linéaire (cf. [75, p. 154]), nous avons la proposition suivante.

Proposition 5.13 (Complexité générale) *Le problème de la partition d'un graphe en deux stables ou cliques peut être résolu en temps $O(n^2)$ et espace linéaire.*

Brandstädt [20] a introduit une généralisation des graphes scindés appelés (k, ℓ)-graphes : un graphe est un (k, ℓ)-graphe si ses sommets admettent une partition en exactement k

stables et ℓ cliques. De ce point de vue, les graphes scindés sont donc des $(1,1)$-graphes. Brandstädt [20, 21] a étudié le problème de la reconnaissance des (k,ℓ)-graphes, très proche du problème de la partition en stables ou cliques. Pour $k \leq 2$ et $\ell \leq 2$, il donne des algorithmes de reconnaissance en $O((n+m)^2)$. Pour $k \geq 3$ ou $\ell \geq 3$, le problème est $\mathcal{NP}$-complet puisqu'il contient le problème de la 3-coloration. La complexité des problèmes de partition de graphes en stables ou cliques a aussi été étudiée par Feder *et al.* [50], ainsi que Bryś et Lonc [23].

Les $(1,1)$-graphes (*i.e.* les graphes scindés) sont triangulés (cf. [75, p. 151–152]). Mais, à quelles conditions un graphe triangulé est-il aussi un (k,ℓ)-graphe ? Récemment, Hell *et al.* [86] ont répondu à cette question au travers de la proposition suivante.

Proposition 5.14 (Hell *et al.*, 2004) *Tout graphe triangulé admet une partition en k stables et ℓ cliques si et seulement s'il ne contient pas le sous-graphe induit $(\ell+1)K_{k+1}$.*

En s'appuyant sur cette caractérisation, Hell *et al.* [86] donnent un algorithme en temps $O(n(n+m))$ et espace linéaire pour la reconnaissance des (k,ℓ)-graphes triangulés, où n et m dénotent respectivement le nombre de sommets et le nombre d'arêtes du graphe. Leur algorithme trouve, pour $k > 0$ fixé, la plus petite valeur de l tel que le graphe est un (k,ℓ)-graphe.

Proposition 5.15 (Hell *et al.*, 2004) *Le problème de la reconnaissance des (k,ℓ)-graphes triangulés peut être résolu en temps $O(n(n+m))$ et espace linéaire.*

Des travaux de Hell *et al.* [86] découlent plusieurs résultats intéressants concernant le problème de la partition des graphes d'intervalles en stables ou cliques.

Corollaire 5.11 (Nouvelle borne supérieure) *Tout graphe triangulé admet une partition en moins de $2\lfloor\sqrt{n}\rfloor$ stables ou cliques. De plus, cette partition peut être calculée en temps $O(n(n+m))$ et espace linéaire.*

Preuve. Un graphe à n sommets ne contient pas de copie induite de $(\ell+1)K_{k+1}$ pour $k = \ell = \lfloor\sqrt{n}\rfloor$. D'après la proposition 5.14, il admet donc une partition en k stables et ℓ cliques, soit une partition en $2\lfloor\sqrt{n}\rfloor$ stables ou cliques. De plus, cette partition peut être calculée en temps $O(n(n+m))$ et espace linéaire en utilisant l'algorithme de reconnaissance des (k,ℓ)-graphes triangulés de Hell *et al.* [86] avec $k = \lfloor\sqrt{n}\rfloor$. □

Corollaire 5.12 (Nouvelle borne supérieure) *Tout graphe d'intervalles (resp. d'arcs) admet une partition en moins de $2\lfloor\sqrt{n}\rfloor$ (resp. $2\lfloor\sqrt{n}\rfloor+1$) stables ou cliques. De plus, cette partition peut être calculée en temps $O(n(n+m))$ et espace linéaire.*

Preuve. Pour les graphes d'intervalles, le résultat découle du corollaire précédent. Pour les graphes d'arcs, celui-ci s'obtient comme pour le problème de la partition en sous-graphes d'intervalles propres, en retirant les arcs contenant un point p du cercle. □

Comme pour le problème de la partition en sous-graphes d'intervalles propres, nous pensons pouvoir étendre ce type de résultat aux *graphes de tolérances*. D'un autre côté, *existe-t-il des graphes d'intervalles n'admettant aucune partition en moins de* $2\lfloor\sqrt{n}\rfloor$ *stables ou cliques ?* De même, *existe-t-il des graphes parfaits n'admettant aucune partition en moins de* $\lceil n/2 \rceil$ *stables ou cliques ?*

Corollaire 5.13 (Complexité) *Le problème de la partition minimum d'un graphe triangulé (ou d'intervalles) en stables ou cliques peut être résolu en temps* $O(n^{1.5}(n+m))$ *et espace linéaire.*

Preuve. Soit G un graphe triangulé. Tout d'abord, pour $k = 0$, calculons en temps et espace linéaire une partition minimum de G en cliques [70]. Ensuite, pour $k = 1, \ldots, 2\lfloor\sqrt{n}\rfloor$, calculons une partition de G en k stables et ℓ cliques avec ℓ minimum à l'aide de l'algorithme de Hell *et al.* [86] en temps $O(n(n+m))$ et espace linéaire. De ces $2\lfloor\sqrt{n}\rfloor + 1$ partitions, gardons celle de plus petit cardinal. Cette dernière est nécessairement optimale puisque toutes les partitions possibles ont été testées (d'après le corollaire précédent, la partition optimale est de cardinal inférieur ou égal à $2\lfloor\sqrt{n}\rfloor$). □

Corollaire 5.14 (Complexité) *Une partition minimum d'un graphe d'arcs en stables ou cliques peut être approchée à un près en temps* $O(n^{1.5}(n+m)$ *et espace linéaire.*

Preuve. Après avoir retiré les arcs du cercle contenant un point p, il est possible d'effectuer une partition optimale du reste des arcs (devenus des intervalles) en stables ou cliques grâce au corollaire précédent. Comme cette partition est d'un cardinal nécessairement inférieur ou égal au cardinal d'une partition minimum du graphe d'arcs de départ en stables ou cliques, nous obtenons le résultat. □

En conclusion, les questions de complexité suivantes demeurent ouvertes. *Existe-t-il un algorithme linéaire en temps et espace pour déterminer une partition minimum d'un graphe triangulé (ou d'intervalles) en stables ou cliques ? Quid d'un algorithme polynomial pour déterminer une partition minimum d'un graphe d'arcs en stables ou cliques ? D'un algorithme polynomial pour déterminer une partition minimum d'un graphe parfait en stables ou cliques ? D'un algorithme polynomial d'approximation avec ratio constant (ou logarithmique) dans le pire des cas pour déterminer une partition minimum d'un graphe quelconque en stables ou cliques ?*

Note. Dans leur papier, Hell *et al.* [86] affirment avoir abaissé le temps d'exécution de leur algorithme pour la reconnaissance des (k,ℓ)-graphes triangulés à $O(n+m)$ (le résultat doit paraître sous peu dans un papier signé par les mêmes auteurs [87]).

5.3.2 Le plus grand sous-graphe d'intervalles propres

Ce problème est en quelque sorte la version "*packing*" du problème de la plus petite partition en sous-graphes d'intervalles propres. Voici les premiers résultats que nous pouvons établir concernant ce problème.

Proposition 5.16 *Tout graphe scindé (ou à seuil) contient un sous-graphe d'intervalles propres de taille au moins $\lceil n/2 \rceil + 1$, pour $n > 3$. De plus, cette borne est la meilleure possible.*

Preuve. Soit $G = (C \cup S, E)$ un graphe scindé avec S un stable et C une clique ($|S|+|C| > 3$). Nous considérerons que $|C| \geq 1$ et $|S| \geq 3$, sans quoi le graphe possède une représentation par intervalles propres et la borne est immédiate. Si $|C| \geq \lceil n/2 \rceil - 1$, la borne est établie puisque le sous-graphe induit par l'ensemble C et deux sommets quelconques de l'ensemble S est un sous-graphe d'intervalles propres. Dans le cas contraire, le stable S, qui induit de façon triviale un sous-graphe d'intervalles propres, est de taille strictement supérieure à $\lfloor n/2 \rfloor + 1$.

Voici la construction d'un graphe scindé pour lequel la borne en question est atteinte ($n > 3$). Soit S un stable composé de $\lfloor n/2 \rfloor + 1$ sommets. Chaque sommet de ce stable est relié à $\lceil n/2 \rceil - 1$ sommets formant une clique C. Si un sous-graphe d'intervalles propres contient au moins trois sommets du stable S, alors celui-ci est nécessairement de taille inférieure ou égale à $\lfloor n/2 \rfloor + 1$ (puisqu'il ne peut contenir aucun sommet de C sans induire $K_{1,3}$). D'un autre côté, si un sous-graphe d'intervalles propres contient au plus deux sommets de S, alors celui-ci est de taille inférieure à $\lceil n/2 \rceil + 1$ (au mieux il contient tous les sommets de C, plus les deux sommets de S). $\square$

Le lecteur notera que le graphe scindé pour lequel la borne est atteinte est aussi un graphe à seuil.

Corollaire 5.15 (Borne supérieure) *Pour tout entier $n > 3$, il existe un graphe à seuil dont le plus grand sous-graphe d'intervalles propres est de taille au plus $\lceil n/2 \rceil + 1$.*

Voici un corollaire du célèbre Théorème des Graphes Parfaits de Lovász (cf. [75, p. 53–58]) qui établit de façon directe une borne inférieure sur la taille du plus grand sous-graphe d'intervalles propres dans les graphe parfaits, et *a fortiori* dans les graphes d'intervalles.

Proposition 5.17 (Borne inférieure) *Tout graphe parfait contient un sous-graphe d'intervalles propres de taille au moins $\lfloor \sqrt{n} \rfloor$. De plus, ce sous-graphe peut être exhibé en temps et espace polynomial.*

Preuve. Lovász (cf. [75, p. 53–58]) a montré que si G est un graphe parfait, alors $\omega(G)\alpha(G) \geq n$. Par conséquent, si $\omega(G) \leq \lfloor \sqrt{n} \rfloor$, alors $\alpha(G) \geq \lfloor \sqrt{n} \rfloor$ (de façon symétrique, si $\alpha(G) \leq \lfloor \sqrt{n} \rfloor$, alors $\omega(G) \geq \lfloor \sqrt{n} \rfloor$). Ainsi, un sous-graphe d'intervalles propres de taille au moins $\lfloor \sqrt{n} \rfloor$ s'obtient en temps et espace polynomial de la manière suivante : calculer un stable maximum et une clique maximum [79], puis garder le plus grand des deux ensembles. $\square$

Puisque pour les graphes d'intervalles, calculer un stable maximum et une clique maximum ne prend qu'un temps et espace linéaire [70, 81] (voir aussi l'algorithme Cliques-Intervalles, p. 45), nous obtenons le corollaire suivant.

