6

SUR CERTAINS PROBLÈMES LIÉS À LA PLANIFICATION D'HORAIRES DE TRAVAIL

Frédéric Gardi
Sous la direction de Michel Van Caneghem
Laboratoire Informatique de Marseille

Résumé Le problème de la planification d'horaires de travail consiste, de manière schématique, en l'affectation d'un ensemble de tâches à un ensemble d'agents selon des règles bien définies. Nous avons extrait de cette problématique complexe, trois problèmes formulés de manière plus simple : le problème MP_1 équivalent au problème de la partition d'un graphe d'intervalles par des stables de taille bornée et deux problèmes MP_2 et MP_3 pouvant être vus comme des extensions du problème de Bin-Packing.
Dans un premier chapitre, nous étudions la complexité du problème MP_1. Tout d'abord, nous établissons deux conditions suffisantes à sa polynômialité, à partir desquelles nous montrons que MP_1 est polynômial pour la classe des graphes d'intervalles propres. Ensuite, nous décrivons un nouvel algorithme en $O(n \log n)$ pour le problème du couplage parfait dans le complément d'un graphe d'intervalles. Enfin, à partir de ces résultats, nous formulons deux algorithmes d'approximation polynômiaux efficaces pour le cas où MP_1 est $\mathcal{NP}$-complet.
Dans un second chapitre, nous définissons et étudions la complexité du problème de Bin-Packing avec contraintes de compatibilité. Celles-ci étant définies par un graphe de compatibilité, nous établissons le lien étroit qui existe entre l'approximabilité des problèmes MP_2 et MP_3 et l'approximablité du problème de la coloration du complément de ce graphe de compatibilité. Enfin, nous décrivons plusieurs algorithmes d'approximation polynômiaux pour les problèmes MP_2 et MP_3 et donnons leurs facteurs d'approximation dans le pire des cas.

Introduction

La planification d'horaires de travail est un important problème de recherche opérationnelle, étudié depuis plusieurs décennies [10] ; il est d'autant plus d'actualité du fait de la réduction du temps de travail dans les entreprises (passage aux 35 heures). C'est un problème difficile, combinant à la fois des notions liées aux problèmes d'emplois du temps et d'ordonnancement. Sa formulation semble pourtant simple à priori : affecter du travail matérialisé sous forme de tâches, avec une date de début et une date de fin, à des agents, l'ensemble des tâches affectées aux agents formant ainsi des vacations. Mais la difficulté vient du fait que cette affectation du travail doit respecter certaines conditions et une règlementation bien établie (par exemple : tâches nécessitant certaines qualifications, durée maximum d'une journée de travail, pauses repas, repos nocturnes, etc). De plus, l'affectation doit généralement viser à minimiser un ou plusieurs objectifs (par exemple : minimiser le nombre d'heures supplémentaires effectuées par les agents, équilibrer le nombre d'heures de travail de l'ensemble des agents, etc).

Comme nous pouvons le constater, la modélisation du problème dans sa globalité devient, de par les besoins des entreprises et la législation du travail, extrêmement complexe. Il en est évidemment de même pour sa résolution. En effet, la lourde combinatoire cachée derrière le problème rend nécessaire des techniques de recherche opérationnelle élaborées pour le résoudre.

Ainsi, l'entreprise PROLOGIA - Groupe Air Liquide, qui offre une solution logicielle à ces problèmes de planification, les modélise par des problèmes de set-partitionning qu'elle traite par des techniques de programmation entière basées sur l'algorithme du simplexe et la génération de colonnes. La motivation de ce travail de recherche a donc été d'essayer d'aborder ces problèmes de planification de manière plus simple mais plus formelle, de manière à voir s'il était possible de les traiter correctement par des algorithmes purement combinatoires. Pour cela, nous avons donc tout d'abord modélisé le problème de différentes manières simples et épurées. Nous avons ainsi exhibé et étudié trois problèmes qui nous paraissaient intéressants.

Problème $\mathbf{MP_1}$:

Soit n tâches $T_i, i \in \{1, \ldots, n\}$ tel que $T_i = (d_i, f_i)$ avec $d_i \in \mathbb{N}$ date de début de T_i et $f_i \in \mathbb{N}$ date de fin de T_i. Partitionner ces n tâches en le minimum de vacations tel que les tâches appartenant à une vacation ne se chevauchent pas deux-à-deux et que chaque vacation ne contienne pas plus de k tâches, $k \in \mathbb{N}$.

Problème $\mathbf{MP_2}$:

Soit n tâches $T_i, i \in \{1, \ldots, n\}$ tel que $T_i = (d_i, f_i)$ avec $d_i \in \mathbb{N}$ date de début de T_i et $f_i \in \mathbb{N}$ date de fin de T_i. Partitionner ces n tâches en le minimum de vacations tel que les tâches appartenant à une vacation ne se chevauchent pas deux-à-deux et que la somme des durées des tâches de chaque vacation soit au plus égale à D, $D \in \mathbb{N}$.

Problème $\mathbf{MP_3}$:

Soit n tâches T_i, $i \in \{1, \ldots, n\}$ tel que $T_i = (d_i, f_i)$ avec $d_i \in \mathbb{N}$ date de début de T_i et $f_i \in \mathbb{N}$ date de fin de T_i. Partitionner ces n tâches en le minimum de vacations tel que les tâches appartenant à une vacation ne se chevauchent pas deux-à-deux, que chaque vacation ne contienne pas plus de k tâches, $k \in \mathbb{N}$ et que la somme des durées des tâches de chaque vacation soit au plus égale à D, $D \in \mathbb{N}$.

Les tâches pouvant être considérées comme des intervalles sur la droite, nous avons pu reformuler MP_1 comme un problème de partition d'un graphe d'intervalles par des stables de taille au plus k. MP_2 peut être vu, lui, comme un problème de Bin-Packing avec des contraintes de compatibilité entre les objets définies par le complément d'un graphe d'intervalles. Enfin, nous modéliserons MP_3 de la même manière que MP_2 en y ajoutant une contrainte : le nombre d'objets par boîte limité à k.

La première approche sera ainsi étudiée dans le Chapitre 1 et les deux suivantes dans le Chapitre 2. Les résultats que nous avons pu établir au travers de ces deux chapitres seront exprimés sous la forme de lemmes, propositions ou corollaires. Les théorèmes que nous énoncerons au premier chapitre sont des résultats déjà établis, ou en sont des conséquences directes.
Ainsi, le Chapitre 1, intitulé *"Sur le problème de la partition d'un graphe d'intervalles par des stables de taille bornée"*, traitera de la complexité du problème lui-même. Tout d'abord, nous rappelerons quelques propriétés ramarquables des graphes d'intervalles puis nous ferons l'état de l'art du problème. Ensuite, nous établirons des conditions suffisantes pour lesquelles il est polynômial, qui nous permettront ensuite de prouver la polynômialité de celui-ci pour les graphes d'intervalles propres. Nous poursuivrons en présentant un nouvel algorithme en $O(n \log n)$ pour le cas $k = 2$, soit le problème du couplage parfait dans le complément d'un graphe d'intervalles. Enfin, à partir de l'ensemble de ces résultats, nous élaborerons des algorithmes d'approximation polynômiaux performants pour le cas $\mathcal{NP}$-complet.

Le Chapitre 2 porte, lui, *"Sur une extension du problème de Bin-Packing"* où nous définirons le problème de Bin-Packing avec contraintes de compatibilité entre les objets. Nous représenterons celles-ci par un hypergraphe ou un graphe de compatibilité entre les objets et nous montrerons que le problème est approximable si et seulement si le problème de la coloration du complément de son graphe de compatibilité l'est aussi. Dans le même temps, nous donnerons une série de résultats concernant l'approximation

des problèmes MP_2 et MP_3.

Enfin, nous conclurons ce travail en évoquant certaines ouvertures possibles vers d'autres problématiques posées par la planification d'horaires. Mais avant toute chose, clarifions quelque peu les notations ainsi que le vocabulaire relatifs à la Théorie des Graphes, à la Combinatoire et à la Théorie de la Complexité que nous allons utiliser tout au long de ce mémoire.

Définitions et notations

Les symboles et notations ainsi que le vocabulaire que nous utiliseront sont ceux usuellement employés en Théorie des Graphes, en Combinatoire et en Théorie de la Complexité. Rappelons tout de même les définitions et notations nécessaires à une bonne compréhension du mémoire [7][2].
Tout d'abord, nous devons préciser que nous ne ferons référence ici qu'à des graphes non-orientés. Nous noterons donc $G = (V, E)$ comme étant le graphe non-orienté G dont l'ensemble des sommets est V et l'ensemble des arêtes est E. Nous noterons $xy \in E$ l'arête reliant x et y. Une orientation F des arêtes de E sera telle que : $(x, y) \in F$ implique que x et y sont reliés par une arête de E orientée de x vers y. Nous définirons $\overline{G}$ comme étant le complément de G, soit $\overline{G} = (V, \overline{E})$ où $\overline{E} = \{\{x, y\} \in V \times V \ / \ x \neq y$ et $xy \notin E\}$. Intuitivement, les arêtes de G deviennent les non-arêtes de $\overline{G}$ et vice versa. Un graphe G sera dit complet si toute paire de sommets distincts de G est reliée par une arête. Le graphe complet à n sommets sera noté K_n. Soit un sous-ensemble $A \subseteq V$ de sommets de G, nous définirons le sous-graphe induit par A comme étant $G_A = (A, E_A)$ où $E_A = \{xy \in E \ / \ x \in A$ et $y \in A\}$.

Donnons maintenant quelques définitions plus techniques relatives à $G = (V, E)$ un graphe non-orienté quelconque.
Une *clique* de G est un ensemble $C \subseteq V$ de sommets de G tel que $\forall x, y \in C$ avec $x \neq y$, $xy \in E$, c'est-à-dire tel que le sous-graphe induit par C soit complet. Une clique C est dite maximale si aucune autre clique de G ne contient proprement C comme sous-ensemble. Une clique C est donc dite maximum si aucune autre clique de G n'est de cardinalité strictement supérieure à C. Certains auteurs utilisent le terme d'ensemble complet pour désigner une clique. $\omega(G)$ dénote la cardinalité de la clique maximum de G.
Une *couverture par des cliques* $\mathcal{C}$ de G de taille q est une partition $C_1, \ldots, C_q$ de l'ensemble des sommets V tel que chaque $C_i \in \mathcal{C}$ soit une clique. $k(G)$ désigne la taille de la couverture de G par un minimum de cliques.
Un *stable* est un ensemble $S \subseteq V$ de sommets de G tel que $\forall x, y \in S$ avec $x \neq y$, $xy \notin E$, c'est-à-dire tel que le sous-graphe induit par S soit vide. Certains auteurs utilisent le terme d'ensemble indépendant pour désigner un stable. $\alpha(G)$ dénote la cardinalité du stable maximum de G.
Une *q-coloration* de G ou encore *une partition par des stables* $\mathcal{S}$ de G de taille q est une partition $S_1, \ldots, S_q$ de l'ensemble des sommets V tel que chaque $S_i \in \mathcal{S}$ soit un stable. $\chi(G)$, appelé nombre chromatique de G, désigne la taille de la partition de G par un minimum de stables.
De la même manière, nous définirons $\chi(G, k)$ comme étant la taille de la partition de G par un minimum de stables de taille au plus k, $k \in \mathbb{N}$.

A partir de ces quelques définitions, nous pouvons facilement voir que, pour un graphe $G = (V, E)$ quelconque où $|V| = n$, nous avons :

- $\omega(G) \leq \chi(G)$ et $\alpha(G) \leq k(G)$
- $\omega(G) = \alpha(\overline{G})$ et $\chi(G) = k(\overline{G})$
- $\chi(G, 1) = n$ et $\forall n' \in \mathbb{N}$, $n' \geq n$, $\chi(G, n') = \chi(G)$

Enfin, évoquons la notion de graphe biparti. Un graphe $G = (V, E)$ est biparti si ses sommets peuvent être partitionnés en deux stables disjoints X et Y. De manière équivalente, G est biparti si et seulement si G est 2-colorable. Il sera donc noté $G = (X, Y, E)$. Un graphe biparti complet sur $m + n$ sommets

partitionnés en deux stables de taille m et n sera noté $K_{m,n}$.