Corollaire 5.16 (Borne inférieure) *Tout graphe d'intervalles contient un sous-graphe d'intervalles propres de taille au moins $\lfloor \sqrt{n} \rfloor$. De plus, ce sous-graphe peut être exhibé en temps et espace linéaire.*

Nous pouvons cependant améliorer cette borne inférieure en utilisant les résultats que nous avons établis concernant le problème de la partition d'un graphe d'intervalles en sous-graphes d'intervalles propres.

Proposition 5.18 (Nouvelle borne inférieure) *Tout graphe d'intervalles (resp. d'arcs) contient un sous-graphe d'intervalles propres de taille supérieure ou égale à* $\lceil n/\lceil \log_3 n \rceil \rceil$ *(resp.* $\lceil n/(\lceil \log_3 n \rceil + 1)\rceil$*), pour* $n > 1$*. De plus, ce sous-graphe peut être exhibé en temps* $O(n \log n + m)$ *et espace linéaire.*

Preuve. D'après la proposition 5.4, tout graphe d'intervalles (resp. d'arcs) admet une partition en moins de $\lceil \log_3 n \rceil$ (resp. $\lceil \log_3 n \rceil + 1$) sous-graphes d'intervalles propres. D'après le principe des casiers (de l'anglais *pigeon hole principle*), au moins un de ces sous-graphes contient plus de $\lceil n/\lceil \log_3 n \rceil \rceil$ (resp. $\lceil n/(\lceil \log_3 n \rceil + 1)\rceil$) sommets. □

Notons que $\lceil n/\lceil \log_3 n \rceil \rceil \geq \lfloor \sqrt{n} \rfloor$ pour tout $n > 1$. En nous appuyant sur les lemmes suivants, nous allons voir qu'il est encore possible d'affiner cette borne.

Lemme 5.4 *Soit* G *un graphe d'intervalles avec* $1 < \alpha(G) < t$*. Alors* G *contient un sous-graphe d'intervalles propres de taille au moins* $\lceil n/\lceil \log_3((3t-3)/2)\rceil \rceil$.

Preuve. D'après le corollaire 5.3 et le principe des casiers. □

Lemme 5.5 *Soit* G *un graphe d'intervalles avec* $t \leq \alpha(G) \leq n$*. Alors* G *contient un sous-graphe d'intervalles propres de taille au moins* t.

Preuve. Puisque tout stable induit un sous-graphe d'intervalles propres, le plus grand sous-graphe d'intervalles propres est de taille au moins t. □

Remarque. Une conséquence intéressante de ces deux lemmes est celle-ci. Soit G un graphe d'intervalles satisfaisant une des deux conditions suivantes, pour $t = O(1)$:

(1) G ne possède pas de copie induite de $K_{1,t}$ (ou de stable de taille supérieure à t) ;
(2) G possède une copie induite de $K_{1,n-t}$ (ou un stable de taille supérieure à $n - t$).

Alors G contient un sous-graphe d'intervalles propres de taille $\Omega(n)$ calculable en temps et espace linéaire. Notons que cette remarque reste valable même si le lemme de partition linéaire est utilisé pour effectuer la partition.

En joignant les deux lemmes précédents, nous obtenons que tout graphe d'intervalles contient un sous-graphe d'intervalles propres de taille au moins $f(n,t) = \min\{\lceil n/f_0(t)\rceil, t\}$ pour n'importe quel entier t variant entre 2 et n, où $f_0(t) = \lceil \log_3((3t-3)/2)\rceil$. La recherche de la valeur de $t \in \{2, \ldots, n\}$ pour laquelle $f(n,t)$ atteint son maximum se fait de la façon suivante. La fonction $\lceil n/f_0(t)\rceil$ est monotone décroissante, alors que la fonction t est monotone croissante. De ce fait, une borne inférieure du maximum recherché est $n/f_0(\lceil \tilde{t} \rceil)$ avec $\tilde{t}$ solution unique de l'équation $\tilde{t} = n/\tilde{f}_0(\tilde{t})$ où $\tilde{f}_0(t) = \log_3(3t/2)$. La valeur de $\tilde{t}$ est donnée par

$$\tilde{t} = \frac{\tilde{n}}{W_0(\beta \tilde{n})}$$

où $\tilde{n} = n \ln 3$, $\beta = \frac{3}{2}$ et W_0 est la branche principale de la fonction W de Lambert.

La fonction W de Lambert vérifie $W(z)e^{W(z)} = z$ pour $z \in \mathbb{C}$. Cette fonction n'est pas injective et comporte une infinité de branches, dont une seule de $\mathbb{R}^+$ dans $\mathbb{R}^+$. Cette dernière, appelée branche principale et notée W_0, est connue en analyse combinatoire pour sa relation avec la série génératrice exponentielle $T(z)$ des arbres étiquetés enracinés (connue sous le nom de *tree function* en anglais). En effet, si T_n dénote le nombre d'arbres enracinés sur n sommets étiquetés, alors $T(z) = \sum_{n=1}^{\infty} T_n \frac{z^n}{n!} = -W_0(-z)$. Le comportement asymptotique en zéro ou à l'infini de la branche W_0 est donné par

$$W_0(z) \sim \ln z - \ln \ln z + \sum_{k=0}^{\infty} \sum_{m=1}^{\infty} c_{k,m} \frac{(\ln \ln z)^m}{(\ln z)^{k+m}}$$

où les $c_{k,m}$ sont des constantes indépendantes de z. Lorsque $z \geq e/\alpha$, nous avons l'encadrement $\ln(\alpha z) - \ln \ln(\alpha z) \leq W_0(\alpha z) \leq \ln(\alpha z)$. Pour plus de détails sur la fonction W de Lambert et sa branche principale W_0, nous renvoyons le lecteur à [27].

Lorsque $n \geq 2$, nous pouvons donc borner $\tilde{t}$ par

$$\tilde{t} \leq \frac{n}{\log_3(\beta' n) - \log_3 \ln(\beta' n)}$$

où $\beta' = \frac{3 \ln 3}{2}$, et de là obtenir la borne inférieure $n/f_0(\lceil \tilde{t} \rceil) \geq n/\lceil \log_3(\beta \tilde{t}) \rceil$ où $\beta = \frac{3}{2}$. Afin d'y voir plus clair, voici un encadrement de cette borne inférieure :

$$\frac{n}{\lceil \log_3 n \rceil} \leq \frac{n}{\lceil \log_3(\beta \tilde{t}) \rceil} \leq \frac{n}{\lceil \log_3(\beta' n) - \log_3 \log_3(\beta' n) \rceil}$$

Enfin, nous pouvons vérifier que $\lceil n/\lceil \log_3 n \rceil \rceil < n/\lceil \log_3(\beta \tilde{t}) \rceil$ pour $n \geq 244$, et donc que la borne inférieure $n/\lceil \log_3(\beta \tilde{t}) \rceil$ est de façon asymptotique meilleure que la borne de la proposition 5.18 (voir la figure 5.11).

Les lemmes 5.4 et 5.5 peuvent aussi être utilisés afin d'affiner la borne inférieure en vigueur pour les graphes d'arcs (proposition 5.18). Tout d'abord, retirons de la représentation l'ensemble des arcs contenant un point p du cercle. Si le graphe d'intervalles G' induit par les intervalles restants est tel que $1 < \alpha(G') < t$, alors le graphe d'arc contient un sous-graphe d'intervalles propres de taille au moins $\lceil n/(\lceil \log_3((3t-3)/2) \rceil + 1) \rceil$ (puisque celui-ci admet une partition en $\lceil \log_3((3t-3)/2) \rceil + 1$ sous-graphes d'intervalles propres). Dans le cas contraire ($t \leq \alpha(G') \leq n$), il existe un sous-graphe d'intervalles propres de taille au moins t. Par conséquent, nous rechercherons ici la valeur de t pour laquelle la fonction $\min\{\lceil n/(f_0(t)+1) \rceil, t\}$ atteint son maximum. En procédant à une analyse similaire à celle que nous avons menée précédemment, nous obtenons

$$\tilde{t} = \frac{\tilde{n}}{W(\beta \tilde{n})} \leq \frac{n}{\log_3(\beta' n) - \log_3 \ln(\beta' n)}$$

$$n/(f_0(\lceil \tilde{t} \rceil) + 1) \geq n/\lceil \log_3(\beta \tilde{t}) \rceil$$

où $\tilde{n} = n \ln 3$, $\beta = \frac{9}{2}$ et $\beta' = \frac{9 \ln 3}{2}$.

En conclusion de la discussion précédente, nous pouvons établir le résultat suivant.

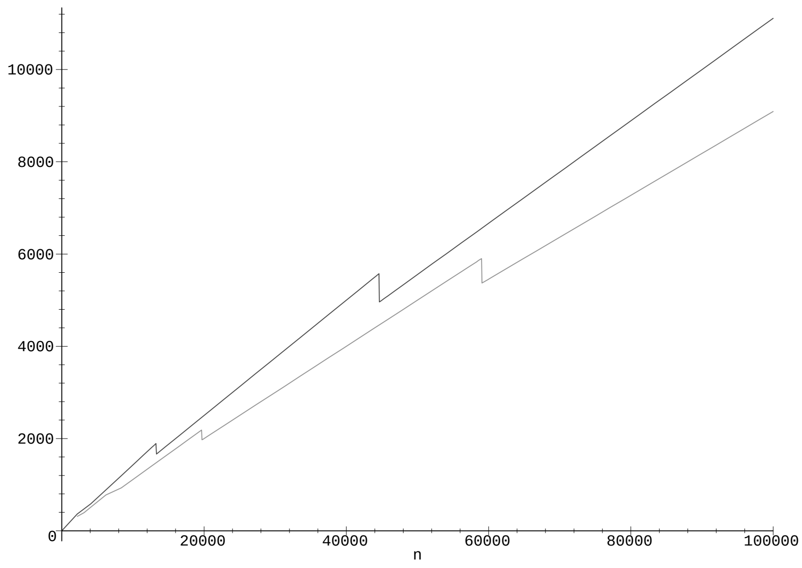


FIG. 5.11 – L'évolution des bornes supérieures : la borne de la proposition 5.19 (tracé foncé) est de façon asymptotique supérieure à la borne de la proposition 5.18 (tracé clair).

Proposition 5.19 (Borne inférieure raffinée) *Tout graphe d'intervalles contient un sous-graphe d'intervalles propres de taille au moins*

$$\frac{n}{\left\lceil \log_3 \left(\frac{3}{2} \frac{n}{\log_3(\frac{3\ln 3}{2}n) - \log_3 \ln(\frac{3\ln 3}{2}n)} \right) \right\rceil}$$

et tout graphe d'arcs contient un sous-graphe d'intervalles propres de taille au moins

$$\frac{n}{\left\lceil \log_3 \left(\frac{9}{2} \frac{n}{\log_3(\frac{9\ln 3}{2}n) - \log_3 \ln(\frac{9\ln 3}{2}n)} \right) \right\rceil}$$

pour $n > 1$. De plus, ce sous-graphe peut être exhibé en temps $O(n \log(\frac{n}{\log n - \log\log n}) + m)$ et espace linéaire.

À présent, discutons de la complexité du problème. Déterminer un plus grand sous-graphe d'intervalles propres dans un graphe contient le problème de la reconnaissance des graphes d'intervalles propres, qui est soluble en temps et espace linéaire [31]. Comme pour le problème de la partition en sous-graphes d'intervalles propres, nous ne sommes pas parvenus à déterminer la complexité du problème, même en nous restreignant aux graphes d'intervalles. Toutefois, nous montrons que le problème devient polynomial lorsque l'on se restreint à la sous-classe des graphes à seuil.

Proposition 5.20 (Complexité) *Le problème de la détermination du plus grand sous-graphe d'intervalles propres peut être résolu en temps et espace linéaire pour les graphes à seuil.*

Preuve. Nous rappelons qu'un graphe à seuil est composé d'une clique $C = C_1 \cup \cdots \cup C_r$ et d'un stable $S = S_1 \cup \cdots \cup S_r$ ($r \leq n$ et C_i, S_i non vides pour $i = 1, \ldots, r$) tel qu'un sommet de S_i est adjacent à un sommet de $C_{i'}$ si et seulement si $i' > i$ pour tout $i, i' \in \{1, \ldots, r\}$. Une telle représentation peut être calculée en temps et espace linéaire à partir de la partition des sommets du graphe selon leur degré (cf. [75, p. 223–227] pour plus de détails).

Tout sous-graphe d'intervalles propres de cardinal maximum dans un graphe à seuil doit contenir au moins un sommet de C. En effet, si un sous-graphe d'intervalles propres ne contient que des sommets de S, alors il ne peut être de cardinal maximum puisque l'on peut lui ajouter les sommets de C_1 (qui ne sont reliés à aucun sommet de $S_1 \cup \cdots \cup S_r$ par définition). Maintenant, soit i le plus grand indice tel qu'un sommet de C_i appartient à un plus grand sous-graphe d'intervalles propres ($1 \leq i \leq r$). Nous affirmons que ce sous-graphe contient (i) tous les sommets de $C_1 \cup \cdots \cup C_i$, (ii) tous les sommets de $S_i \cup \cdots \cup S_r$ et (iii) un sommet de $S_1 \cup \cdots \cup S_{i-1}$ si $i > 1$ et $|S_1 \cup \cdots \cup S_{i-1}| \geq 1$. Les assertions (i) et (ii) découlent respectivement du fait que tout sommet de C_i induit une clique avec les sommets de $C_1 \cup \cdots \cup C_i$ et n'est relié à aucun sommet de $S_i \cup \cdots \cup S_r$. Pour prouver (iii), il suffit de voir que le sous-graphe induit par un sommet de C_i, un sommet de C_1 et deux sommets de $S_1 \cup \cdots \cup S_{i-1}$ est isomorphe à $K_{1,3}$. Ainsi, la taille d'un plus grand sous-graphe d'intervalles propres contenant au moins un sommet de C_i est donnée par $t_i = \sum_{j=1}^{i} |C_j| + \sum_{j=i}^{r} |S_j|$ ($+1$ si $\sum_{j=1}^{i-1} |S_j| \geq 1$).

Déterminer un plus grand sous-graphe d'intervalles propres dans un graphe à seuil revient à calculer l'indice i ($1 \leq i \leq r$) pour lequel l'expression t_i atteint son maximum. Correctement implanté, ce calcul se fait en temps et espace linéaire. $\square$

Remarque. En d'autres termes, le nombre de sous-graphe d'intervalles propres de taille maximale dans un graphe à seuil est exactement $r \leq n$.

Enfin, voici un résultat similaire à celui que nous avons obtenu pour le problème de la partition minimum en sous-graphe d'intervalles propres (voir le corollaire 5.6).

Corollaire 5.17 (Complexité) *Un plus grand sous-graphe d'intervalles propres peut être approché en temps $O(n \log(\frac{n}{\log n - \log\log n}) + m)$ et espace linéaire avec un ratio*

$$\frac{1}{\left\lceil \log_3 \left(\frac{3}{2} \frac{n}{\log_3(\frac{3\ln 3}{2} n) - \log_3 \ln(\frac{3 \ln 3}{2} n)} \right) \right\rceil}$$

pour les graphes d'intervalles et un ratio

$$\frac{1}{\left\lceil \log_3 \left(\frac{9}{2} \frac{n}{\log_3(\frac{9\ln 3}{2} n) - \log_3 \ln(\frac{9 \ln 3}{2} n)} \right) \right\rceil}$$

pour les graphes d'arcs (avec $n > 1$).

Preuve. D'après la proposition 5.19 et le fait que la taille d'un plus grand sous-graphe d'intervalles propres est bornée par n. □

En conclusion, deux questions restent à élucider. *Tout graphe d'intervalles contient-il un graphe d'intervalles propres de taille au moins* $\lceil n/2 \rceil + 1$ *? Existe-t-il un algorithme polynomial pour déterminer le plus grand sous-graphe d'intervalles propres dans un graphe d'intervalles ?*

5.3.3 Le plus petit nombre d'intervalles de longueurs différentes

Dans cette dernière partie, nous abordons brièvement un problème introduit par Leibowitz dans sa thèse de doctorat [108] et à nouveau posé par Golumbic [75, p. 197] dans son livre.

Soit G un graphe d'intervalles. Nous noterons IC(G) le plus petit nombre d'intervalles de longueurs différentes nécessaires à la représentation d'un graphe d'intervalles G (dit *interval count* en anglais). Clairement, IC(G) $= 1$ si et seulement si G est un graphe d'intervalles unitaires. Comme IC($K_{1,3}$) $= 2$, nous avons d'après le théorème de Roberts [131] (voir aussi le théorème 1.3) que IC(G) $= 1$ si et seulement si G est un graphe d'intervalles sans $K_{1,3}$.

Golumbic [75, p. 197] pose les deux problèmes suivants : *caractériser les graphes pour lesquels* IC(G) $= k$ *(avec* $k \geq 2$*) et déterminer de bonnes bornes supérieure et inférieure pour* IC(G) *?* À notre connaissance, aucun résultat n'est paru sur le sujet depuis la première étude menée par Leibowitz [108], malheureusement jamais publiée. Pour commencer, nous relatons deux résultats obtenus Leibowitz [108] et rapportés par Golumbic [75, p. 197].

Proposition 5.21 (Leibowitz, 1978) *Pour tout graphe* G *appartenant à une des deux classes suivantes, nous avons* IC(G) $= 2$ *:*

(1) les graphes d'intervalles bipartis ;

(2) les graphes à seuil.

Preuve. (1). Tout graphe d'intervalles biparti est composé d'une union d'arbres T_i, où chaque T_i est composé d'étoiles numérotées de 1 à i tel que le sommet central de l'étoile j est relié au sommet central de l'étoile $j+1$ $(1 \leq j \leq i-1)$. Nous montrons comment construire une représentation par intervalles ouverts d'un arbre T_i à l'aide d'intervalles de deux longueurs différentes. Il est ensuite très facile d'obtenir une représentation de l'union de plusieurs arbres T_i sans modifier la longueur des intervalles (en décalant les représentations de chaque arbre sur l'axe des réels).

Soit ℓ le plus grand nombre d'arêtes portées par une étoile de T_i. Nous pouvons représenter les sommets centraux des i étoiles de T_i par un intervalle de longueur ℓ et les sommets périphériques par des intervalles de longueur un. Notons $\bar{I}_j$ l'intervalle de longueur ℓ correspondant au sommet central de l'étoile j. Les i intervalles $\bar{I}_1, \ldots, \bar{I}_i$ sont disposés de façon à ce que l'intervalle $\bar{I}_j$ chevauche l'intervalle $\bar{I}_{j-1}$ par la droite avec $|\bar{I}_{j-1} \cap \bar{I}_j| \leq 1$ $(2 \leq j \leq i)$. Pour chaque $j = 1, \ldots, i$, les intervalles de longueur un correspondant aux sommets périphériques de l'étoile j peuvent alors être insérés entre les intervalles $\bar{I}_{j-1}$ et $\bar{I}_{j+1}$ de manière à être recouverts entièrement par l'intervalle $\bar{I}_j$ et à ne pas se chevaucher

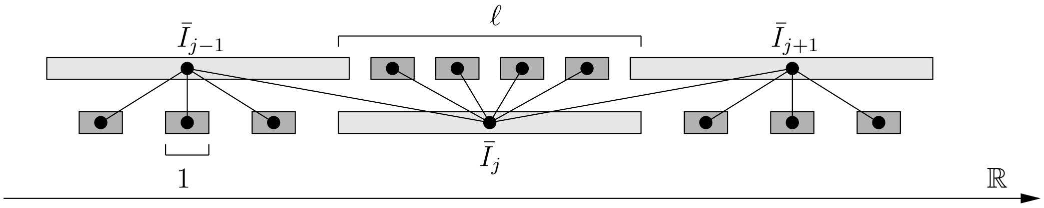


FIG. 5.12 – La représentation d'un graphe d'intervalles biparti.

deux-à-deux (voir la figure 5.12). En effet, les étoiles 1 et i ne comportent pas plus de $\ell - 1$ sommets périphériques et les autres étoiles (de 2 à $i - 1$) pas plus de $\ell - 2$.

(2). Un graphe à seuil est représentable par une clique $C = C_1 \cup \cdots \cup C_r$ et un stable $S = S_1 \cup \cdots \cup S_r$ ($r \leq n$ et C_i, S_i non vides pour $i = 1, \ldots, r$) tel qu'un sommet de S_i soit adjacent à un sommet de $C_{i'}$ si et seulement si $i' > i$ pour tout $i, i' \in \{1, \ldots, r\}$ (cf. [75, p. 223–227] pour plus de détails). Soit ℓ le nombre de sommets de l'ensemble S. Les sommets du graphe à seuil seront représentés par des intervalles ouverts, de longueur $\ell + 1$ pour les sommets de C et de longueur un pour les sommets de S. Les intervalles correspondant aux sommets de C_i auront les mêmes extrémités gauches et droites, pour tout $i = 1, \ldots, r$.

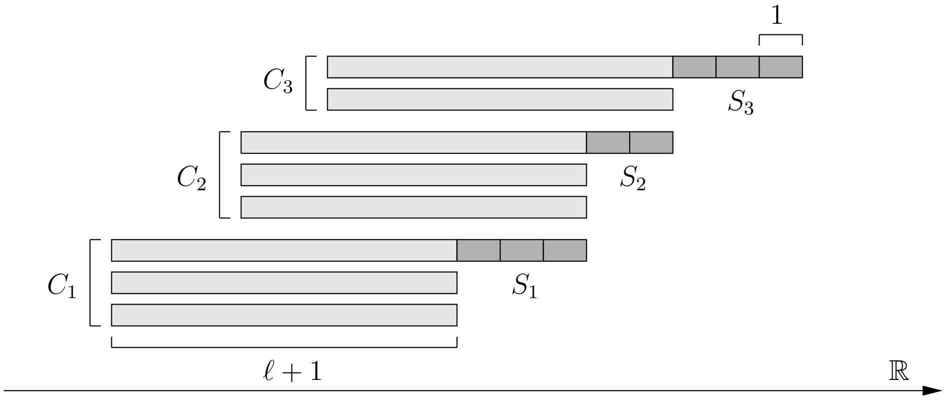


FIG. 5.13 – La représentation d'un graphe à seuil.