Concernant les hypergraphes, nous n'aurons réellement besoin que de leur définition précise.
Soit $X = \{x_1, \ldots, x_n\}$ un ensemble fini. Un hypergraphe sur X est une famille $H = (E_1, \ldots, E_m)$ de parties de X ($H \subseteq \mathcal{P}(X)$) avec :

- $E_i \neq \emptyset$
- $\bigcup_{i=1}^{m} E_i = X$

Un graphe est donc un hypergraphe dont toutes les arêtes sont de cardinalité 2. Un hypergraphe H sera souvent représenté en dessinant sur le plan des points correspondant aux sommets, une arête E_j étant représentée par une boucle sur son élément si $|E_j| = 1$, par un trait continu joignant ses deux éléments si $|E_j| = 2$ ou par un trait plein entourant ses éléments si $|E_j| \geq 3$.
On peut aussi définir un graphe G ou un hypergraphe H par sa matrice d'incidence $A = \left((a_j^i)\right)$, les colonnes représentant les arêtes $E_1, \ldots, E_m$, les lignes représentant les sommets $x_1, \ldots, x_n$, avec $a_j^i = 0$ si $x_i \notin E_j$ et $a_j^i = 1$ si $x_i \in E_j$.

Voici, pour terminer, quelques notations de la Théorie de la Complexité que nous utiliserons fréquemment dans ce travail.
Nous noterons $\mathcal{NP}$ comme étant la classe des problèmes polynomiaux non-déterministes et $\mathcal{P}$ la classe des problèmes polynomiaux déterministes. Un problème $P \in \mathcal{NP}$ sera dit $\mathcal{NP}$-complet si $\forall P' \in \mathcal{NP}$, P' se réduit polynômialement à P. Un problème de minimisation P sera dit α-approximable avec $\alpha \in \mathbb{R}$ s'il existe un algorithme A retournant une solution S_{ALG} tel que $S_{ALG} \leq \alpha S_{OPT}$ avec S_{OPT} la solution optimale pour P.
La Théorie de la Complexité a permis d'établir la $\mathcal{NP}$-complétude de nombreux problèmes, essentiellement des problèmes combinatoires. Nous allons ici en présenter un comme exemple : *le problème de la partition numérique* 3*-dimensionnelle*, problème peu connu, généralisant le problème de Tripartition (que nous énoncerons dans la suite du mémoire), et qui est à la "base" de la $\mathcal{NP}$-complétude de tous les problèmes que nous allons étudier.

Problème : *Partition numérique* 3*-dimensionnelle (Numerical 3-Dimensional Matching)*

Entrée : *W, X, Y trois ensembles disjoints contenant chacun m éléments, la taille $s(a) \in \mathbb{N}$ de chaque élément $a \in W \cup X \cup Y$ et une borne $Z \in \mathbb{N}$ tel que $\sum_{a \in W \cup X \cup Y} s(a) = mZ$.*

Question : *$W \cup X \cup Y$ peut-il être partitionné en m ensembles disjoints A_i tel que A_i contienne exactement un élément de W, de X, de Y et que pour $i \in \{1, \ldots, m\}$, $\sum_{a \in A_i} s(a) = Z$?*

Ce problème est donc $\mathcal{NP}$-complet si $\forall a \in W \cup X \cup Y$, $s(a) < \frac{Z}{2}$ [6]. Pour davantage de détails au sujet de la Théorie de la Complexité , nous renvoyons le lecteur à [4].

Nous pouvons à présent aborder le Chapitre 1.

1. SUR LE PROBLÈME DE LA PARTITION D'UN GRAPHE D'INTERVALLES PAR DES STABLES DE TAILLE BORNÉE

La formalisation mathématique du problème MP_1 nous a naturellement conduit, du point de vue de la Théorie des Graphes, au problème de la partition d'un graphe d'intervalles par des stables de taille bornée. En effet, soit $G = (V, E)$ le graphe défini de la manière suivante :

- $V = \{T_1, \ldots, T_n\}$ l'ensemble des tâches,

– $E = \{e = T_iT_j \ / \ \{i,j\} \in \{1,\ldots,n\}^2, i \neq j, T_i \text{ et } T_j \text{ se chevauchent }\}$.

La notion de vacation, comme étant un ensemble de tâches ne se chevauchant pas deux-à-deux, correspond exactement à la notion de stable dans G. De plus, les tâches pouvant être considérées comme des intervalles sur l'échelle du temps, le graphe G est donc le graphe des intersections d'un ensemble d'intervalles, appelé communément graphe d'intervalles. Ainsi, les vacations comprenant, dans le problème MP_1, un nombre de tâches limité à k, celui-ci se traduira-t-il par le problème de la partition du graphe d'intervalles G par des stables de taille au plus k.

Nous allons, pour débuter, discuter plus amplement de cette structure de graphe d'intervalles, prédominante dans l'ensemble de ce travail, et de ces riches propriétés, avant de nous lancer dans l'étude du problème tel que nous venons de le définir.

1.1 DÉFINITIONS ET PROPRIÉTÉS FONDAMENTALES DES GRAPHES D'INTERVALLES

Les graphes d'intervalles appartiennent à la grande classe des graphes des intersections d'une famille d'ensembles, que nous pouvons définir de la manière suivante.

Définition 1.1 *Soit $\mathcal{F}$ une famille d'ensembles non-vides. Le graphe des intersections de $\mathcal{F}$ est obtenu en représentant chaque ensemble de $\mathcal{F}$ par un sommet et en reliant deux sommets par une arête si et seulement si leurs ensembles correspondants s'intersectent.*

Ainsi, l'intersection d'une famille d'intervalles d'un ensemble linéairement ordonné (comme la droite réelle) est appelé graphe d'intervalles.

Définition 1.2 *Un graphe G non-orienté est appelé graphe d'intervalles si ses sommets peuvent être mis un-à-un en correspondance avec un ensemble d'intervalles $\mathcal{I}$ d'un ensemble linéairement ordonné de telle manière à ce que deux sommets soient reliés par une arête de G si et seulement si leurs intervalles corespondants s'intersectent. Nous appelerons $\mathcal{I}$ la représentation par intervalles de G.*

Par conséquent, nous assimilerons constamment dans cette étude les sommets d'un graphe d'intervalles à leurs intervalles correspondants (ces derniers étant toujours considérés comme fermés). Précisons donc quelques notations à ce sujet.

Notation. Soit $\mathcal{I}$ un ensemble d'intervalles.
Soit $I_i \in \mathcal{I}$, nous noterons d_{I_i} l'indice de début de I_i et f_{I_i} son indice de fin.
Soit $\{I_i, I_j\} \in \mathcal{I} \times \mathcal{I}$, nous noterons $I_i \cap I_j = \emptyset$ si I_i et I_j ne s'intersectent pas et $I_i \cap I_j \neq \emptyset$ s'ils s'intersectent.
Ainsi, nous noterons $I_i \prec I_j$ si $f_{I_i} < d_{I_j}$ (ce qui impliquera évidemment $I_i \cap I_j = \emptyset$).

Une autre définition importante est celle des graphes d'intervalles propres et des graphes d'intervalles unitaires que nous étudierons Section 4.

Définition 1.3 *Des intervalles sont dits unitaires si ces intervalles ont une taille unitaire. Un graphe d'intervalles unitaires est donc un graphe d'intervalles dont tous les intervalles sont unitaires.*
Des intervalles sont dits propres si aucun de ces intervalles n'en contient proprement (strictement) un autre. Un graphe d'intervalles propres est donc un graphe dont tous les intervalles sont propres.

Nous verrons d'ailleurs que la classe des graphes d'intervalles unitaires et la classe des graphes d'intervalles propres coïncident (Roberts [1969]). Définissons maintenant une autre classe de graphe en relation avec la structure d'intervalles ou plus exactement avec l'ordre défini par celle-ci.

Définition 1.4 *Un graphe $G = (V, E)$ non-orienté est dit de comparabilité s'il existe une orientation F des arêtes de E satisfaisant $F \cap F^{-1} = \emptyset$, $F \cup F^{-1} = E$ et $F^2 \subseteq F$ avec $F^{-1} = \{(y, x) \,/\, \{x, y\} \in V \times V, (x, y) \in F\}$ et $F^2 = \{(x, z) \,/\, \{x, y, z\} \in V \times V \times V, (x, y) \in F$ et $(y, z) \in F\}$.*
La relation F est un ordre partiel strict de V dont la relation de comparabilité est exactement E. F est appelé orientation transitive de G (ou de E) et $G = (V, F)$ graphe d'un ordre partiel.

L'ensemble des propriétés que nous allons énoncer maintenant sous forme de théorèmes, ne contiennent pas les démonstrations. Celles-ci figurent dans [7].

Théorème 1.1 (Propriété d'hérédité) *Un sous-graphe induit d'un graphe d'intervalles est un graphe d'intervalles.*

Théorème 1.2 (Hajös [1958]) *Un graphe d'intervalles est triangulé, c'est-à-dire que tout cycle de longueur 4 possède une corde.*

Théorème 1.3 (Ghouila-Houri [1962]) *Le complément d'un graphe d'intervalles est transitivement orientable.*

Preuve. En effet, un ensemble d'intervalles est un ensemble partiellement ordonné. $\diamondsuit$

Enfin, Gilmore et Hoffman ont donné une caractérisation totale des graphes d'intervalles.

Théorème 1.4 (Gilmore, Hoffman [1964]) *Un graphe G est un graphe d'intervalles si et seulement si G est triangulé et son complément $\overline{G}$ est un graphe de comparabilité. De plus, les cliques maximales de G peuvent être linéairement ordonnées tel que, pour chaque x, sommet de G, les cliques maximales contenant x apparaissent consécutivement.*

Nous pouvons en déduire les propriétés algorithmiques intéressantes suivantes.

Théorème 1.5 (Corollaire du Théorème 1.4) *Si G est un graphe d'intervalles alors :*
- *$\omega(G) = \chi(G)$ et $\alpha(G) = k(G)$ (G est dit parfait),*
- *$\omega(G)$, $\chi(G)$, $\alpha(G)$, $k(G)$ peuvent être déterminés en temps polynômial,*
- *en particulier, les problèmes de la partition de G par un minimum de stables et de la clique maximum de G peuvent être résolus en $O(n \log n)$.*

Etant donné son importance dans nos futures démonstations, nous allons tout de même décrire l'algorithme en $O(n \log n)$ de coloration d'un ensemble de n intervalles (partition d'un ensemble de n intervalles par un minimum de stables).

Algorithme de coloration d'un ensemble d'intervalles :

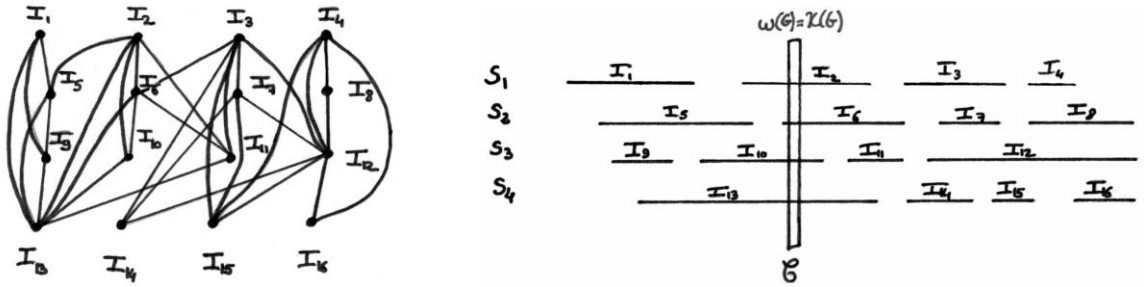
Entrée :
Un ensemble $\mathcal{I}$ de n intervalles $I_i = (d_{I_i}, f_{I_i})$, $i \in \{1, \ldots, n\}$;
Sortie :
Une partition $\mathcal{S}$ de $\mathcal{I}$ par un minimum de stables ;

Etape 1 :
Trier $\mathcal{I}$ selon les d_I croissants ;
Soit $\mathcal{S} = \emptyset$;
Pour chaque stable S_i faire
Définir un couple $(d_{S_i}, f_{S_i})_{S_i}$ avec d_{S_i} l'indice de début du premier intervalle de S_i et f_{S_i} l'indice de fin du dernier intervalle de S_i ;
Soit $S_1 = \{I_1\}$;
$\mathcal{S} = \mathcal{S} \cup S_1$;
Pour chaque I_i, $i \in \{2, \ldots, n\}$ faire
Rechercher par dichotomie selon f_{S_j}, $j \in \{1, \ldots, |\mathcal{S}|\}$ le plus petit indice f_{S_j} tel que $d_{I_i} > f_{S_j}$;

Soit f_{S_t} cet indice ;
Si $t \in \{1, \ldots, |\mathcal{S}|\}$ faire
$S_t = S_t \cup \{I_i\}$;
$f_{S_t} = f_{I_i}$;

Sinon faire
Soit $S_{|\mathcal{S}|+1} = \{I_i\}$;
$d_{S_{|\mathcal{S}|+1}} = d_{I_i}$ et $f_{S_{|\mathcal{S}|+1}} = f_{I_i}$;
$\mathcal{S} = \mathcal{S} \cup S_{|\mathcal{S}|+1}$;

Etape 2 :
Retourner $\mathcal{S}$;



Voici un exemple de graphe d'intervalles avec sa représentation par un ensemble d'intervalles partitionnés en un minimum de stables.