Une fois les intervalles de C_1 placés, nous procédons de la manière suivante. Les intervalles de S_i ($1 \leq i \leq r - 1$) sont disposés à la droite des intervalles de C_i sur l'axe des réels, de façon à ce que l'extrémité gauche du premier intervalle de S_i se confonde avec l'extrémité droite des intervalles de C_i et que l'extrémité gauche du $j^{ème}$ intervalle de S_i se confonde avec l'extrémité droite du précédent ($j > 1$). Ensuite, nous recouvrons les intervalles de $C_1, \ldots, C_i$ et $S_1, \ldots, S_i$ à l'aide des intervalles de C_{i+1}, de manière à ce que les extrémités droites de ces derniers se confondent avec l'extrémité droite de l'intervalle de S_i le plus à droite (voir la figure 5.13). Ainsi, les intervalles de S_i, qui ne chevauchent aucun intervalle de $C_1 \cup \cdots \cup C_i$, induisent un stable avec les intervalles de $S_1 \cup \cdots \cup S_{i-1}$, et les intervalles de C_{i+1}, qui chevauchent tous les intervalles de $S_1 \cup \cdots \cup S_i$, induisent une clique avec les intervalles de $C_1 \cup \cdots \cup C_i$. En déroulant ce procédé de construction pour $i = 1, \ldots, r - 1$, puis en plaçant les intervalles du stable S_r à la droite des intervalles de C_r, nous obtenons la représentation désirée. □

Remarque. Correctement implantées, les constructions décrites ci-dessus peuvent être réalisées en temps et espace linéaire de façon à ce que les extrémités des intervalles soient des entiers entre 1 et n. Une méthode de construction similaire permet d'obtenir une représentation par intervalles fermés où les extrémités des intervalles sont entre 1 et $2n$.

Proposition 5.22 (Leibowitz, 1978) *Si après la suppression d'un de ses sommets, un graphe d'intervalles G devient un graphe d'intervalles unitaires, alors* $\mathrm{IC}(G) = 2$.

Bien que paraissant simple intuitivement, la construction d'une telle représentation nécessite quelques détails techniques fastidieux sur lesquels nous ne nous attarderons pas ici (la figure 5.14 illustre le type de configurations qui posent problème).

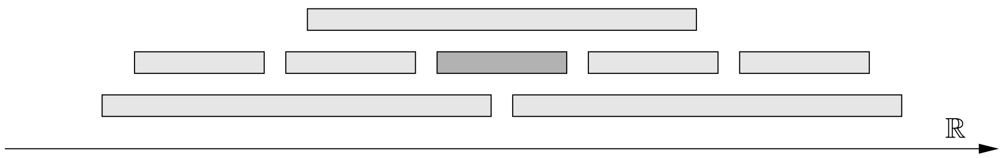


FIG. 5.14 – Les intervalles clairs induisent un graphe d'intervalles unitaires.

Ce problème a aussi un lien étroit avec le problème de la partition en sous-graphes d'intervalles propres. Clairement, tout graphe d'intervalles G admet une partition en moins de $\mathrm{IC}(G)$ sous-graphes d'intervalles propres, puisque les intervalles de même longueur induisent de façon triviale un sous-graphe d'intervalles unitaires. Ainsi, pour tout graphe G d'intervalles, nous avons $\xi(G) \leq \mathrm{IC}(G)$, où $\xi(G)$ représente le cardinal d'une partition minimum de G en sous-graphes d'intervalles propres. Suite à cette remarque, la proposition 5.2 et le corollaire 5.1 nous offrent le résultat suivant.

Corollaire 5.18 (Borne inférieure) *Pour tout entier $k \geq 1$, le graphe k-parti H_k est tel que $\mathrm{IC}(H_k) = k$ et pour tout entier $n \geq 1$, il existe un graphe G d'intervalles avec* $\mathrm{IC}(G) = \lfloor \log_3(2n+1) \rfloor$.

Note. Golumbic [75, p. 197] rapporte que Leibowitz [108] a réussi à construire pour tout entier k, un graphe G tel que $\mathrm{IC}(G) = k$.

Nous allons voir au travers de la proposition suivante que cette borne peut être largement relevée en utilisant simplement un sous-graphe du graphe H_k.

Proposition 5.23 (Nouvelle borne inférieure) *Pour tout entier $k \geq 1$, il existe un graphe k-parti H'_k à $n = 3k - 2$ sommets tel que* $\mathrm{IC}(H'_k) = (n+2)/3$.

Preuve. Une représentation par intervalles de ce graphe H_k s'obtient en définissant récursivement r stables $S_1, \ldots, S_k$ comme suit. Le stable S_1 est composé d'un unique intervalle ouvert. Pour $i = 2, \ldots, k$, le stable S_i est formé de trois intervalles ouverts de longueur égale ne se chevauchant pas deux-à deux et entièrement recouverts par l'intervalle central du stable S_{i-1} (voir la figure 5.15 pour un exemple de construction). L'ensemble $S_1, \ldots, S_k$ des stables résultants de cette construction induit un graphe k-parti dont le nombre de sommets est clairement $n = 3k - 2$.

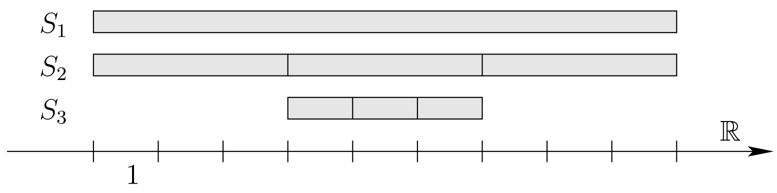


FIG. 5.15 – Une représentation par intervalles du graphe H'_3.

La représentation par intervalles que nous venons de décrire est telle que $\mathrm{IC}(H'_k) \leq k$, puisque les intervalles de chaque stable sont de même longueur. À présent, nous affirmons que l'intervalle central d'un stable S_i est nécessairement d'une longueur strictement inférieure à la longueur de l'intervalle central du stable S_{i-1}, pour tout $i = 2, \ldots, k$. En effet, le contraire implique immédiatement une contradiction, puisque l'intervalle central du stable S_{i-1} induit avec les trois intervalles du stable S_i une copie de $K_{1,3}$. De cette affirmation, nous déduisons aisément que les k intervalles centraux des stables $S_1, \ldots, S_k$ sont tous de longueurs différentes et de là que $\mathrm{IC}(H'_k) = k = (n + 2)/3$. □

Remarque. Comme H_k, la construction du graphe H'_k peut être réalisée à l'aide d'intervalles ayant tous des extrémités entières entre 0 et $(2n + 1)/3$. Il est à noter que H'_k est aussi un graphe scindé, puisque les intervalles centraux des stables $S_1, \ldots, S_k$ induisent une clique et les intervalles restants un stable.

Corollaire 5.19 (Nouvelle borne inférieure) *Pour tout entier $n \geq 1$, il existe un graphe scindé d'intervalles G avec $\mathrm{IC}(G) = \lfloor (n + 2)/3 \rfloor$.*

Preuve. Définissons le graphe H'_k avec k le plus grand entier tel que $n \geq 3k-2$, puis ajoutons à celui-ci exactement $n-3k+2$ sommets isolés. Clairement, ce graphe reste un graphe scindé et, par la proposition précédente, n'admet aucune partition en moins de $\lfloor (n + 2)/3 \rfloor$ sous-graphes d'intervalles propres. □

Remarque. Le lecteur notera le rapport surprenant entre la borne supérieure 2 établie pour les graphes à seuil et la borne inférieure $\lfloor (n + 2)/3 \rfloor$ établie pour les graphes scindés d'intervalles, qui sont pourtant deux sous-classes des graphes d'intervalles assez proches.

D'un autre côté, la meilleure borne supérieure que nous ayons actuellement est issue de la proposition 1.1.

Proposition 5.24 (Borne supérieure) *Pour tout graphe d'intervalles G, le nombre $\mathrm{IC}(G)$ est inférieur ou égal à $n - 1$.*

Preuve. La proposition 1.1 nous permet de représenter tout graphe d'intervalles G par des intervalles ouverts dont les extrémités sont dans $\{1, \ldots, n\}$. Or, dans une telle représentation, tous les intervalles ont des longueurs entières comprises entre 1 et $n - 1$. □

Remarque. De plus, cette représentation s'obtient en temps et espace linéaire (d'après la proposition 1.1, p. 11).

Une question que nous n'avons pas encore abordée jusqu'ici, sinon au travers de quelques remarques, concerne la complexité du problème. Cette question, qui reste ouverte à ce jour, est intimement liée à la caractérisation des graphes d'intervalles pour lesquels $\mathrm{IC}(G) = k$, lorsque $k \geq 2$. Nous pensons que ces graphes sont justement *les graphes d'intervalles qui ne contiennent pas H'_k comme sous-graphe induit*. Cette conjecture semble difficile à aborder de manière constructive au vue des efforts déployés pour prouver la proposition 5.22.

5.4 Synthèse des résultats

La figure 5.16 offre une synthèse des résultats que nous avons obtenus lors de l'étude des quatre problèmes suivants : (a) la plus petite partition d'un graphe d'intervalles en sous-graphes d'intervalles propres, (b) la plus petite partition d'un graphe d'intervalles en stables ou cliques, (c) le plus grand sous-graphe d'intervalles propres dans un graphe d'intervalles et enfin (d) le plus petit nombre d'intervalles de longueurs différentes nécessaire à la représentation d'un graphe d'intervalles.

	Borne inférieure	Borne supérieure	Complexité
Problème (a)	$\lfloor \log_3(2n+1) \rfloor$	$\lceil \log_3 n \rceil$	*ouvert*
Problème (b)	$\lfloor \sqrt{n} \rfloor$	$2\lfloor \sqrt{n} \rfloor$	$O(n^{1.5}(n+m))$
Problème (c)	$\dfrac{n}{\left\lceil \log_3\left(\frac{3}{2}\frac{n}{\log_3(\frac{3\ln 3}{2}n)-\log_3\ln(\frac{3\ln 3}{2}n)}\right)\right\rceil}$	$\lceil n/2 \rceil + 1$	*ouvert*
Problème (d)	$\lfloor (n+2)/3 \rfloor$	$n-1$	*ouvert*

FIG. 5.16 – Les principaux résultats du chapitre 5.

Il est à noter que toutes les bornes (inférieures et supérieures) ont été obtenues de façon constructive.

Chapitre 6

Conclusion

Dans ce dernier chapitre, nous dressons un bilan des principaux résultats présentés dans cette thèse, avant de dégager quelques perspectives de travail.

6.1 Bilan

Tout d'abord, voici un récapitulatif des résultats que nous avons obtenus concernant le problème d'ordonnancement avec exclusion mutuelle au travers des chapitres 2, 3 et 4.

	Graphes d'intervalles propres	Graphes à seuil	Graphes d'intervalles
$k = 2$	$O(n + m)$	$O(n + m)$	$O(n + m)$
$k = 3$	$O(n + m)$	$O(n + m)$	*ouvert*
$k \geq 4$	$O(n + m)$	$O(n + m)$	$\mathcal{NP}$-difficile [14]

Fig. 6.1 – La complexité du problème Ordo pour les graphes d'intervalles.

	Graphes d'arcs propres	Graphes d'arcs
$k = 2$	$O(n + m)$	couplage maximum [122, 74]
$k = 3$	$O(n^2)$	*ouvert*
$k \geq 4$	$O(n^2)$	$\mathcal{NP}$-difficile (même si parfaits ou de Helly) [14]

Fig. 6.2 – La complexité du problème Ordo pour les graphes d'arcs circulaires.

	Graphes de tolérances propres	Graphes de tolérances
$k = 2$	couplage maximum [122, 74]	couplage maximum [122, 74]
$k \geq 3$	*ouvert* (même si unitaires et bornées)	$\mathcal{NP}$-difficile (même si bornées)

Fig. 6.3 – La complexité du problème Ordo pour les graphes de tolérances.

	Graphes sans $K_{1,3}$	Graphes d'intervalles	Graphes d'arcs
$k \geq 2$	$O(n^2/k)$	$O(n+m)$	$O(n+m)$

FIG. 6.4 – La complexité du redécoupage pour les graphes sans $K_{1,3}$ et les graphes d'arcs.

	Graphes de tolérances propres	Graphes de tolérances
$k = 2$	$O(n+m)$	contre-exemple
$k \geq 3$	*ouvert* (même si unitaires et bornées)	contre-exemple

FIG. 6.5 – La complexité du redécoupage pour les graphes de tolérances.

	Forêts	Graphes scindés	Graphes triangulés
$k \leq 4$	$O(n+m)$	$O(n+m)$	$O(n+m)$
$k \geq 5$	$O(n+m)$	$O(n+m)$	*ouvert*

FIG. 6.6 – La complexité du redécoupage pour les graphes triangulés.

Une cartographie partielle de la complexité du problème est proposée figure 6.7. Bien que pour la majeure partie des classes de graphes étudiées le problème reste $\mathcal{NP}$-difficile lorsque k est fixé, quelques questions intéressantes restent ouvertes concernant la complexité du problème k-ORDO pour les graphes d'intervalles et les graphes de permutation, ainsi que pour d'autres classes qui les englobent. Comme nous l'avons évoqué en introduction, le problème ORDO peut être résolu en temps et espace polynomial pour les compléments de graphes d'intervalles et même les compléments de graphes fortement triangulés. Or, la question de la complexité du problème ORDO pour les *compléments des graphes triangulés* restent ouvertes, de même que pour les *compléments des graphes d'arcs*. Enfin, l'étude concernant la propriété de redécoupage que nous avons introduite au chapitre 4 est encourageante et mérite d'être approfondie.

6.2 Perspectives

Voici deux autres problèmes que nous avons extraits de la problématique de planification de personnel rencontrée lors de notre séjour au sein de la firme PROLOGIA du Groupe Air Liquide.

Le premier problème se définit comme suit. Soit $\{T_i\}_{i=1,\ldots,n}$ un ensemble de tâches journalières à affecter à des employés, chacune possédant une date de début d_i et une date de fin f_i. Cette fois-ci, un employé ne peut effectuer un ensemble de tâches (deux-à-deux disjointes dans le temps) que si la somme des durées de celles-ci est inférieure à une valeur donnée D, la question restant : combien d'employés doit-on mobiliser pour qu'à la fin de la journée toutes les tâches aient été réalisées ? Ce problème est équivalent à un problème de remplissage de boîtes (en anglais *bin-packing*) avec certaines contraintes de compatibilité entre les objets (ici définies par le graphe d'intervalles sous-jacent), et par conséquent est $\mathcal{NP}$-difficile

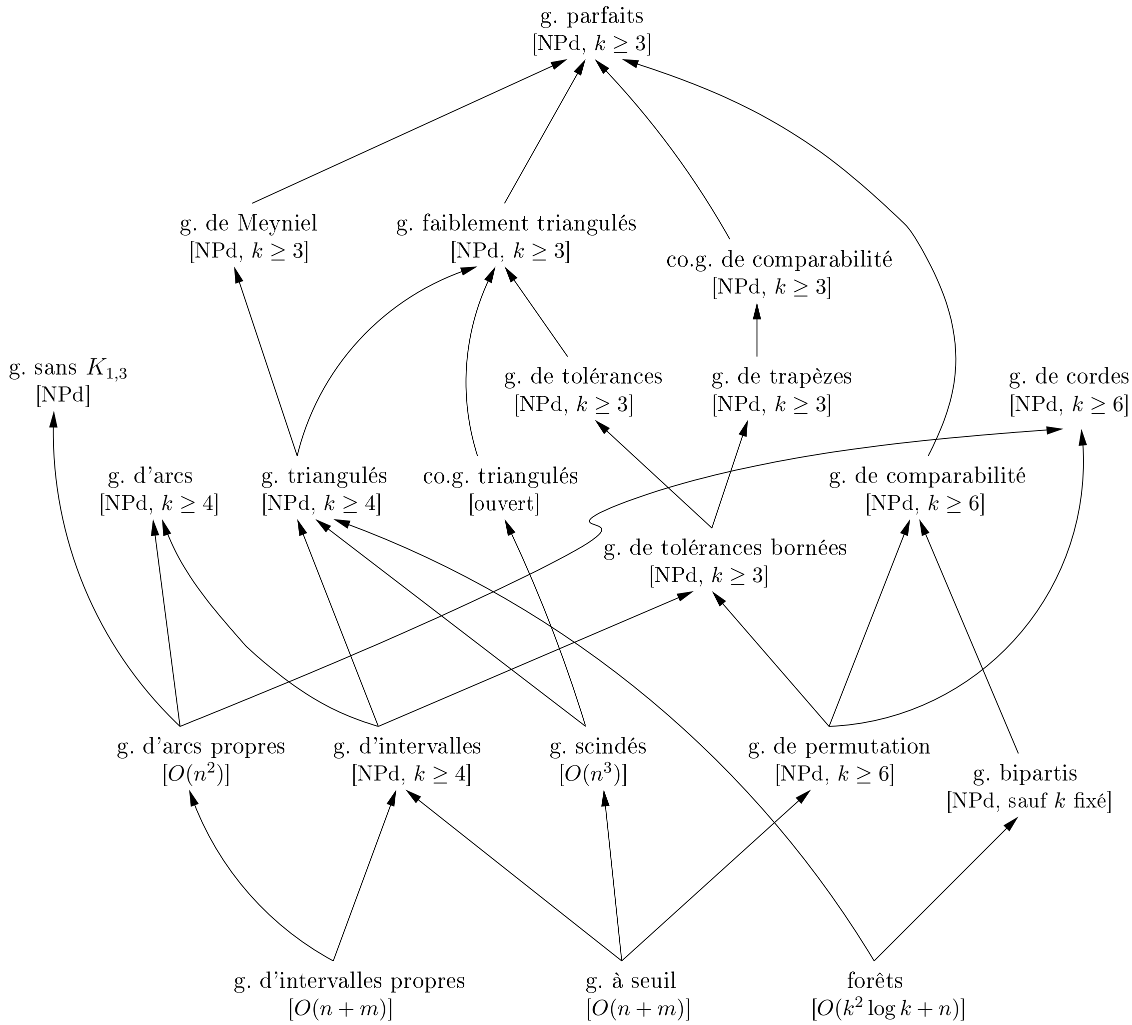


FIG. 6.7 – Une cartographie de la complexité du problème ORDO.

(même lorsque les intervalles sont tous deux-à-deux disjoints). Dans notre mémoire de DEA [62], nous avons proposé des algorithmes d'approximation efficaces pour ce problème. Le problème du *bin-packing* avec conflits a été étudié plus généralement par Jansen et Öhring [101, 99] (voir aussi [94, 95, 96]). Cette étude mériterait d'être détaillée et enrichie, comme nous l'avons fait pour le problème d'ordonnancement avec exclusion mutuelle.

Le second problème diffère du précédent par le fait que le temps de latence entre les différentes tâches de la vacation d'un employé est comptabilisé dans le calcul de son temps de travail. La contrainte devient : un employé ne peut effectuer un ensemble de tâches (deux-à-deux disjointes dans le temps) que si le temps écoulé entre le début de la première tâche et la fin de la dernière est inférieur à une valeur donnée D. Le problème se réduit alors à la coloration minimum d'un graphe $G = (V, E_i \cup E_c)$ où les arêtes de l'ensemble E_i induisent un graphe d'intervalles et les arêtes de l'ensemble E_c induisent un graphe de comparabilité. À notre connaissance, aucun résultat n'est paru au sujet de ce problème.

Pour conclure, mentionnons le problème de l'ordonnancement par lots (en anglais *batch-scheduling*) avec contraintes de compatibilité (ou d'incompatibilité) entre les tâches. Ce problème, récemment étudié par Boudhar [19] et Finke *et al.* [53], étend le problème de l'ordonnancement avec exclusion mutuelle de la façon suivante. Chaque sommet $v \in V$ du graphe G est pondéré par un poids $p(v) \in \mathbb{N}$ et le poids d'un stable de G est défini comme étant le maximum parmi les poids de ses différents sommets. Le but est alors de trouver une partition de G en stables de taille au plus k qui minimise la somme des poids des stables. Le problème est déjà $\mathcal{NP}$-difficile pour les graphes scindés, même lorsque k est un paramètre fixé supérieur ou égal à trois [19]. Toutefois, plusieurs questions restent ouvertes le concernant [53].

Bibliographie

[1] N. Alon (1983). A note on decomposition of graphs into isomorphic matchings. *Acta Mathematica Hungarica* 42, pp. 221–223.

[2] H. Alt, N. Blum, K. Mehlhorn et M. Paul (1991). Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$. *Information Processing Letters* 37, pp. 237–240.

[3] M.G. Andrews, M.J. Atallah, D.Z. Chen et D.T. Lee (2000). Parallel algorithms for maximum matching in complements of interval graphs and related problems. *Algorithmica* 26, pp. 263–289.

[4] A. Apostolico, M.J. Atallah et S.E. Hambrusch (1992). New clique and independent set algorithms for cicle graphs. *Discrete Applied Mathematics* 36(1), pp. 1–24.

[5] M.J. Atallah, M.T. Goodrich et S.R. Kosaraju (1994). On the parallel complexity of evaluating some sequences of set manipulation operations. *Journal of the ACM* 41, pp. 1049–1088.

[6] B.S. Baker et E.G. Coffman, Jr. (1996). Mutual exclusion scheduling. *Theoretical Computer Science* 162, pp. 225–243.