Nous pouvons à présent définir clairement notre problème.

1.2 DÉFINITION DU PROBLÈME ET ÉTAT DE L'ART

Donnons tout d'abord la définition du problème de la partition d'un graphe par des stables de taille bornée sous la forme d'un problème de décision.

Problème $\mathbf{P_1}$: *Partition d'un graphe par des stables de taille bornée*

Entrée : *$G = (V, E)$ un graphe, $k' \in \mathbb{N}$, $k \in \mathbb{N}$.*

Question : *Existe-t-il une partition de V en k' stables $S_1, \ldots, S_{k'}$ tel que $|S_i| \leq k$ pour $i \in \{1, \ldots, n\}$?*

Ce problème, défini ici pour un graphe quelconque, est $\mathcal{NP}$-complet. En effet, il contient le problème de la partition d'un graphe par des stables [9]. Il peut aussi être vu comme le problème de la coloration de G en k' couleurs tel que chaque couleur ne serve pas à colorer plus de k sommets.

Nous allons donc, dans ce premier chapitre, analyser la complexité du problème P_1 (version "minimisation") pour les graphes d'intervalles. Ce problème n'a été étudié que très récemment, en voici les deux principaux résultats.

Théorème 1.6 (Edmonds [1965] - Andrews, Atallah, Chen et Lee [2000]) *Pour $k = 2$, le problème P_1 peut être résolu en temps polynômial pour un graphe $G = (V, E)$ quelconque. En particulier, il peut être résolu en $O(n \log n)$ pour un graphe d'intervalles $G = (V, E)$ avec $|V| = n$.*

Preuve. Le problème pour $k = 2$ est équivalent au problème du couplage maximum dans $\overline{G}$. Celui-ci peut être résolu par l'Algorithme d'Edmonds en $O(n^3)$ [5]. Pour le problème du couplage maximum dans le complément d'un graphe d'intervalles, Andrews, Atallah, Chen et Lee ont mis au point un algo-

rithme s'exécutant en $O(n \log n)$ [1]. $\diamondsuit$

Théorème 1.7 (Boadlander, Jansen [1995]) *Le problème P_1 pour les graphes d'intervalles est $\mathcal{NP}$-complet pour $k \geq 4$. Il est en revanche polynômial quelque soit k, si k' est une constante.*

La preuve de ce théorème est complexe. Elle peut se faire par une réduction polynômiale à partir du *problème de la partition numérique* 3*-dimensionnelle* (que nous avons défini dans le Chapitre Définitions et Notations) [3]. La seconde affirmation du théorème peut être vérifiée simplement en montrant que l'énumération de toutes les solutions peut se faire en temps polynômial.

Problème ouvert : *La complexité du problème P_1 pour les graphes d'intervalles avec $k = 3$.*

L'ensemble des travaux effectués dans ce premier chapitre ont en fait été motivés par la recherche d'un algorithme polynômial pour le problème P_1 pour les graphes d'intervalles avec $k = 3$. Bien que cette recherche n'ait pas encore abouti, elle nous a permis de mettre en avant l'aspect polynômial de certains cas particuliers du problème et ainsi de le réduire de manière conséquente.

La détermination de $\chi(G, k)$ est donc un problème $\mathcal{NP}$-complet même pour les graphes d'intervalles. Nous allons voir qu'il est cependant possible de borner $\chi(G, k)$, et dans certains cas, d'en donner une valeur exacte. Mais énonçons tout d'abord la proposition suivante, triviale mais utile à la compréhension de nos futures analyses.

Proposition 1.1 *Soit $G = (V, E)$ un graphe quelconque avec $|V| = n$. Soit $k \in \mathbb{N}$.*

$$\chi(G, k) \geq \max(\chi(G), \lceil \frac{n}{k} \rceil)$$

Preuve. Clairement, nous avons les inégalités $\chi(G, k) \geq \chi(G)$ et $\chi(G, k) \geq \lceil \frac{n}{k} \rceil$, ce qui nous permet d'en déduire le résultat. $\diamondsuit$

Cette proposition donne une borne inférieure pour $\chi(G, k)$. Nous allons maintenant voir, dans le cas des graphes d'intervalles, pour quelles conditions suffisantes cette borne sera atteinte et l'impact que cela aura sur la complexité du problème.

1.3 DEUX CONDITIONS SUFFISANTES POUR LESQUELLES $\chi(G, K) = \chi(G)$ ET $\chi(G, K) = \lceil \frac{N}{K} \rceil$

Proposition 1.2 *Soit $G = (V, E)$ un graphe d'intervalles avec $|V| = n$. Soit $k \in \mathbb{N}$.*
Soit $S_1, \ldots, S_{\chi(G)}$ une partition de G par un minimum de stables.
Si, $\forall i$, $i \in \{1, \ldots, \chi(G)\}$, $|S_i| \leq k$ alors $\chi(G, k) = \chi(G)$.
Si, $\forall i$, $i \in \{1, \ldots, \chi(G)\}$, $|S_i| \geq k$ alors $\chi(G, k) = \lceil \frac{n}{k} \rceil$.
De plus, dans les deux cas, le problème P_1 peut être résolu en $O(n \log n)$.

Preuve. Le premier point est immédiat par la Proposition 1.1 et le Théorème 1.5. La démonstration du second point est elle aussi algorithmique. Elle est notamment basée sur le lemme suivant.

Lemme 1.1 *Soit G un graphe d'intervalles. Soit $k \in \mathbb{N}$. Soit k' stables $S_1, \ldots, S_{k'}$ de G tel que :*
- *$k' \in \{1, \ldots, k\}$*
- *$\forall i$, $i \in \{1, \ldots, k'\}$, $|S_i| = k + r_i$, $r_i \in \{1, \ldots, k-1\}$*
- *$r = \sum_{i=1}^{k'} r_i$, $1 \leq r \leq k$*

Il existe un stable $S' = \{I'_1, \ldots, I'_r\}$ de G de taille r tel que $\forall i$, $i \in \{1, \ldots, k'\}$, r_i intervalles de S' appartiennent à S_i. Autrement dit, il existe une partition de l'ensemble des intervalles des k' stables $S_1, \ldots, S_{k'}$ en k' stables de taille k et un stable de taille r.

De plus, la construction de S' peut se faire en temps linéaire.

Preuve du Lemme 1.1. Nous allons présenter un algorithme polynômial pour la construction de S'. Le principe de cet algorithme est de sélectionner successivement, parmi les stables de taille au moins k, les intervalles ayant leurs indices de début les plus grands, de les extraire du stable auquels ils appartiennent et de les inclure dans le stable S'.

Algorithme 1.1 :

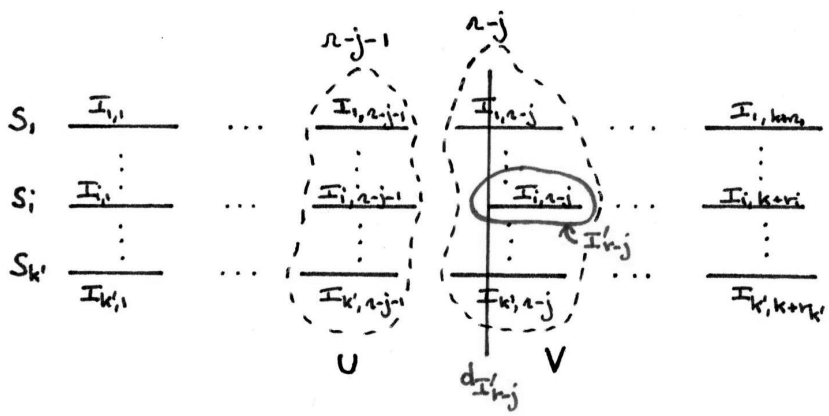
Entrée :
Un ensemble de stables $S_1, \ldots, S_{k'}$ tel que
$S_1 = \{I_{1,1}, \ldots, I_{1,k+r_1}\}$ avec $I_{1,1} \prec \ldots \prec I_{1,k+r_1}$
$\vdots$
$S_{k'} = \{I_{k',1}, \ldots, I_{k',k+r_{k'}}\}$ avec $I_{k',1} \prec \ldots \prec I_{k',k+r_{k'}}$;

Sortie :
Un stable S' ;

Etape 1 :
$S' = \emptyset$, $j = 0$;
Tant que $|S'| < r$ faire
$F = \emptyset$;
Pour chaque $i \in \{1, \ldots, k'\}$ faire
Si $r_i > j$ faire
$F = F \cup \{I_{i,k+r_i-j}\}$;
Sélectionner dans F l'intervalle I'_{r-j} tel que $\forall I \in F, d_I \leq d_{I'_{r-j}}$;
$S' = S' \cup \{I'_{r-j}\}$ et $j = j + 1$;

Etape 2 :
Retourner S' ;

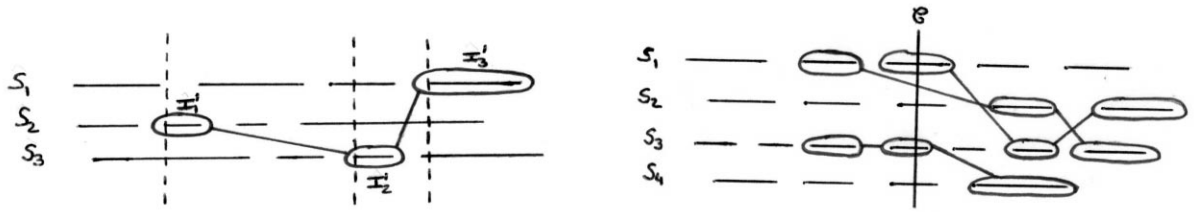
L'algorithme s'exécute en $O(kk')$, il est donc linéaire en le nombre d'intervalles considérés. Montrons à présent que S' est bien un stable tel qu'il est décrit dans le lemme.
Par construction, il est clair que $|S'| = r$ et que $\forall i$, $i \in \{1, \ldots, k'\}$, r_i intervalles de S' appartiennent à S_i.
De plus, $\forall j$, $j \in \{0, \ldots, r-1\}$, $I'_{r-j-1} \prec I'_{r-j}$. En effet, soit $j \in \{0, \ldots, r-1\}$. Soit $U = \{I_{1,k+r_1-j-1}, \ldots, I_{k',k+r_{k'}-j-1}\}$ et $V = \{I_{1,k+r_1-j}, \ldots, I_{k',k+r_{k'}-j}\}$. Si $I'_{r-j} \in V$ est tel que $\forall I_v \in V$, $d_{I_v} \leq d_{I'_{r-j}}$ alors $\forall I_u \in U$, $I_u \prec I'_{r-j}$ (voir la figure ci-dessous). En effet, s'il existait $I_{u,k+r_u-j-1} \in U$ tel que $I_{u,k+r_u-j-1} \cap I'_{r-j} \neq \emptyset$, alors $f_{I_{u,k+r_u-j-1}} \geq d_{I'_{r-j}}$ et donc $d_{I_{u,k+r_u-j}} > d_{I'_{r-j}}$ avec $I_{u,k+r_u-j} \in V$, ce qui est une contradiction.
Or $I'_{r-j-1} \in U$. Par conséquent, $I'_{r-j-1} \prec I'_{r-j}$, $\forall j$, $j \in \{0, \ldots, r-1\}$ et le lemme est vérifié. $\diamondsuit$



Corollaire 1.1 *Soit G un graphe d'intervalles. Soit $k \in \mathbb{N}$.*
Soit k stables $S_1, \ldots, S_k$ de G tel que $\forall i$, $i \in \{1, \ldots, k\}$, $|S_i| = k + 1$.
Il existe un stable $S' = \{I'_1, \ldots, I'_k\}$ de G de taille k tel que $I'_1 \in S_1, \ldots, I'_k \in S_k$. De plus, la construction de S' peut se faire en temps linéaire par l'Algorithme 1.1.