[7] Logiciel Bamboo, édité par Prologia - Groupe Air Liquide, Parc Scientifique et Technologique de Luminy, Marseille, France.
`http://prologia.fr/bamboo/bamboo.html`

[8] B. Beauquier, J.-C. Bermond, L. Gargano, P. Hell, S. Perennes et U. Vaccaro (1997). Graph problems arising from wavelength-routing in all-optical networks. In *Proceedings of the 2nd Workshop on Optics and Computer Science.* (WOCS'97, Genève, Suisse)
`http://www-sop.inria.fr/sloop/personnel/Jean-Claude.Bermond/`
`http://www.dia.unisa.it/~lg/`

[9] C. Berge (1983). *Graphes.* Gautier-Villars, Paris, France. (troisième édition française)

[10] B.K. Bhattacharya, P. Hell et J. Huang (1996). A linear algorithm for maximum weight cliques in proper circular arc graphs. *SIAM Journal on Discrete Mathematics* 9(2), pp. 274–289.

[11] B.K. Bhattacharya et D. Kaller (1997). An $O(m + n \log n)$ algorithm for the maximum-clique problem in circular-arc graphs. *Journal of Algorithms* 25(2), pp. 336–358.

[12] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt et J. Weglarz (2001). *Scheduling Computer and Manufacturing Processes.* Springer-Verlag, Berlin, Allemagne. (deuxième édition)

[13] H.L. Bodlaender et F.V. Fomin (2004). Equitable colorings of bounded treewidth graphs. In *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science* (sous la direction de J. Fiala, V. Koubek et J. Kratochvíl), *LNCS* 3153, pp. 180–190. Springer-Verlag, Berlin, Allemagne.

[14] H.L. Bodlaender et K. Jansen (1995). Restrictions of graph partition problems. Part I. *Theoretical Computer Science* 148, pp. 93–109.

[15] H.L. Bodlaender, K. Jansen et G.J. Woeginger (1994). Scheduling with incompatible jobs. *Discrete Applied Mathematics* 55(3), pp. 219–232.

[16] K.P. Bogart, P.C. Fishburn, G. Isaak et L.J. Langley (1995). Proper and unit tolerance graphs. *Discrete Applied Mathematics* 60, pp. 99–117.

[17] K.P. Bogart et D.B. West (1999). A short proof that "proper = unit". *Discrete Mathematics* 201, pp. 21–23.

[18] K.S. Booth et G.S. Lueker (1976). Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences* 13, pp. 335–379.

[19] M. Boudhar (2001). Static scheduling on a single batch processing machine with split compatibility graphs. Cahier n° 28. Laboratoire Leibniz (IMAG), Grenoble, France. `http://www-leibniz.imag.fr/LesCahiers/`

[20] A. Brandstädt (1996). Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics* 152, pp. 47–54. (*Corrigendum* dans *Discrete Mathematics* 186, p. 295)

[21] A. Brandstädt (1998). The complexity of some problems related to 3-colorability. *Discrete Applied Mathematics* 89, pp. 59–73.

[22] A. Brandstädt, V.B. Le et J.P. Spinrad (1999). *Graph Classes : A Survey.* SIAM Monographs on Discrete Mathematics and Applications 3, SIAM Publications, Philadelphie, PA.

[23] K. Bryś et Z. Lonc (1998). Clique and anticlique partitions of graphs. *Discrete Mathematics* 185, pp. 41–49.

[24] M. Chudnovsky, N. Robertson, P.D. Seymour et N. Thomas (2002). The strong perfect graph theorem. (manuscript)
`http://www.math.gatech.edu/~thomas/spgc.html`

[25] M. Chudnovsky, N. Robertson, P.D. Seymour et N. Thomas (2003). Progress on perfect graphs. *Mathematical Programming Series B* 97, pp. 405–422.

[26] E. Cohen et M. Tarsi (1991). $\mathcal{NP}$-completeness of graph decomposition problem. *Journal of Complexity* 7, pp. 200–212.

[27] R.M. Corless, G.H. Gonnet, D.E.G. Hare, D.J. Jeffrey et D.E. Knuth (1996). On the Lambert W function. *Advances in Computational Mathematics* 5, pp. 329–359.

[28] T. Cormen, C. Leiserson et R. Rivest (1994). *Introduction à l'Algorithmique.* Dunod, Paris, France. (édition française)

[29] D.G. Corneil (1985). The complexity of generalized clique packing. *Discrete Applied Mathematics* 12, pp. 233–239.

[30] D.G. Corneil (2004). A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs. *Discrete Applied Mathematics* 138, pp. 371–379.

[31] D.G. Corneil, H. Kim, S. Natarajan, S. Olariu et A. Sprague (1995). Simple linear time recognition of unit interval graphs. *Information Processing Letters* 55, pp. 99–104.

[32] D.G. Corneil, S. Olariu et L. Stewart (1998). The ultimate interval graph recognition algorithm ? In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 175–180. ACM Press, New York, NY.

[33] D.G. Corneil, Y. Perl et L. Stewart (1985). A linear recognition algorithm for cographs. *SIAM Journal on Computing* 4, pp. 926–934.

[34] G. Cornuéjols (2002). The strong perfect graph conjecture. In *Proceedings of the International Congress of Mathematicians*, Vol. 3, pp. 547–559. Higher Education Press of China, Pékin (Beijing), Chine. `http://integer.gsia.cmu.edu/webpub/SPGCsurvey.pdf`

[35] G. Cornuéjols, X. Liu et K. Vušković (2003). A polynomial algorithm for recognizing perfect graphs. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pp. 20–27. IEEE Computer Society Publications, Los Alamitos, CA.

[36] E. Dahlhaus et M. Karpinski (1998). Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics* 84, pp. 79–91.

[37] E. Dekel et S. Sahni (1984). A parallel matching algorithm for convex bipartite graphs and applications to scheduling. *Journal of Parallel Distributed Computing* 1, pp. 185–205.

[38] C.M.H. De Figueiredo, J. Meidanis et C.P. De Mello (1995). A linear-time algorithm for proper interval graph recognition. *Information Processing Letters* 56(3), pp. 179–184.

[39] X. Deng, P. Hell et J. Huang (1996). Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM Journal on Computing* 25(2), pp. 390–403.

[40] D. De Werra (1978). On line perfect graphs. *Mathematical Programming* 15, pp. 236–238.

[41] D. De Werra (1985). An introduction to timetabling. *European Journal of Operational Research* 19, pp. 151–162.

[42] D. De Werra (1985). Some uses of hypergraphs in timetabling. *Asia-Pacific Journal of Operational Research* 2(1), pp. 2–12.

[43] D. De Werra (1996). Extensions of coloring models for scheduling purposes. *European Journal of Operational Research* 92, pp. 474–492.

[44] D. De Werra (1997). Restricted coloring models for timetabling. *Discrete Mathematics* 165/166, pp. 161–170.

[45] D. De Werra, A.J. Hoffman, N.R.V. Mahadev et U.N. Peled (1996). Restrictions and preassignments in preemptive open shop scheduling. *Discrete Applied Mathematics* 68, pp. 169–188.

[46] R. Diestel (1991). *Graph Theory.* Graduate Texts in Mathematics 173, Springer-Verlag, New York, NY. (deuxième édition)

[47] G. Durán, A. Gravano, R.M. McConnel, J. Spinrad et A. Tucker. Polynomial time algorithm recognition of unit circular-arc graphs. (à paraître dans *Journal of Algorithms*)

[48] J. Edmonds (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics* 17, pp. 449–467.

[49] R. Faudree, E. Flandrin et Z. Ryjàček (1997). Claw-free graphs – A survey. In *Proceedings of the 2nd Krakow Conference of Graph Theory* (sous la direction de A. Rycerz, A.P. Wojda et M. Wozniak), *Discrete Mathematics* 164, pp. 87–147.

[50] T. Feder, P. Hell, S. Klein et R. Motwani (1999). The complexity of graph partition problems. In *Proceedings of the 31rd Annual ACM Symposium on Theory of Computing* (sous la direction de F.W. Thatcher et R.E. Miller), pp. 464–472. Plenum Press, New York, NY.

[51] T. Feder et R. Motwani (1991). Clique partitions, graph compression and speeding up algorithms. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pp. 123–133. ACM Press, New York, NY.

[52] S. Felsner, R. Müller et L. Wernisch (1997). Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Applied Mathematics* 74(1), pp. 13–32.

[53] G. Finke, V. Jost, M. Queyranne et A. Sebö (2004). Batch processing with interval graph compatibilities between tasks. Cahier n° 108. Laboratoire Leibniz (IMAG), Grenoble, France. `http://www-leibniz.imag.fr/LesCahiers/`

[54] P.C. Fishburn (1985). *Interval Orders and Interval Graphs.* Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, New York, NY.

[55] G.N. Frederickson (1983). Scheduling unit-time tasks with integer release times and deadlines. *Information Processing Letters* 16, pp. 171–173.

[56] D.R. Fulkerson et O.A. Gross (1965). Incidence matrices and interval graphs. *Pacific Journal of Mathematics* 15, pp. 835–855.

[57] H.N. Gabow et R.E. Tarjan (1985). A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences* 30, pp. 209–221.

[58] G. Gallo (1984). An $O(n \log n)$ algorithm for the convex bipartite matching problem. *Operations Research Letters* 3(1), pp. 31–34.

[59] M.R. Garey et D.S. Johnson (1976). Some simplified $\mathcal{NP}$-complete graph problems. *Theoretical Computer Science* 1(3), pp. 237–267.

[60] M.R. Garey et D.S. Johnson (1979). *Computer and Intractability : A Guide to the Theory of $\mathcal{NP}$-Completeness*. W.H. Freeman & Co, San Francisco, CA.

[61] M.R. Garey, D.S. Johnson, G.L. Miller et C.H. Papadimitriou (1980). The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic and Discrete Methods* 1(2), pp. 216–227.

[62] F. Gardi (2001). Sur certains problèmes liés à la planification d'horaires de travail. Mémoire de DEA. Laboratoire d'Informatique Fondamentale (LIF, Université de la Méditerranée – Aix-Marseille II), Marseille, France.
`http://www.dil.univ-mrs.fr/dea/dea2001/memoires.html`

[63] F. Gardi (2003). A note on the Roberts characterization of proper and unit interval graphs. Rapport de Recherche n° 11-2003. Laboratoire d'Informatique Fondamentale (LIF, Université de la Méditerranée – Aix-Marseille II), Marseille, France. (soumis à *Discrete Mathematics*, Novembre 2002, en cours de révision)
`http://lif-sud.univ-mrs.fr/Rapports/11-2003-Gardi.html`

[64] F. Gardi (2003). Planification d'horaires de travail et théorie des graphes. In *Actes du 5ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*, pp. 99–100. Laboratoire d'Informatique d'Avignon (LIA, Université d'Avignon et des Pays de Vaucluse), Avignon, France.

[65] F. Gardi (2003). Efficient algorithms for disjoint matchings among intervals and related problems. In *Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science* (sous la direction de C.S. Calude, M.J. Dinneen et V. Vajnovszki), *LNCS* 2731, pp. 168–180. Springer-Verlag, Berlin, Allemagne.

[66] F. Gardi (2003). Planification d'horaires de travail et colorations de graphes. In *Actes des Journées Graphes, Réseaux et Modélisation*. Laboratoire d'Infomatique Algorithmique, Fondements et Applications (LIAFA, Université Denis Diderot – Paris VII), Paris, France. `http://www.liafa.jussieu.fr/~grm/journees/`

[67] F. Gardi (2004). On partitioning interval and circular-arc graphs into proper interval subgraphs with applications. In *Proceedings of the 6th Latin American Symposium on Theoretical Informatics* (sous la direction de M. Farach-Colton), *LNCS* 2976,

pp. 129–140. Springer-Verlag, Berlin, Allemagne.

[68] F. Gardi (2004). A sufficient condition for optimality in mutual exclusion scheduling for interval graphs and related classes. In *Abstracts of the 9th International Workshop on Project Management and Scheduling* (sous la direction de A. Oulamara et M.-C. Portmann), pp. 154–157. Laboratoire Loria (INRIA Lorraine), Nancy, France.

[69] L. Gargano et A.A. Rescigno (2000). Coloring circular arcs with applications. In *Proceedings of ICALP Satellite Workshops* (sous la direction de J.D.P. Rolim, A.Z. Broder, A. Corradini, R. Gorrieri, R. Heckel, J. Hromkovic, U. Vaccaro et J.B. Wells), Workshop on Approximation and Randomized Algorithms in Communication Networks, pp. 155–166. Carleton Scientific, Waterloo, Ontario, Canada.

[70] F. Gavril (1972). Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing* 1(2), pp. 180–187.

[71] F. Gavril (1996). Intersection graphs of Helly families of subtrees. *Discrete Applied Mathematics* 66, pp. 45–56.

[72] P.C. Gilmore et A.J. Hoffman (1964). A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics* 16, pp. 539–548.

[73] F. Glover (1967). Maximum matchings in a convex bipartite graph. *Naval Research Logistics Quarterly* 14(3), pp. 313–316.

[74] A.V. Golberg et A.V. Karzanov (1995). Maximum skew-symmetric flows. In *Proceedings of the Third Annual European Symposium on Algorithms* (sous la direction de P.G. Spirakis), *LNCS* 2385, pp. 155–170. Springer-Verlag, Berlin, Allemagne.

[75] M.C. Golumbic (1980). *Algorithmic Graph Theory and Perfect Graphs.* Computer Science and Applied Mathematics Series, Academic Press, New York, NY.

[76] M.C. Golumbic, C.L. Monma et W.T. Trotter, Jr. (2004). Tolerance Graphs. *Discrete Applied Mathematics* 9(2), pp. 157–170.

[77] M.C. Golumbic et A. Siani (2002). Coloring algorithms for tolerance graphs : reasoning and scheduling with interval constraints. In *Proceedings of Artificial Intelligence, Automated Reasoning, and Symbolic Computation, Joint International Conferences* (sous la direction de J. Calmet, B. Benhamou, O. Caprotti, L. Henocque et V. Sorge), *LNCS* 2385, pp. 196–207. Springer-Verlag, Berlin, Allemagne.

[78] M.C. Golumbic et A.N. Trenk (2004). *Tolerance Graphs.* Cambridge Studies in Advanced Mathematics 89, Cambridge University Press, Cambridge, Angleterre, Royaume-Uni.

[79] M. Grötschel, L. Lovász et A. Schrijver (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1, pp. 169–197.

[80] U.I. Gupta, D.T. Lee et J.Y.-T. Leung (1979). An optimal solution for the channel-assignment problem. *IEEE Transactions on Computers* C-28(11), pp. 459–467.

[81] U.I. Gupta, D.T. Lee et J.Y.-T. Leung (1982). Efficient algorithms for interval graphs and circular-arc graphs. *Networks* 12, pp. 459–467.

[82] M. Habib, R.M. McConnel, C. Paul et L. Viennot (2000). Lex-BSF and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science* 234, pp. 59–84.

[83] P. Hansen, A. Hertz et J. Kuplinski (1993). Bounded vertex colorings of graphs. *Discrete Mathematics* 111, pp. 305–312.

[84] R. Hayward (1985). Weakly triangulated graphs. *Journal of Combinatorial Theory Series B* 39, pp. 200–209.

[85] R. Hayward, C.T. Hoàng et F. Maffray (1989). Optimizing weakly triangulated graphs. *Graphs and Combinatorics* 5, pp. 339–349. (*Erratum* dans *Graphs and Combinatorics* 6, pp. 33–35)

[86] P. Hell, S. Klein, L.T. Nogueira et F. Protti (2004). Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics* 141, pp. 185–194.

[87] P. Hell, S. Klein, L.T. Nogueira et F. Protti. Packing r-cliques in weighted chordal graphs. (à paraître dans *Annals of Operations Research*).

[88] D.S. Hochbaum (1997). *Approximation Algorithms for $\mathcal{NP}$-Hard Problems* (sous la direction de D.S. Hochbaum). PWS Publishing Company, Boston, MA.

[89] J.E. Hopcroft et R.M. Karp (1973). An $O(n^{5/2})$ algorithm for maximum matching in bipartite graphs. *SIAM Journal on Computing* 2, pp. 225–231.

[90] I.J. Holyer (1981). The $\mathcal{NP}$-completeness of edge colorings. *SIAM Journal on Computing* 10, pp. 718–720.

[91] W.-L. Hsu (1981). How to color claw-free perfect graphs. *Annals of Discrete Mathematics* 11, pp. 189–197.

[92] W.-L. Hsu et G.L. Nemhauser (1981). Algorithms for maximum weight cliques, minimum weighted clique covers and minimum colorings of claw-free perfect graphs. *Annals of Discrete Mathematics* 21, pp. 357–369.

[93] W.-L. Hsu et K.-H. Tsai (1991). Linear time algorithms on circular-arc graphs. *Information Processing Letters* 40, pp. 123–129.

[94] S. Irani et V.J. Leung (1996). Scheduling with conflicts, and applications to traffic signal control. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 85–94. ACM Press, New York, NY.

[95] S. Irani et V.J. Leung (1997). Probabilistic analysis for scheduling with conflicts. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 286–295. ACM Press, New York, NY.

[96] S. Irani et V.J. Leung (2003). Scheduling with conflicts on bipartite and interval graphs. *Journal of Scheduling* 6(3), pp. 287–307.

[97] M.S. Jacobson, F.R. McMorris et H.M. Mulder (1991) An introduction to tolerance intersection graphs. In *Graph Theory, Combinatorics and Applications* (sous la direction de Y. Alavi, G. Chartrand, O.R. Oellerman et A.J. Schwenk), Vol. 2, pp. 705–723. John Wiley & Sons, New York, NY.

[98] K. Jansen (1993). Scheduling with constrained processor allocation for interval orders. *Computers and Operations Research* 20(6), pp. 587–595.

[99] K. Jansen (1999). An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization* 3(4), pp. 363–377.

[100] K. Jansen (2003). The mutual exclusion scheduling problem for permutation and comparability graphs. *Information and Computation* 180(2), pp. 71–81.

[101] K. Jansen et S. Öhring (1997). Approximation algorithms for time constrained scheduling. *Information and Computation* 132(2), pp. 85–108.

[102] M. Jarvis et B. Zhou (2001). Bounded vertex coloring of trees. *Discrete Mathematics* 232, pp. 145–151.

[103] D. Kaller, A. Gupta et T.C. Shermer (1995). The χ_t-coloring problem. In *Proceedings of the 12th Annual Symposium on the Theoretical Aspects of Computer*

Science (sous la direction de E.W. Mayr et C. Puech), *LNCS* 900, pp. 409–420. Springer-Verlag, Berlin, Allemagne.

[104] R.M. Karp (1972). Reducibilty among combinatorial problems. In *Complexity of Computer Computations* (sous la direction de R.E. Miller et J.W. Thatcher), pp. 85–103. Plenum Press, New York, NY.

[105] D.E. Knuth (1998). *The Art of Computer Programming, Volume 3 : Sorting and Searching*. Addison-Wesley, Reading, MA. (deuxième édition)

[106] J. Krarup et D. De Werra (1982). Chromatic optimization : limitations, objectives, uses, references. *European Journal of Operational Research* 11, pp. 1–19.

[107] M. Kubale (1993). Interval edge coloring of a graph with forbidden colors. *Discrete Mathematics* 121, pp. 135–143.

[108] R. Leibowitz (1978). *Interval counts and threshold graphs*. Thèse de Doctorat. Rutgers University, New Brunswick, NJ.

[109] B. Levêque et F. Maffray (2004). Coloring Meyniel graphs in linear time. Cahier n° 105. Laboratoire Leibniz (IMAG), Grenoble, France. `http://www-leibniz.imag.fr/LesCahiers/`

[110] Y.D. Liang et N. Blum (1995). Circular convex bipartite graphs : maximum matching and hamiltonian circuits. *Information Processing Letters* 56, pp. 215–219.

[111] Y.D. Liang et C. Rhee (1993). Finding a maximum matching in a circular-arc graph. *Information Processing Letters* 45, pp. 185–190.

[112] W. Lipski, Jr. et F.P. Preparata (1981). Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems. *Acta Informatica* 15, pp. 329–346.

[113] Z. Lonc (1991). On complexity of some chain and antichain partition problems. In *Proceedings of the 17th International Workshop on Graph–Theoretic Concepts in Computer Science* (sous la direction de G. Schmidt et R. Berghammer), *LNCS* 570, pp. 97–104. Springer-Verlag, Berlin, Allemagne.

[114] P.J. Looges et S. Olariu (1993). Optimal greedy algorithms for indifference graphs. *Computers and Mathematics with Applications* 25, pp. 15–25.

[115] R.D. Luce (1956). Semiorders and a theory of utility discrimination. *Econometrica* 24, pp. 178–191.

[116] F. Maffray (1992). Kernels in perfect line-graphs. *Journal of Combinatorial Theory Series B* 55, pp. 1–8.

[117] G.K. Manacher et T.A. Mankus (1997). Finding a maximum clique in a set of proper circular arcs in time $O(n)$ with applications. *International Journal of Foundations of Computer Science* 8(4), pp. 443–467.

[118] D. Marx (2003). A short proof of the $\mathcal{NP}$-completeness of circular arc coloring. (manuscript) `http://www.cs.bme.hu/~dmarx/papers/circularNP.pdf`

[119] J.F. Maurras (2002). *Programmation Linéaire, Complexité : Séparation et Optimisation*. Mathématiques et Applications 38, Springer-Verlag, Berlin, Allemagne.

[120] R.M. McConnel (2003). Linear time recognition of circular-arc graphs. *Algorithmica* 37(2), pp. 93–147.

[121] R.M. McConnel et J. Spinrad (1997). Linear-time transitive orientation. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 19–25. ACM Press, New York, NY.