Preuve du Corollaire 1.1. Immédiate par le Lemme 1.1, en posant $k' = k$ et $r_i = 1$, $\forall i$, $i \in \{1, \ldots, k\}$. ♢

Clairement, le Lemme 1.1 peut s'appliquer récursivement sur un ensemble de stables $S_1, \ldots, S_{k'}$ avec $k' \geq k$ et $r \geq k$ pour créer des stables de taille k, tant que $r \geq k$ ou $k' \geq k$, puis créer un stable de taille au plus k, lorsque $1 \leq r \leq k$ et $1 \leq k' \leq k$. Par conséquent, en posant $k' = \chi(G)$ et $r = n - k\chi(G)$, il sera possible d'obtenir en temps linéaire une partition de G par un ensemble de stables de taille k et un stable de taille $r \in \{1, \ldots, k\}$. Ainsi, la taille de cette partition sera égale à $\lceil \frac{n}{k} \rceil$ qui est la borne inférieure pour $\chi(G, k)$ dans le cas présent, elle sera donc optimale.
La partition de G par un minimum de stables $S_1, \ldots, S_{\chi(G)}$ pouvant être calculée en $O(n \log n)$ d'après le Théorème 1.5, la Proposition 1.2 est donc démontrée dans sa totalité.
♢



Voici deux exemples d'applications du Lemme 1.1 et de la Proposition 1.2.

Nous allons voir que la Proposition 1.2 est fondamentale dans la démonstration de la polynômialité du problème P_1 pour les graphes d'intervalles propres.

1.4 $P_1 \in P$ POUR LES GRAPHES D'INTERVALLES PROPRES

Les graphes d'intervalles propres (intervalles n'incluant pas proprement d'autres intervalles) forme une sous-classe importante des graphes d'intervalles. Cette classe a été en effet très étudiée, elle possède de nombreuses propriétés intéressantes. Nous pouvons notamment énoncer le théorème suivant les caractérisant.

Théorème 1.8 (Roberts [1969]) *Soit G un graphe. Les conditions suivantes sont équivalentes :*
(i) G est un graphe d'intervalles ne contenant pas $K_{1,3}$ comme sous-graphe induit,
(ii) G est un graphe d'intervalles propres,
(iii) G est un graphe d'intervalle unitaires.

La preuve de ce théorème figure dans [7].

Nous allons montrer que pour cette classe de graphe, le problème P_1 est polynômial, et ceci grâce à la proposition suivante.

Proposition 1.3 *Soit $G = (V, E)$ un graphe d'intervalles propres avec $|V| = n$. Soit $k \in \mathbb{N}$.*
Il existe une partition de G par un minimum de stables $S_1, \ldots, S_{\chi(G)}$ tel que :
- *soit $\forall i$, $i \in \{1, \ldots, \chi(G)\}$, $|S_i| \leq k$*
- *soit $\forall i$, $i \in \{1, \ldots, \chi(G)\}$, $|S_i| \geq k$*

De plus, cette partition peut être calculée en temps polynômial.

Preuve. Soit $G = (V, E)$ un graphe d'intervalles propres. Soit $S_1, \ldots, S_{\chi(G)}$ une partition de G par un minimum de stables obtenue par un algorithme polynômial de coloration de G (Théorème 1.5).
Si $\forall i$, $i \in \{1, \ldots, \chi(G)\}$, $|S_i| \leq k$ ou si $\forall i$, $i \in \{1, \ldots, \chi(G)\}$, $|S_i| \geq k$, la Proposition 1.3 est vérifiée. Nous allons donc montrer que si $\exists S_i$ tel que $|S_i| < k$ et $\exists S_j$ tel que $|S_j| > k$, l'algorithme suivant nous ramène en temps polynômial à un des deux cas précédemment cités.
Le principe de cet algorithme est d'extraire des sommets d'un stable S_j tel que $|S_j| > k$ et de les inclure dans un stable S_i tel que $|S_i| < k$ de telle manière à ce que, après cette opération que nous appelerons une "redéfinition" de S_i et S_j, soit $|S_i| = k$, soit $|S_j| = k$. Une telle permutation des sommets de S_i et S_j sera recherchée par énumération de toutes les combinaisons des couples de stables constructibles à partir des sommets de S_i et S_j, soit $C^2_{O((|S_i|+|S_j|)^k)}$ couples. La garantie de l'existence de cette permutation nous sera fournie par le Lemme 1.2 que nous exposerons après la description de l'algorithme en question.

Algorithme 1.2 :

Entrée :
$S_1, \ldots, S_{\chi(G)}$: une partition de G graphe d'intervalles propres par un minimum de stables ;

Sortie :
$S_1, \ldots, S_{\chi(G)}$: une partition de G graphe d'intervalles propres par un minimum de stables
tel que soit $\forall S_i$, $|S_i| \leq k$, soit $\forall S_i$, $|S_i| \geq k$;

Etape 1 :
Si $\forall S_i$, $|S_i| \leq k$ faire
retourner $S_1, \ldots, S_{\chi(G)}$;
Si $\forall S_i$, $|S_i| \geq k$ faire
retourner $S_1, \ldots, S_{\chi(G)}$;

Etape 2 :
Choisir en temps polynômial deux stables S_i et S_j tel que $|S_i| < k$ et $|S_j| > k$;
Enumérer les $C^2_{O((|S_i|+|S_j|)^k)}$ manières de construire deux nouveaux stables S_i et S_j
à partir de l'ensemble des $|S_i| + |S_j|$ sommets de S_i et S_j ;
Parmi l'ensemble des couples de stables solutions de l'énumération,
en choisir un redéfinissant S_i et S_j tel que soit $|S_i| = k$, soit $|S_j| = k$;

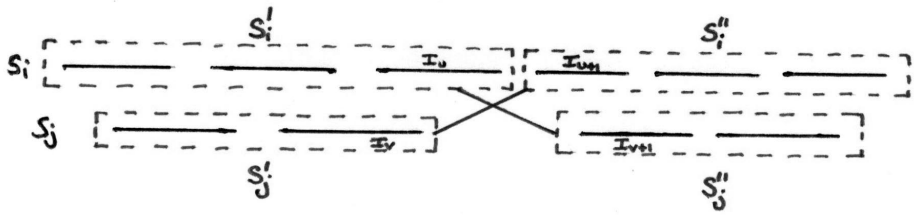
Etape 3 :
Aller à l'Etape 1 ;

Montrons maintenant la validité de cet algorithme. Pour cela, nous allons montrer qu'il existe toujours un couple solution de l'énumération tel que soit $|S_i| = k$, soit $|S_j| = k$.

Lemme 1.2 *Soit G un graphe d'intervalles propres. Soit $k \in \mathbb{N}$.*
Soit deux stables S_i et S_j de G tel que $|S_i| = k + r_i$ et $|S_j| = k - r_j$ avec $r_i \geq 1$ et $r_j \geq 1$.
Il existe une permutation des sommets de S_i et S_j redéfinissant ceux-ci tel que $|S_i| = k + r_i - \alpha$ et $|S_j| = k - r_j + \alpha$ avec $\alpha \geq 1$, $r_i - \alpha \geq 0$ et $\alpha - r_j \leq 0$.

Preuve du Lemme 1.2 Soit $S_i = \{I_{i_1}, \ldots, I_{i_{k+r_i}}\}$ et $S_j = \{I_{j_1}, \ldots, I_{j_{k+r_j}}\}$ avec $I_{i_1} \prec \ldots \prec I_{i_{k+r_i}}$ et $I_{j_1} \prec \ldots \prec I_{j_{k+r_j}}$.
Tout d'abord s'il existe un intervalle $I_i \in S_i$ tel que $\forall I_j \in S_j$, $I_i \cap I_j = \emptyset$ alors il existe une permutation des sommets de S_i et S_j redéfinissant ceux-ci tel que $S_i = S_i \setminus \{I_i\}$ et $S_j = S_j \cup \{I_i\}$, le lemme est alors prouvé avec $\alpha = 1$.
Si ce n'est pas le cas, alors $\forall I_i \in S_i$, $\exists I_j \in S_j$ / $I_i \cap I_j \neq \emptyset$. Nous pouvons traduire cela plus simplement par la condition "tout intervalle de S_i est intersecté par au moins un intervalle de S_j".
Or le graphe G étant sans $K_{1,3}$, tout intervalle de S_j ne peut intersecter au plus que deux intervalles de S_i.
Supposons donc que $\forall I_j \in S_j$, I_j intersecte $I_u \in S_i$ et $I_v \in S_i$, $u \neq v$. Les intervalles du graphe étant propres, nous pouvons affirmer que $\forall I_j \in S_j$, si $I_j \cap I_{i_u} \neq \emptyset$ et $I_j \cap I_{i_v} \neq \emptyset$ avec $\{I_{i_u}, I_{i_v}\} \in S_i \times S_i$, $u \neq v$, $u \neq k + r_i$ et $v \neq 1$ alors $v = u + 1$ selon l'ordre défini sur les S_i. C'est-à-dire, si un intervalle

de S_j intersectent deux intervalles de S_i alors nécessairement ces deux intervalles forment une paire d'intervalles consécutifs dans S_i.
Il existe dans S_i, $k + r_i - 1$ paires d'intervalles consécutifs $\{I_{i_1}, I_{i_2}\}, \ldots, \{I_{i_{k+r_i-1}}, I_{i_{k+r_i}}\}$ et r_i étant au moins égal à 1, nous pouvons affirmer que dans le pire des cas, il en existe au moins k. Or, il existe au plus $k - 1$ intervalles dans S_j ($r_j = 1$), chacun pouvant intersecter au plus une paire d'intervalles consécutifs de S_i. Par conséquent, dans le pire des cas, il existera au moins une paire d'intervalles consécutifs $\{I_{i_u}, I_{i_{u+1}}\} \in S_i$ avec $u \neq k + r_i$ et $u + 1 \neq 0$ tel que $\nexists I_j \in S_j \ / \ I_j \cap I_{i_u} \neq \emptyset$ et $I_j \cap I_{i_{u+1}} \neq \emptyset$ (voir la figure ci-dessous).



Ainsi, nous pouvons écrire $S_i = S'_i \cup S''_i$ avec $S'_i = \{I_{i_1}, \ldots, I_{i_u}\}$ et $S''_i = \{I_{i_{u+1}}, \ldots, I_{i_{k+r_i}}\}$, $S_j = S'_j \cup S''_j$ avec $S'_j = \{I_{j_1}, \ldots, I_{j_v}\}$ et $S''_j = \{I_{j_{v+1}}, \ldots, I_{i_{k+r_j}}\}$, tel que $I_{j_v} \cap I_{i_{u+1}} = \emptyset$ et $I_{j_u} \cap I_{i_{v+1}} = \emptyset$. Il existe donc une permutation des sommets de S_i et S_j redéfinissant ceux-ci tel que $S_i = S'_i \cup S''_j$ et $S_j = S'_j \cup S''_i$. Or d'après la discussion précédente, $|S'_j| < |S'_i|$ et $|S''_j| < |S''_i|$. Par conséquent nous pouvons poser $|S''_j| = |S''_i| - \alpha$ avec $1 \leq \alpha \leq r_i$ et en conclure que $|S_i| = k + r_i - \alpha$ et $|S_j| = k - r_j + \alpha$ avec $\alpha \geq 1$, $r_i - \alpha \geq 0$ et $\alpha - r_j \leq 0$. $\diamondsuit$

A partir du Lemme 1.2, nous pouvons établir le Corollaire 1.2 qui nous permet donc de conclure quant à la validité de l'Algorithme 1.2.