[122] S. Micali et V.V. Vazirani (1980). An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science*, pp. 17–27. IEEE Computer Society Publications, Los Alamitos, CA.

[123] G.J. Minty (1980). On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory Series B* 28, pp. 284–304.

[124] S. Olariu (1991). An optimal greedy heuristic to color interval graphs. *Information Processing Letters* 37, pp. 21–25.

[125] B.S. Panda et S.K. Das (2003). A linear time recognition algorithm for proper interval graphs. *Information Processing Letters* 87(3), pp. 153–161.

[126] C.H. Papadimitriou et M. Yannakakis (1979). Scheduling interval-ordered tasks. *SIAM Journal on Computing* 8, pp. 405–409.

[127] G. Ramalingam et C. Pandu Rangan (1988). A unified approach to domination problems on interval graphs. *Information Processing Letters* 27(5), pp. 271–274.

[128] J.L. Ramírez-Alfonsín et B.A. Reed (2001). *Perfect Graphs* (sous la direction de J.L. Ramírez-Alfonsín et B.A. Reed). Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester, Angleterre, Royaume-Uni.

[129] C. Rhee et Y.D. Liang (1995). Finding a maximum matching in a permutation graph. *Acta Informatica* 32, pp. 779–792.

[130] F.S. Roberts (1968). *Representations of indifference relations.* Thèse de Doctorat. Stanford University, Stanford, CA.

[131] F.S. Roberts (1969). Indifference graphs. In *Proof Techniques in Graph Theory* (sous la direction de F. Harary), pp. 139–146. Academic Press, New York, NY.

[132] F.S. Roberts (1976). *Discrete Mathematical Models with Applications to Social, Biological, and Environmental Problems.* Prentice-Hall, Englewood Cliffs, NJ.

[133] F.S. Roberts (1978). *Graph Theory and its Application to the Problems of Society.* CBMS-NSF Regional Conference Series in Applied Mathematics 29, SIAM Publications, Philadelphie, PA.

[134] F. Roussel et I. Rusu (1999). Holes and dominoes in Meyniel graphs. *International Journal of Foundations of Computer Science* 10, pp. 127–146.

[135] N. Sbihi (1980). Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. *Discrete Mathematics* 29, pp. 53–76.

[136] A. Schrijver (1986). *Theory of Linear and Integer Programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester, Angleterre, Royaume-Uni.

[137] A. Schrijver (2003). *Combinatorial Optimization : Polyhedra and Efficiency.* Algorithms and Combinatorics 24, Vol. A, Springer-Verlag, Berlin, Allemagne.

[138] D.S. Scott et P. Suppes (1958). Foundation aspects of theories of measurement. *Journal of Symbolic Logic* 23, pp. 113–128.

[139] M. Scutella et G. Scevola (1988). A modification of Lipski-Preparata's algorithm for the maximum matching problem on bipartite convex graphs. *Ricerca Operativa* 46, pp. 63–77.

[140] D. Seinsche (1974). On a property of the class of n-colorable graphs. *Journal of Combinatorial Theory Series B* 16, pp. 191–193.

[141] W.-K. Shih et W.-L. Hsu (1985). An $O(n^{1.5})$ algorithm to color proper circular-arc graphs. *Discrete Applied Mathematics* 25(3), pp. 321–323.

[142] J.P. Spinrad (1987). On comparability and permutation graphs. *SIAM Journal on Computing* 14, pp. 658–670.

[143] J.P. Spinrad (1994). Recognition of circle graphs. *Journal of Algorithms* 16(2), pp. 264–282.

[144] J.P. Spinrad et N. Sritharan (1995). Algorithms for weakly triangulated graphs. *Discrete Applied Mathematics* 19, pp. 181–191.

[145] G. Steiner et J.S. Yeomans (1993). Level schedules for mixed-model, just-in-time processes. *Management Science* 39(6), pp. 728–735.

[146] G. Steiner et J.S. Yeomans (1993). A note on "Scheduling unit-time tasks with integer release times and deadlines". *Information Processing Letters* 47, pp. 165–166.

[147] G. Steiner et J.S. Yeomans (1996). A linear time algorithm for maximum matchings in convex, bipartite graphs. *Computers and Mathematics with Applications* 31(12), pp. 91–96.

[148] R.E. Tarjan (1975). Efficiency of a good but not linear set union algorithm. *Journal of the ACM* 22, pp. 215–225.

[149] R.E. Tarjan (1983). *Data Structures and Network Algorithms.* CBMS-NSF Regional Conference Series in Applied Mathematics 44, SIAM Publications, Philadelphie, PA.

[150] L.E. Trotter (1977). Line perfect graphs. *Mathematical Programming* 12, pp. 255–259.

[151] A. Tucker (1974). Structure theorems for some circular-arc graphs. *Discrete Mathematics* 7, pp. 167–195.

[152] A. Tucker (1975). Coloring a family of circular arcs. *SIAM Journal on Applied Mathematics* 29, pp. 493–502.

[153] W. Unger (1988). On the k-coloring of circle graphs. In *Proceedings of the 5th Annual Symposium on the Theoretical Aspects of Computer Science* (sous la direction de R. Cori et M. Wirsing), *LNCS* 294, pp. 61–72. Springer-Verlag, Berlin, Allemagne.

[154] W. Unger (1992). The complexity of colouring circle graphs (extended abstract). In *Proceedings of the 9th Annual Symposium on the Theoretical Aspects of Computer Science* (sous la direction de A. Finkel et M. Jantzen), *LNCS* 577, pp. 389–400. Springer-Verlag, Berlin, Allemagne.

[155] M. Valencia-Pabon (2003). Revisiting Tuker's algorithm to color circular-arc graphs. *SIAM Journal on Computing* 32(4), pp. 1067–1072.

[156] P. Van Emde Boas (1977). Preserving order in a forest in less than logarithmic time and linear space. *Information Processing Letters* 6, pp. 80–82.

[157] G. Wegner (1967). *Eigenschaften der nerven homologische-einfacher familien in* $\mathbb{R}^n$. Thèse de Doctorat. Georg-August-Göttingen Universität, Göttingen, Allemagne.

[158] M.S. Yu et C.H. Yang (1993). An $O(n)$ time algorithm for maximum matching on cographs. *Information Processing Letters* 47, pp. 89–93.

Laboratoire d'Informatique Fondamentale
(CNRS UMR 6166)
Équipe Combinatoire et Recherche Opérationnelle
Parc Scientifique et Technologique de Luminy
case 901, 163 avenue de Luminy
13288 Marseille Cedex 9, France

frederic.gardi@lif.univ-mrs.fr

Prologia – Groupe Air Liquide
Parc Scientifique et Technologique de Luminy
case 919, bâtiment CCIMP
13288 Marseille Cedex 9, France

frederic.gardi@prologia.fr

Faculté des Sciences de Luminy
Université de la Méditerranée – Aix-Marseille II
Parc Scientifique et Technologique de Luminy
case 901, 163 avenue de Luminy
13288 Marseille Cedex 9, France

École Doctorale de Mathématiques et Informatique de Marseille
(École Doctorale n° 184)
Centre de Mathématiques et Informatique
39, rue F. Joliot Curie
13453 Marseille Cedex 13, France

Résumé :

Le *problème d'ordonnancement avec exclusion mutuelle* peut être formulé comme suit en les termes de la théorie des graphes : étant donné un graphe G non orienté et un entier k, déterminer une coloration minimum de G tel que chaque couleur apparaisse au plus k fois. Lorsque le graphe G est un *graphe d'intervalles*, ce problème a des applications dans le domaine de la *planification de personnel*. Ainsi, cette thèse a pour objet l'étude détaillée de la complexité du problème d'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles ou de classe apparentée.

Les deux premiers chapitres de la thèse traitent de la complexité du problème pour les graphes d'intervalles, ainsi que deux extensions, les graphes d'arcs circulaires et les graphes de tolérances. Lorsque le problème s'avère être polynomial, nous proposons des algorithmes à la fois simples et efficaces pour le résoudre (notamment des algorithmes en temps et espace linéaire).

Motivé par des aspects pratiques, nous analysons ensuite l'impact de la propriété suivante sur la complexité du problème : le graphe G admet une coloration où chaque couleur apparaît au moins k fois. Nous montrons que si un graphe d'intervalles satisfait cette condition, alors celui-ci admet une partition optimale en $\lceil n/k \rceil$ stables de taille au plus k qui peut être calculée en temps et espace linéaire. Cette assertion est ensuite étendue aux graphes sans $K_{1,3}$, aux graphes d'arcs circulaires, ainsi qu'aux graphes triangulés pour $k \leq 4$.

Le dernier chapitre est consacré à certains problèmes de partition relatifs aux graphes d'intervalles. Nous y étudions en particulier le problème de la partition d'un graphe d'intervalles ou d'arcs en sous-graphes d'intervalles propres, pour lequel nous établissons la proposition suivante : tout graphe d'intervalles admet une partition en moins de $\lceil \log_3 n \rceil$ sous-graphes d'intervalles propres et celle-ci peut être calculée en temps $O(n \log n + m)$ et espace linéaire. Ce résultat est mis à profit lors de la conception d'algorithmes polynomiaux d'approximation pour un problème de planification de personnel lié à l'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles ou d'arcs.

Mots-clefs : ordonnancement avec exclusion mutuelle, coloration de graphes, planification de personnel, graphes d'intervalles, classes de graphes.

Abstract :

The *mutual exclusion scheduling problem* can be formulated as follows in graph-theoretic terms : given an undirected graph G and an integer k, determine a minimum coloring of G such that each color is used at most k times. When the graph G is an *interval graph*, this problem has some applications in *workforce planning*. Then, the subject of this thesis is to detail the complexity of the mutual exclusion scheduling problem for interval graphs and related classes.

The first two chapters of the thesis deal with the complexity of the problem for interval graphs, as well as two extensions, circular-arc graphs and tolerance graphs. When the problem is proved to be polynomial, we propose some simple and efficient algorithms for its resolution (in particular linear time and space algorithms).

Motivated by practical aspects, we also analyse the impact of the following property on the complexity of the problem : the graph G admits a coloring such that each color appears at least k times. We show that if an interval graph satisfies this property, then it admits an optimal partition into $\lceil n/k \rceil$ stable sets of size at most k which can be computed in linear time and space. Then, this assertion is extended to $K_{1,3}$-free graphs, circular-arc graphs, as well as chordal graphs for $k \leq 4$.

The last chapter is devoted to partition problems related to interval graphs. We investigate particularly the problem of partitioning interval ou circular-arc graphs into proper interval subgraphs, for which the following proposition is established : any interval graph admits a partition into less than $\lceil \log_3 n \rceil$ proper interval graphs and this one can be computed in $O(n \log n + m)$ time and linear space. This result is used in the design of polynomial approximation algorithms for a workforce planning problem related to mutual exclusion scheduling for interval or circular-arc graphs.

Title : Mutual exclusion scheduling with interval graphs or related classes : complexity and algorithms.

Keywords : mutual exclusion scheduling, graph coloring, workforce planning, interval graphs, classes of graphs.