Corollaire 1.2 *Soit G un graphe d'intervalles propre. Soit $k \in \mathbb{N}$.*
Soit deux stables S_i et S_j de G tel que $|S_i| = k + r_i$ et $|S_j| = k - r_j$ avec $r_i \geq 1$ et $r_j \geq 1$.
Il existe une permutation des sommets de S_i et S_j redéfinissant ceux-ci tel que :
- *si $r_i = r_j$ alors $|S_i| = k$ et $|S_j| = k$*
- *si $r_i > r_j$ alors $|S_i| = k + r_i - r_j$ et $|S_j| = k$*
- *si $r_i < r_j$ alors $|S_i| = k$ et $|S_j| = k + r_j - r_i$*

Preuve du Corollaire 1.2. Par application successive du Lemme 1.2 tant que $|S_i| > k$ et $|S_j| < k$, nous obtenons le résultat souhaité. $\diamondsuit$

Analysons à présent la complexité de l'Algorithme 1.2. A chaque passage par l'Etape 2, nous fixons soit $|S_i|$ à k, soit $|S_j|$ à k. Or il y a $\chi(G)$ stables. Par conséquent, nous n'effecturons qu'au plus $\chi(G)$ fois les Etapes 1, 2 et 3. Les Etapes 1 et 3 se font en temps linéaire. L'Etape 2, elle, se fait en $O(C^2_{O((|S_i|+|S_j|)^k)})$. Ainsi la complexité de l'Algorithme 1.3, noté $C(n)$, sera :

$$C(n) = \chi(G)(O(C^2_{O((|S_i|+|S_j|)^k)}) + O(n))$$

Or $\chi(G) \leq n$, $k \geq 1$ et $\forall \{i, j\} \in \{1, \ldots, \chi(G)\}^2$, $i \neq j$, $|S_i| + |S_j| \leq n$. Par conséquent,

$$C(n) = O(n^{(2k+1)})$$

ce qui est bien polynômial en n, quelque soit k. La Proposition 1.3 est donc entièrement démontrée. $\diamondsuit$

Ainsi, nous pouvons établir le résultat suivant.

Proposition 1.4 *Soit $G = (V, E)$ un graphe d'intervalles propres avec $|V| = n$. Soit $k \in \mathbb{N}$.*

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

De plus, le problème P_1 peut être résolu en temps polynômial pour G.

Preuve. La preuve est immédiate par le Théorème 1.5, les Propositions 1.2 et 1.4. $\diamondsuit$

1.5 UN ALGORITHME EN $O(N\ LOG\ N)$ POUR LE PROBLÈME DU COUPLAGE MAXIMUM DANS LE COMPLÉMENT D'UN GRAPHE D'INTERVALLES

Dans notre recherche d'un algorithme polynômial résolvant le problème P_1 pour $k = 3$, nous nous sommes intéressés au problème P_1 pour $k = 2$ (le problème de la détermination de $\chi(G, 2)$). Le problème est alors équivalent au problème du couplage maximum dans le complément d'un graphe d'intervalles. Nous avons trouvé à ce sujet un article récent [1], dans lequel les auteurs proposent un algorithme récursif complexe pour résoudre ce problème. Nous allons ici présenter un algorithme itératif original basé sur les propriétés des intervalles que nous avons mises en avant jusqu'ici.

Algorithme 1.3 :

Entrée :
Un ensemble $\mathcal{I}$ de n intervalles $I_i = (d_{I_i}, f_{I_i})$, $i \in \{1, \ldots, n\}$;

Sortie :
Une partition $\mathcal{S}_2$ de $\mathcal{I}$ par un minimum de stables de taille au plus 2 ;

Etape 1 :
$\mathcal{S}_2 = \emptyset$; Colorier $\mathcal{I}$ (Algorithme de coloration d'un ensemble d'intervalles) ;
Soit $\mathcal{S} = \{S_1, \ldots, S_{\chi}(G)\}$ la partition de $\mathcal{I}$ ainsi obtenue ;

Etape 2 :
Si $\forall i$, $|S_i| \leq 2$ faire
Retourner $\mathcal{S}_2 = \mathcal{S}$;
Si $\forall i$, $|S_i| \geq 2$ faire
Aller à l'Etape 5 ;

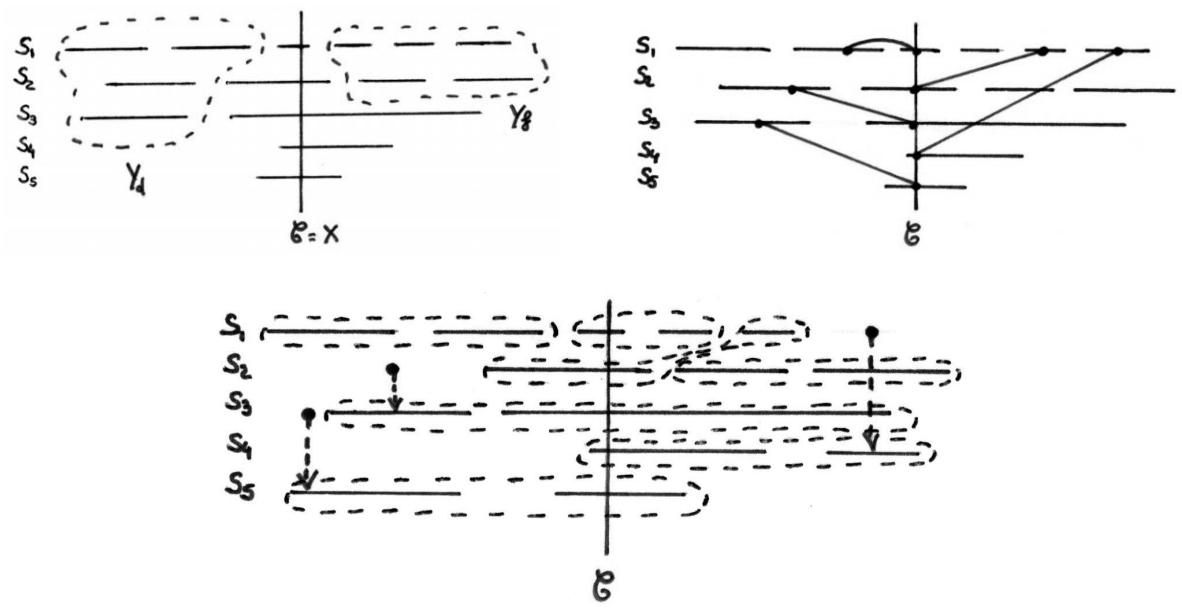
Etape 3 :
Calculer une clique maximum $\mathcal{C} = \{c_1, \ldots, c_{\chi(G)}\}$ de G avec $c_1 \in S_1, \ldots, c_{\chi(G)} \in S_{\chi(G)}$;
Calculer un couplage maximum $\mathcal{M}$ dans $G_b = (X, Y, A)$, le graphe biparti convexe tel que $X = \mathcal{C}$, $Y = \mathcal{I} \setminus \mathcal{C}$, $A = \{xy \ / \ x \in X, y \in Y \text{ et } xy \in \overline{E}\}$;

Etape 4 :
Compléter les stables $S_i = \{I_i\}$ de taille 1 par les intervalles I_j
avec lesquels ils sont couplés dans $\mathcal{M}$ (en extrayant I_j du stable auquel il appartient) ;
Inclure dans $\mathcal{S}_2$ les stables $S_i = \{I_i\}$ de taille 1 n'ayant pu être complétés ;

Etape 5 :
Découper les stables restants (de taille ≥ 2) en un ensemble de stables de taille 2
et un stable de taille 1 si le nombre d'intervalles de l'ensemble de ces stables est impair (Proposition 1.2) ;

Etape 6 :
Retourner $\mathcal{S}_2$;

Proposition 1.5 *Soit un ensemble $\mathcal{I} = \{I_1, \ldots, I_n\}$ de n intervalles. L'Algorithme 1.3 retourne une partition de $\mathcal{I}$ par un minimum de stables de taille au plus 2 en $O(n \log n)$.*



La figure ci-dessus décrit les Etapes 3, 4 et 5.

Preuve. Une preuve formelle de la validité ainsi que de la complexité de cet algorithmme serait longue et complexe, nous préférons donc en expliquer ici le principe et décrire les propriétés qui nous permettent d'affirmer qu'il peut s'exécuter en $O(n \log n)$.
Par le Théorème 1.5, l'Etape 1 peut s'exécuter en $O(n \log n)$. Après l'Etape 1, nous avons une partition de $\mathcal{I}$ par un minimum de stables, soit $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$. Par la Proposition 1.2, nous savons que si, $\forall i$, $|S_i| \leq 2$, alors $\chi(G, 2) = \chi(G)$ et $\mathcal{S}_2 = \mathcal{S}$, et que si, $\forall i$, $|S_i| \geq 2$, alors $\chi(G, 2) = \lceil \frac{n}{2} \rceil$ et $\mathcal{S}_2$ peut être obtenu en temps linéaire par l'Etapes 5.
Le problème est donc d'obtenir une partition de $\mathcal{I}$ par $\chi(G)$ stables où l'on est un minimum de stables de taille 1. Ceci est fait à l'Etape 3 par un couplage maximum entre les intervalles appartenant à une clique maximum $\mathcal{C}$ (dont font nécessairement partie les intervalles des stables de taille 1) et le reste des intervalles de $\mathcal{I}$, soit $\mathcal{I} \setminus \mathcal{C}$. La recherche d'une clique maximum $\mathcal{C}$ peut se faire en $O(n \log n)$, en énumérant les cliques maximales de $\mathcal{I}$ à partir de $\mathcal{S}$ (par le Théorème 1.4, nous pouvons affirmer qu'il y en a n au plus). Ensuite, le graphe biparti G_b tel qu'il est défini dans l'algorithme étant convexe, le problème du couplage maximum dans celui-ci peut se résoudre en $O(n \log n)$ [1]. Définissons donc ce qu'est la convexité pour un graphe biparti.

Définition 1.4 *Un graphe biparti convexe $G = (X, Y, E)$ est un graphe biparti avec $X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, \ldots, y_n\}$ et E l'ensemble des arêtes. Une arête $x_i y_j \in E$ si et seulement si $x_i \in X$, $y_j \in Y$ et $D_i \leq j \leq F_i$, où D_i (respectivement l_i) est l'indice du premier (respectivement du dernier) sommet dans Y auquel x_i est relié.*

La convexité de G_b peut se montrer par construction. En effet, soit Y_d l'ensemble des intervalles I_j tel que $I_j \in \mathcal{I}\setminus\mathcal{C}$, $\exists I_i \in \mathcal{C}$ / $I_j \prec I_i$ (c'est-à-dire les intervalles à "gauche de la clique $\mathcal{C}$") trié par ordre croissant des f_{I_j} et Y_f l'ensemble des intervalles I_j tel que $I_j \in \mathcal{I}\setminus\mathcal{C}$, $\exists I_i \in \mathcal{C}$ / $I_j \succ I_i$ (c'est-à-dire les intervalles à "droite de la clique $\mathcal{C}$") trié par ordre croissant des d_{I_j}. Nous pouvons construire Y de la manière suivante : $Y = Y_d \cup Y_f = \{y_1, \ldots, y_t, y_{t+1}, \ldots, y_{|Y_d \cup Y_f|}\}$, $Y_d = \{y_1, \ldots, y_t\}$ et $Y_f = \{y_{t+1}, \ldots, y_{|Y_d \cup Y_f|}\}$. Soit $x_i \in X$. Si D_i est l'indice du premier sommet (intervalle) auquel x_i est relié (couplé) dans Y et F_i l'indice du dernier alors l'ordre défini sur Y est tel que les sommets (intervalles) situés entre F_i et L_i seront nécessairement reliés eux aussi à x_i (pourront être couplé avec x_i). D'après la Définition 1.4, G_b est donc convexe.
De plus, il n'est pas nécessaire de construire explicitement G_b pour y calculer un couplage maximum. En effet, en définissant pour chaque $x \in X$ le couple (D_i, F_i) et en triant l'ensemble X selon les D_i croissants (et selon les F_i croissants en cas d'égalité des D_i), il sera possible de trouver un couplage

maximum de G_b par un simple parcours de l'ensemble X, en le maintenant trié par insertion dichotomique ($O(\log n)$) si nécessaire. Par conséquent, le calcul du couplage maximum pourra s'effectuer en $O(n \log n)$.
L'Etape 4 utilise ensuite le résultat de ce couplage pour compléter les stables de taille 1 par un intervalle de $\mathcal{I} \setminus \mathcal{C}$ si cela est possible : ce processus doit se poursuivre tant qu'il se crée des stables $S_i = \{I_i\}$ de taille 1 tel que $\{I_i, I_j\} \in \mathcal{M}$. Cette étape peut s'effectuer en $O(\chi(G))$, c'est-à-dire en $O(n)$ ($\chi(G) \leq n$). Ayant ainsi minimiser le nombre de stables de taille 1, l'Etapes 5 nous permet de coupler de manière optimale les intervalles des stables restants (de taille supérieure ou égale à 2) en $O(n)$ (Proposition 1.2). $\diamondsuit$

1.6 ALGORITHMES D'APPROXIMATION POLYNÔMIAUX POUR LE CAS NP-COMPLET

Après avoir détaillé quelques cas où P_1 est polynômial pour les graphes d'intervalles, nous allons maintenant proposer un algorithme d'approximation polynômial pour traiter le cas difficile, soit G un graphe d'intervalles avec $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$ une partition minimum de G par des stables tel que $\exists S_i$ tel que $|S_i| < k$ et $\exists S_j$ tel que $|S_j| > k$.

Algorithme 1.4 :

Entrée :
G un graphe d'intervalles et $k \in \mathbb{N}$;

Sortie :
Une partition $\mathcal{S}_k$ de G par des stables de taille au plus k ;
Etape 1 :
$\mathcal{S}_k = \emptyset$; Colorier G ;
Soit $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$ une partition minimum de G par des stables ;

Etape 2 :
Soit $\mathcal{S}'$ l'ensemble des stables $S_i \in \mathcal{S}$ tel que $|S_i| \geq k$;
Partitionner $\mathcal{S}'$ en le minimum de stables de taille au plus k ;
Inclure l'ensemble des stables ainsi obtenus dans $\mathcal{S}_k$;

Etape 3 :
Pour chaque $S \in \mathcal{S} \setminus \mathcal{S}'$ ($|S| < k$) faire
Inclure S dans $\mathcal{S}_k$;

Etape 4 :
Retourner $\mathcal{S}_k$;

Proposition 1.6 *Soit $\chi(G,k)_{ALG} = |\mathcal{S}_k|$. L'Algorithme 1.4 est un algorithme d'approximation en $O(n \log n)$ pour le problème P_1 tel que*

$$\chi(G,k)_{ALG} \leq \chi(G,k) + \frac{k-1}{k}\chi(G) + \frac{1}{k}$$

Soit,

$$\chi(G,k)_{ALG} \leq \frac{2k-1}{k}\chi(G,k) + \frac{1}{k}$$

Preuve. Cet algorithme peut s'exécuter en $O(n \log n)$ par le Théorème 1.5 et la Proposition 1.2. Analysons donc $|\mathcal{S}_k|$. Notons $|\mathcal{S}'|$ le nombre d'intervalles de l'ensemble des stables de $\mathcal{S}'$ et $\chi'(G)$ le nombre de stables $S \in \mathcal{S} \setminus \mathcal{S}'$. Nous avons donc

$$\chi(G,k)_{ALG} = \lceil \frac{|\mathcal{S}'|}{k} \rceil + \chi'(G)$$

Or $|\mathcal{S}'| + \sum_{\forall S \in \mathcal{S} \setminus \mathcal{S}'} |S| \leq k\chi(G,k)$, donc

$$\chi(G,k)_{ALG} \leq \chi(G,k) + \chi'(G) - \frac{\sum_{\forall S \in \mathcal{S} \setminus \mathcal{S}'} |S|}{k} + 1$$

Or $\forall S \in \mathcal{S} \setminus \mathcal{S}'$, $|S| \geq 1$ et $\chi'(G) \leq \chi(G) - 1$. Par conséquent,

$$\chi(G,k)_{ALG} \leq \chi(G,k) + \frac{k-1}{k}\chi(G) + \frac{1}{k}$$

$\diamondsuit$

Corollaire 1.3 *Soit $G = (V, E)$ un graphe d'intervalles avec $|V| = n$. Soit $k \in \mathbb{N}$.*

$$\max(\chi(G), \lceil \frac{n}{k} \rceil) \leq \chi(G,k) \leq \frac{n}{k} + \chi(G)$$

Preuve. Par la Propositions 1.1, nous avons la première inégalité. Ensuite,

$$\chi(G,k)_{ALG} = \lceil \frac{|\mathcal{S}'|}{k} \rceil + \chi'(G)$$

Or $\chi(G,k) \leq \chi(G,k)_{ALG}$, $\lceil \frac{|\mathcal{S}'|}{k} \rceil \leq \lceil \frac{n}{k} \rceil$ et $\chi'(G) \leq \chi(G) - 1$. Par conséquent,

$$\chi(G,k) \leq \frac{n}{k} + \chi(G)$$

$\diamondsuit$

Nous pouvons améliorer les performances de cet algorithme en utilisant certaines propriétés de l'Algorithme 1.3. En effet, il nous suffit de remplacer l'Etape 1 de l'Algorithme 1.4 par les Etapes 1, 2, 3 et 4 pour obtenir la borne suivante.

Proposition 1.7 *L'algorithme obtenu en remplaçant l'Etape 1 de l'Algorithme 1.4 par les Etapes 1, 2, 3 et 4 de l'Algorithme 1.3 donne une approximation en $O(n \log n)$ pour le problème P_1 tel que*

$$\chi(G,k)_{ALG} \leq \chi(G,k) + \frac{k-2}{k}\chi(G) + \frac{2}{k}$$

Soit,

$$\chi(G,k)_{ALG} \leq \frac{2k-2}{k}\chi(G,k) + \frac{2}{k}$$

Preuve. Par le Théorème 1.5, les Propositions 1.2 et 1.5, le nouvel algorithme peut s'exécuter en $O(n \log n)$. La preuve de la borne est identique à celle de la Proposition 1.6. L'amélioration réside dans le fait que tous les stables S_i de taille 1 appartiennent nécessairement à une solution optimale. Ainsi, nous obtenons ici l'inégalité,

$$\chi(G,k)_{ALG} = \chi(G,k) + \chi'(G) - \frac{\sum_{\forall S \in \mathcal{S} \setminus \mathcal{S}'} |S|}{k} + 1$$

Or $\forall S \in \mathcal{S} \setminus \mathcal{S}'$, $|S| \geq 2$ et $\chi'(G) \leq \chi(G) - 1$. Par conséquent,

$$\chi(G,k)_{ALG} = \chi(G,k) + \frac{k-2}{k}\chi(G) + \frac{2}{k}$$

$\diamondsuit$

Remarque. Les deux algorithmes décrits ci-dessus donnent une 2-approximation pour le problème P_1 quelque soit $k \in \mathbb{N}$.

Pour conclure ce chapitre, nous pouvons rappeler les avancées que nous avons réalisées dans l'étude de la complexité du problème P_1 pour les graphes d'intervalles. Soit $G = (V, E)$ un graphe d'intervalles avec $|V| = n$ et $k \in \mathbb{N}$:

- Si $S_1, \ldots, S_{\chi(G)}$ est une partition de G par un minimum de stables tel que $\forall S_i$, $|S_i| \leq k$, alors P_1 est polynômial pour G et $\chi(G, k) = \chi(G)$,
- Si $S_1, \ldots, S_{\chi(G)}$ est une partition de G par un minimum de stables tel que $\forall S_i$, $|S_i| \geq k$, alors P_1 est polynômial pour G et $\chi(G, k) = \lceil \frac{n}{k} \rceil$,
- Si G est sans $K_{1,3}$ (tous les intervalles sont propres ou unitaires) alors P_1 est polynômial pour G et $\chi(G, k) = \max(\chi(G), \lceil \frac{n}{k} \rceil)$,
- P_1 peut être résolu en $O(n \log n)$ pour $k = 2$ et il est donc possible d'exhiber de la partition de G par un minimum de stables, tous les stables de taille 1 appartenant à une solution optimale.

Ainsi, nous pouvons réduire la recherche de la complexité du problème pour $k = 3$ à la recherche de la complexité du problème P_1 (tel que nous l'avons défini dans la Section 2) avec comme nouvelle entrée :

Entrée : $G = (V, E)$ graphe d'intervalles, dont certains intervalles ne sont pas propres, avec $S_1, \ldots, S_{\chi(G)}$ une partition $\mathcal{S}$ de G par un minimum de stables tel que $\chi(G) = 3q$, $q \in \mathbb{N}$ et que le nombre de stables de $\mathcal{S}$ de taille 2 (respectivement de taille 3, de taille 4) soit exactement q (c'est-à-dire tel que la partition de G par un minimum de stables contienne exactement q stables de taille 2, q stables de taille 3 et q stables de taille 4), $k = 3$, $k' \in \mathbb{N}$.

C'est-à-dire Nous allons maintenant aborder les problèmes MP_2 et MP_3 au travers du Chapitre 2.

2. SUR UNE EXTENSION DU PROBLÈME DE BIN-PACKING

Le problème de Bin-Packing est un célèbre problème de recherche opérationnelle. Il a été très étudié dans les années 70 mais aujourd'hui encore, son analyse est à l'origine de nouvelles approches en algorithmique [8].
De nombreuses variantes du Bin-Packing ont été proposées et étudiées [8], nous allons ici en décrire une qui semble naturelle : le Bin-Packing avec contraintes de compatibilité entre les objets. Ainsi, lorsque les objets sont représentés par des intervalles sur une droite, ce même problème devient effectivement équivalent au deuxième problème MP_2 défini dans l'introduction.
Nous avons cependant pensé qu'une étude plus générale serait intéressante.

2.1 DÉFINITION ET COMPLEXITÉ DU PROBLÈME GÉNÉRAL

Définissons tout d'abord le problème classique de Bin-Packing.

Problème BP : *Bin-Packing*

Entrée : *$\mathcal{O} = \{O_1, \ldots, O_n\}$ un ensemble de n objets, une fonction $s : \mathcal{O} \longmapsto \mathbb{R}$ associant à chaque objet un nombre réel représentant sa taille, $D \in \mathbb{N}$ tel que $\forall O_i \in \mathcal{O}$, $s(O_i) \leq D$ et $k' \in \mathbb{N}$.*

Question :*Existe-t-il une partition de $\mathcal{O}$ en k' boites $B_1, \ldots, B_{k'}$ tel que la somme des tailles des objets de chaque boîte soit au plus égale à D ?*

Ce problème est $\mathcal{NP}$-complet [6]. Il existe néanmoins de nombreuses heuristiques le résolvant avec de bons facteurs approximations [8], nous en proposerons une dans les sections suivantes.
Dans ce problème, n'importe quelles combinaisons d'objets sont autorisées. Supposons maintenant que seulement certaines combinaisons d'objets soient valides : cela nous amène à définir ce que nous appe-

lerons un hypergraphe de compatibilité d'un ensemble d'objets.

Définition 2.1 *Soit $\mathcal{O} = \{O_1, \ldots, O_n\}$ un ensemble de n objets. L'hypergraphe de compatibilité $H(\mathcal{O}) = (V, E)$ de l'ensemble des objets $\mathcal{O}$ peut être défini de la manière suivante :*
– *$V = \mathcal{O}$*
– *$E = \{\forall \mathcal{O}' \ / \ \mathcal{O}' \in \mathcal{P}(\mathcal{O})$ et les objets de $\mathcal{O}'$ sont compatibles$\}$.*
L'hypergraphe d'incompatibilité de $\mathcal{O}$ sera alors défini comme étant $\overline{H}(\mathcal{O})$.

Nous pouvons à présent définir de manière précise l'extension du problème BP.

Problème BPC : *Bin-Packing avec contraintes de compatibilité entre les objets*

Entrée : *$\mathcal{O} = \{O_1, \ldots, O_n\}$ un ensemble de n objets, $H(\mathcal{O}) = (V, E)$ son hypergraphe de compatibilité (ou $\overline{H}(\mathcal{O})$ son hypergraphe d'incompatibilité), une fonction $s : \mathcal{O} \longmapsto \mathbb{R}$ associant à chaque objet un nombre réel représentant sa taille, $D \in \mathbb{R}$ tel que $\forall O_i \in \mathcal{O}$, $s(O_i) \leq D$ et $k' \in \mathbb{N}$.*

Question : *Existe-t-il une partition de $\mathcal{O}$ en k' boites $B_1, \ldots, B_{k'}$ tel que la somme des tailles des objets de chaque boîte soit au plus égale à D et que chaque boîte contienne un ensemble d'objets compatibles ?*

Caractérisons la complexité du problème BPC.

Proposition 2.1 *Le problème BPC est $\mathcal{NP}$-complet quelque soit $H(\mathcal{O})$. De plus, il est non-α-approximable en temps polynômial avec $\alpha \in \mathbb{R}$.*

Preuve. Le problème BPC contient le problème BP ($H(\mathcal{O})$ l'hypergraphe complet). Il contient aussi le problème de la coloration d'un hypergraphe (coloration de $\overline{H}(\mathcal{O})$) lorsque $\forall O_i \in \mathcal{O}$, $s(O_i) = 1$ et $D = n$ qui est un problème $\mathcal{NP}$-complet [9] et non-α-approximable en temps polynômial avec $\alpha \in \mathbb{R}$ [6]. $\diamondsuit$

Nous allons nous intéresser à une certaine classe de problème BPC : la classe des problèmes BPC où la relation de compatibilité entre les objets est héréditaire, c'est-à-dire où la relation de compatibilité vérifie les deux conditions suivantes équivalentes :
(i) $\forall \mathcal{O}' \in \mathcal{P}(\mathcal{O})$, $\mathcal{O}'$ compatibles si et seulement si $\forall \mathcal{P}(\mathcal{O}')$, $\mathcal{P}(\mathcal{O}')$ compatibles,
(ii) $H(\mathcal{O})$ est un graphe et $\forall \mathcal{O}' \in \mathcal{P}(\mathcal{O})$, $\mathcal{O}'$ compatibles si et seulement si $\mathcal{O}'$ forme une clique dans $H(\mathcal{O})$, c'est-à-dire si et seulement si les objets de $\mathcal{O}'$ sont deux-à-deux compatibles.

2.2 LA CLASSE DES PROBLÈMES BPC OÙ $H(O)$ EST UN GRAPHE

Définition 2.2 *Soit $\mathcal{O} = \{O_1, \ldots, O_n\}$ un ensemble de n objets dont les contraintes de compatiblité sont héréditaires. Le graphe de compatibilité $G(\mathcal{O}) = (V, E)$ de l'ensemble des objets $\mathcal{O}$ peut être défini de la manière suivante :*
– *$V = \mathcal{O}$,*
– *$E = \{\forall e = O_i O_j \ / \ \{O_i, O_j\} \in \mathcal{O} \times \mathcal{O}$, $O_i \neq O_j$, O_i et O_j compatibles$\}$.*
Le graphe d'incompatibilité de $\mathcal{O}$ sera alors défini comme étant $\overline{G}(\mathcal{O})$.

Nous aurons tendance, dans la suite de notre analyse, à davantage utiliser le graphe d'incompatibilité $\overline{G}(\mathcal{O})$.

Problème $\mathbf{P_2}$: *Bin-Packing avec contraintes de compatibilité héréditaires entre les objets*

Entrée : *$\mathcal{O} = \{O_1, \ldots, O_n\}$ un ensemble de n objets, $G(\mathcal{O}) = (V, E)$ son graphe de compatibilité (ou $\overline{G}(\mathcal{O})$ son graphe d'incompatibilité), une fonction $s : \mathcal{O} \longmapsto \mathbb{R}$ associant à chaque objet un nombre réel représentant sa taille, $D \in \mathbb{R}$ tel que $\forall O_i \in \mathcal{O}$, $s(O_i) \leq D$ et $k' \in \mathbb{N}$.*

Question : *Existe-t-il une partition de $\mathcal{O}$ en k' boites $B_1, \ldots, B_{k'}$ tel que la somme des tailles des objets de chaque boîte soit au plus égale à D et que chaque boîte contienne un ensemble d'objets compatibles ?*

Caractérisons maintenant la complexité du problème P_2.

Proposition 2.2 *Le problème P_2 est $\mathcal{NP}$-complet quelque soit $G(\mathcal{O})$. De plus, il est α-approximable en temps polynômial avec $\alpha \in \mathbb{R}$ si et seulement si le problème de la coloration de $\overline{G}(\mathcal{O})$ (respectivement le problème de la couverture de $G(\mathcal{O})$ par des cliques) est β-approximable en temps polynômial avec $\beta \in \mathbb{R}$ et $\alpha \geq \beta$.*

Preuve. Pour le premier point, il suffit de voir que le problème P_2 contient le problème BP ($G(\mathcal{O})$ le graphe complet).
Pour le second point, nous allons démontrer l'implication dans un sens puis dans l'autre.
$\Longrightarrow$:
En posant $\forall O_i \in \mathcal{O}$, $s(O_i) = 1$ et $D = n$ dans le problème P_2, nous pouvons obtenir une α-approximation en temps polynômial avec $\alpha \in \mathbb{R}$ pour le problème de la coloration du graphe $\overline{G}(\mathcal{O})$ (respectivement le problème de la couverture du graphe $G(\mathcal{O})$ par des cliques).
$\Longleftarrow$:
Nous allons donner une preuve algorithmique de cette implication. Par commodité, nous noterons ici $\overline{G}(\mathcal{O})$ par G.
Supposons que le problème de la coloration du graphe $\overline{G}(\mathcal{O})$ (respectivement le problème de la couverture du graphe $G(\mathcal{O})$ par des cliques) soit β-approximable en temps polynômial avec $\beta \in \mathbb{R}$. Il existe donc un algorithme A polynômial colorant G en $\chi_{ALG}(G)$ couleurs tel que $\chi_{ALG}(G) \leq \beta\chi(G)$ avec $\beta \in \mathbb{R}$.
Nous pouvons alors décrire l'algorithme suivant pour le problème P_2.

Algorithme 2.1 :

Entrée :
Une instance du problème P_2 ;

Sortie :
Un ensemble de boîte $\mathcal{B}$ définissant une partition de $\mathcal{O}$ et solution du problème P_2 ;

Etape 1 :
Colorier G par l'algorithme A en $\chi_{ALG}(G)$ couleurs tel que $\chi_{ALG}(G) \leq \beta\chi(G)$ avec $\beta \in \mathbb{R}$;

Etape 2 :
Pour chaque stable S_i défini par les objets de couleur i faire
Appliquer l'Algorithme Packing pour le problème BP sur l'ensemble des objets de S_i ;
Inclure l'ensemble des boîtes ainsi obtenu dans $\mathcal{B}$;

Etape 3 :
Retourner l'ensemble des boîtes $\mathcal{B}$;

Décrivons donc notre Algorithme Packing pour le problème BP.

Algorithme Packing :

Entrée :
Une instance du problème BP ;

Sortie :
Un ensemble de boîte $\mathcal{B}$ définissant une partition de $\mathcal{O}$ et solution du problème BP ;

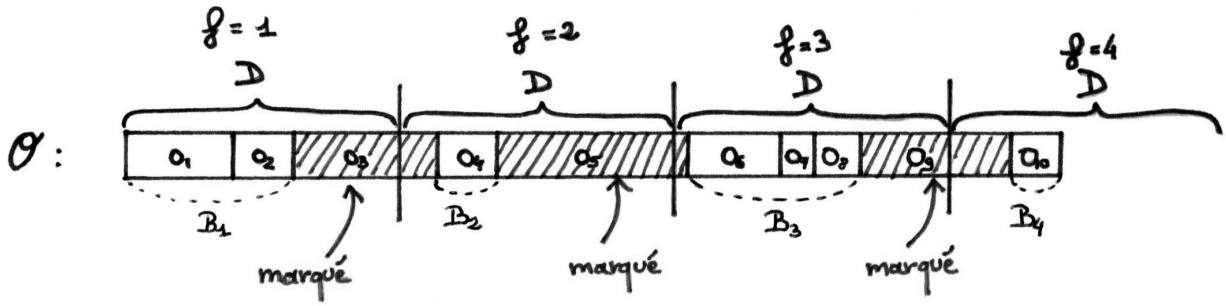
Etape 1 :
Soit $L = 0$ et $f = 1$;
Pour i de 1 à n faire
$L = L + s(O_i)$;
Si $L > f \times D$ faire
Marquer O_i et $f = f + 1$;

Etape 2 :
Soit B la boîte courante, $B = \emptyset$ et $i = 1$;
Pour chaque $O \in \mathcal{O}$ faire
Si O non marqué faire
$\mathcal{O} = \mathcal{O} \setminus \{O\}$, $B = B \cup \{O\}$;
Sinon
Inclure B dans $\mathcal{B}$;
Soit B une nouvelle boîte courante, $B = \emptyset$;

Etape 3 :
Pour chaque $O \in \mathcal{O}$ faire
$\mathcal{O} = \mathcal{O} \setminus \{O\}$;
Soit $s(B)$ la somme des tailles des objets de la boîte B ;
Si $s(B \cup \{O\}) \leq D$ faire
$B = B \cup \{O\}$;
Sinon faire
Inclure B dans $\mathcal{B}$;
Soit B une nouvelle boîte courante et $B = \{O\}$;

Etape 4 :
Retourner l'ensemble des boîtes $\mathcal{B}$;

La figure ci-dessous illustre le principe de l'Algorithme Packing.



L'Algorithme Packing étant linéaire en le nombre d'objets considérés, l'Algorithme 2.1 peut donc s'exécuter en temps polynômial.

Nous pouvons donc maintenant analyser le résultat de l'Algorithme 2.1.
Soit,

- $B_{i_{ALG}}$ le nombre de boîtes retournées par le Bin-Packing des objets de couleur i,
- $B_{i_{FRACT}}$ la somme des tailles des objets de couleur i divisée par D,
- B_{ALG} le nombre de boîtes retournées par l'Algorithme 2.1,
- B_{FRACT} le somme des tailles des objets de O divisée par D,
- B_{OPT} le nombre de boîtes optimal du problème P_2.

L'Algorithme Packing est tel que :

$$B_{i_{ALG}} \leq 2\lfloor B_{i_{FRACT}} \rfloor + 1$$

Par conséquent,

$$B_{ALG} \leq \sum_{i=1}^{\chi_{ALG}(G)} (2B_{i_{FRACT}} + 1)$$

Or $B_{FRACT} = \sum_{i=1}^{\chi_{ALG}(G)} B_{i_{FRACT}}$
Donc

$$B_{ALG} \leq 2B_{FRACT} + \beta\chi(G)$$

Et comme $B_{FRACT} \leq B_{OPT}$ et $\chi(G) \leq B_{OPT}$, nous obtenons

$$B_{ALG} \leq (2 + \beta)B_{OPT} \; avec \; \beta \in \mathbb{R}$$

Il nous suffit alors de poser $\alpha = 2 + \beta$, $\beta \in \mathbb{R}$ pour obtenir l'α-approximation souhaitée avec $\alpha \in \mathbb{R}$. $\diamondsuit$

Remarque. La proposition 2.2 peut être étendue au problème BPC par une preuve similaire.

Corollaire 2.1 *Soit B_{ALG} le nombre de boîtes retournées par l'Algorithme 2.1. Soit $G = \overline{G}(\mathcal{O})$. Si G est un graphe parfait alors $B_{ALG} \leq 3B_{OPT}$. En Particulier, le Problème P_2 est 3-approximable pour les graphes d'intervalles en $O(n \log n)$. De plus, si $\forall O \in \mathcal{O}$, $\theta_1 D \leq s(O) \leq \theta_2 D$ avec $\{\theta_1, \theta_2\} \in [0,1]^2$, $\theta_1 \leq \theta_2$, alors $B_{ALG} \leq (1 + (1 + \theta_2)(1 - \theta_1))B_{OPT}$*

Preuve. Les deux premières affirmations sont immédiates par la preuve de la Proposition 2.2 et le Théorème 1.5. Pour la dernière affirmation, il faut voir que l'Algorithme Packing est tel que

$$B_{i_{ALG}} \leq (1 + \theta_2)\lfloor B_{i_{FRACT}} \rfloor + 1 \;\; et \;\; \lfloor B_{i_{FRACT}} \rfloor \leq B_{i_{FRACT}} - \theta_1$$

Cela nous permet alors d'obtenir

$$B_{ALG} \leq (1 + \theta_2)B_{FRACT} + \beta(1 - \theta_1(1 + \theta_2))\chi(G)$$

puis de conclure en posant $\beta = 1$. $\diamondsuit$

Nous allons à présent voir comment améliorer cette approximation de manière simple dans la partie suivante.

2.3 UNE $\frac{11}{4}$-APPROXIMATION DU PROBLÈME P_2 POUR LES GRAPHES D'INTERVALLES

Nous allons maintenant présenter l'Algorithme 2.2 qui améliore la borne obtenue par l'Algorithme 2.1 tout en conservant son temps d'exécution.

Algorithme 2.2 :

Entrée :
Une instance du problème P_2 ;

Sortie :
Un ensemble de boîte $\mathcal{B}$ définissant une partition de $\mathcal{O}$ et solution du problème P_2 ;

Etape 1 :
Colorier G par l'algorithme A en $\chi_{ALG}(G)$ couleurs tel que $\chi_{ALG}(G) \leq \beta\chi(G)$ avec $\beta \in \mathbb{R}$;
Soit $c(O)$ la couleur d'un objet O ;

Etape 2 :
Partitionner les objets $\mathcal{O}$ en deux ensembles :
- $\mathcal{O}_1$ l'ensemble des objets de $\mathcal{O}$ dont la taille est $> \frac{D}{2}$,
- $\mathcal{O}_2$ l'ensemble des objets de $\mathcal{O}$ dont la taille est $\leq \frac{D}{2}$;

Etape 3 :
Pour chaque objet O de $\mathcal{O}_1$ faire
Mettre O dans une boîte B ;
Compléter en temps polynômial la boîte B par des objets de $\mathcal{O}_2$ de couleur $c(O)$;
Inclure B dans $\mathcal{B}$;

Etape 4 :
Appliquer l'Algorithme 2.1 sur l'ensemble des objets restants dans $\mathcal{O}_2$;
Inclure l'ensemble des boîtes ainsi obtenu dans $\mathcal{B}$;

Etape 5 :
Retourner l'ensemble des boîte $\mathcal{B}$;

Proposition 2.3 *L'Algorithme 2.2 donne une α-approximation du problème P_2 en temps polynômial tel que $\alpha \leq \frac{7}{4} + \beta$ avec $\beta \in \mathbb{R}$.*

Preuve. Soit $G = \overline{G}(\mathcal{O})$. L'Etape 2 pouvant se faire en temps linéaire, l'Algorithme 2.2 reste polynômial.
En reprenant la notation définie dans la preuve précédente, nous allons distinguer les boîtes contenant un objet de taille strictement supérieure à $\frac{D}{2}$ (notée B_1) des autres (notée B_2).
Analysons tout d'abord $B_{2_{FRACT}}$:

$$B_{2_{i_{FRACT}}} = \lfloor B_{2_{i_{FRACT}}} \rfloor + \epsilon \ \ avec \ \ \epsilon \in [0, 1[$$

Deux cas se présentent alors à nous, lors de l'analyse du packing produit par l'Algorithme Packing.
– Si $\epsilon \leq \frac{1}{2}$, nous aurons :

$$B_{2_{i_{ALG}}} \leq \lfloor B_{2_{i_{FRACT}}} \rfloor + \lfloor \frac{\lfloor B_{2_{i_{FRACT}}} \rfloor}{2} \rfloor + 1$$

Par conséquent,

$$B_{2_{i_{ALG}}} \leq \frac{3}{2} B_{2_{i_{FRACT}}} + 1$$

– Si $\epsilon > \frac{1}{2}$, nous aurons :

$$B_{2_{i_{ALG}}} \leq \lfloor B_{2_{i_{FRACT}}} \rfloor + \lceil \frac{\lfloor B_{2_{i_{FRACT}}} \rfloor}{2} \rceil + 1$$

Soit,

$$B_{2_{i_{ALG}}} \leq B_{2_{i_{FRACT}}} - \epsilon + \lceil \frac{B_{2_{i_{FRACT}}} - \epsilon}{2} \rceil + 1$$

Maintenant,
– Si $B_{2_{i_{FRACT}}} - \epsilon = 2t$ avec $t \geq 0$ alors

$$B_{2_{i_{ALG}}} \leq \frac{3}{2} B_{2_{i_{FRACT}}} + 1$$

– Si $B_{2_{i_{FRACT}}} - \epsilon = 2t + 1$ avec $t \geq 0$ alors

$$B_{2_{i_{ALG}}} \leq B_{2_{i_{FRACT}}} + \frac{3}{2} - \epsilon + \frac{B_{2_{i_{FRACT}}} - \epsilon}{2}$$

Or $\epsilon > \frac{1}{2}$, par conséquent,

$$B_{2_{i_{ALG}}} \leq \frac{3}{2} B_{2_{i_{FRACT}}} + 1$$

Nous pouvons donc en conclure que, dans tous les cas, $B_{2_{i_{ALG}}} \leq \frac{3}{2} B_{2_{i_{FRACT}}} + 1$ et donc que,

$$B_{2_{ALG}} \leq \frac{3}{2} B_{2_{FRACT}} + \chi_{ALG}(G) \quad (i)$$

Par ailleurs, $B_{1_{FRACT}} \geq \frac{|\mathcal{O}_1|}{2}$, donc,

$$B_{2_{FRACT}} \leq B_{FRACT} - \frac{|\mathcal{O}_1|}{2} \quad (ii)$$

Ainsi, par (i) et (ii), nous obtenons,

$$B_{2_{ALG}} \leq \frac{3}{2}B_{FRACT} - \frac{3}{4}|\mathcal{O}_1| + \chi_{ALG}(G)$$

Et en ajoutant le nombre de boîtes obtenues à l'Etape 3,

$$B_{ALG} \leq \frac{3}{2}B_{FRACT} + \frac{1}{4}|\mathcal{O}_1| + \chi_{ALG}(G)$$

Or $B_{FRACT} \leq B_{OPT}$, $|\mathcal{O}_1| \leq B_{OPT}$ et $\chi(G) \leq B_{OPT}$, donc

$$B_{ALG} \leq (\frac{7}{4} + \beta)B_{OPT} \; avec \; \beta \in \mathbb{R}$$

Il suffit donc de poser $\alpha = \frac{7}{4} + \beta$ avec $\beta \in \mathbb{R}$ pour obtenir l'α-approximation souhaitée avec $\alpha \in \mathbb{R}$. $\diamondsuit$

Corollaire 2.2 *Soit B_{ALG} le nombre de boîtes retournées par l'Algorithme 2.2. Soit $G = \overline{G}(\mathcal{O})$. Si G est un graphe parfait alors $B_{ALG} \leq \frac{11}{4}B_{OPT}$. En particulier, le Problème P_2 est $\frac{11}{4}$-approximable pour les graphes d'intervalles en $O(n \log n)$.*

Preuve. Le Théorème 1.5, la Proposition 2.3 et le fait que l'Etape 3 peut s'effectuer en $O(n \log n)$ nous permettent de conclure. $\diamondsuit$

Nous allons maintenant aborder un problème plus général que le problème P_2, en lui rajoutant la contrainte du nombre d'objets par boîte limité à k.

2.4 LE PROBLÈME P_3 : UNE EXTENSION DES PROBLÈMES P_1 ET P_2

Nous allons ici formuler le problème P_3 en reprenant les contraintes des deux problèmes P_1 et P_2. Celui-ci correspondra donc au problème MP_3 lorsque $\overline{G}(\mathcal{O})$ sera un graphe d'intervalles.

Problème P_3 : *Bin-Packing avec contraintes de compatiblité héréditaires entre les objets et nombre d'objets limité par boîte*

Entrée : *$\mathcal{O} = \{O_1, \ldots, O_n\}$ un ensemble de n objets, $G(\mathcal{O}) = (V, E)$ son graphe de compatibilité (ou $\overline{G}(\mathcal{O})$ son graphe d'incompatibilité), une fonction $s : \mathcal{O} \longmapsto \mathbb{R}$ associant à chaque objet un nombre réel représentant sa taille, $D \in \mathbb{R}$ tel que $\forall O_i \in \mathcal{O}$, $s(O_i) \leq D$, $k \in \mathbb{N}$ et $k' \in \mathbb{N}$.*

Question : *Existe-t-il une partition de $\mathcal{O}$ en k' boites $B_1, \ldots, B_{k'}$ tel que la somme des tailles des objets de chaque boîte soit au plus égale à D et que chaque boîte contienne un ensemble d'au plus k objets compatibles ?*

Comme précédemment, nous allons faire une étude générale du problème pour en déduire ensuite des résultats pour les graphes d'intervalles. Précisons qu'une étude de ce problème où $G(\mathcal{O}) = K_n$ (le graphe complet) est mentionnée dans [8].
Le fait de rajouter cette contrainte sur le nombre d'objets par boîte agit sur la complexité du problème. En effet, nous pouvons établir la proposition suivante.

Proposition 2.4 *Pour $k = 2$, le problème P_3 est polynômial quelque soit $G(\mathcal{O})$.*

Preuve. Nous pouvons donner l'algorithme polynômial suivant pour résoudre le problème P_3 lorsque $k = 2$.

Algorithme 2.3 :

Entrée :
Une instance du problème P_3 avec $k = 2$;

Sortie :
Un ensemble de boîte $\mathcal{B}$ définissant une partition de $\mathcal{O}$ et solution du problème P_3 avec $k = 2$;

Etape 1 :
Pour chaque arête $e \in E$ de $G(\mathcal{O})$ faire
Soit $e = O_iO_j$, $\{O_i, O_j\} \in V \times V$ avec $i \neq j$;
Si $s(O_i) + s(O_j) > D$ faire
$E = E \setminus e$;

Etape 2 :
Appliquer l'Algorithme d'Edmonds pour trouver un couplage maximum $\mathcal{M}$ dans $G(\mathcal{O})$;
Pour chaque couple $\{O_i, O_j\} \in \mathcal{M}$ faire
Placer O_i et O_j dans une même boîte et inclure celle-ci dans $\mathcal{B}$;
Pour chaque $O \in \mathcal{O}$ tel que $O \notin \mathcal{M}$ faire
Placer O dans une boîte et inclure celle-ci dans $\mathcal{B}$;

Etape 3 :
Retourner l'ensemble des boîtes $\mathcal{B}$;

La complexité de l'Algorithme d'Edmonds étant en $O(n^3)$ [5], l'Algorithme 2.3 peut s'exécuter en $O(n^3)$. Vérifions à présent la validité de cet algorithme. Soit G' le graphe obtenu après l'Etape 1. G' représente en fait la combinatoire de tous les couples d'objets, soit $O_i, O_j \in \mathcal{O}$ avec $i \neq j$, compatibles et respectant la contrainte $s(O_i) + s(O_j) \leq D$ (c'est-à-dire O_i et O_j peuvant être placés dans une même boîte). Par conséquent, un couplage maximum dans G' définira bien une solution optimale de notre problème. $\diamondsuit$

Corollaire 2.3 *Pour $k = 2$ et $\overline{G}(\mathcal{O})$ graphe d'intervalles, le problème P_3 peut être résolu en $O(n^3)$.*

Proposition 2.5 *Pour $k \geq 3$, le problème P_3 est $\mathcal{NP}$-complet quelque soit $G(\mathcal{O})$. De plus, il est α-approximable en temps polynômial avec $\alpha \in \mathbb{R}$ si et seulement si le problème de la coloration de $\overline{G}(\mathcal{O})$ (respectivement le problème de la couverture de $G(\mathcal{O})$ pardes cliques) est β-approximable en temps polynômial avec $\beta \in \mathbb{R}$ et $\alpha \geq \beta$.*

Preuve. Pour le premier point, il suffit de voir que le problème P_3 contient le problème de Tripartition ($G(\mathcal{O})$ le graphe complet et $k = 3$). Rappelons tout de même le problème de Tripartition de manière brève.

Problème : *Tripartition*

Entrée : *Un ensemble $A = \{a_i \ / \ i \in I\}$ de $3q$ nombres entiers, $q \in \mathbb{N}$, avec $\sum_{i \in I} a_i = qD$ et, $\forall i$, $\frac{D}{4} < a_i < \frac{3D}{4}$, $D \in \mathbb{N}$.*

Question : *Existe-t-il une partition de A en q sous-ensembles, chacun de cardinal 3 et de poids D ?*

Pour le second point, nous allons démontrer l'implication dans un sens puis dans l'autre.
$\Longrightarrow$:
En posant $\forall O_i \in \mathcal{O}$, $s(O_i) = 1$ et $D = k = n$ dans le problème P_3, nous pouvons obtenir une α-approximation en temps polynômial avec $\alpha \in \mathbb{R}$ pour le problème de la coloration du graphe $\overline{G}(\mathcal{O})$ (respectivement le problème de la couverture du graphe $G(\mathcal{O})$ par des cliques).

$\Longleftarrow$:
Nous allons donner une preuve algorithmique de cette implication. Par commodité, nous noterons $\overline{G}(\mathcal{O})$ par G.
Supposons que le problème de la coloration du graphe $\overline{G}(\mathcal{O})$ (respectivement le problème de la couverture du graphe $G(\mathcal{O})$ par des cliques) soit β-approximable en temps polynômial avec $\beta \in \mathbb{R}$. Il existe donc un algorithme A polynômial colorant G en $\chi_{ALG}(G)$ couleurs tel que $\chi_{ALG}(G) \leq \beta\chi(G)$ avec $\beta \in \mathbb{R}$.
Nous pouvons alors décrire l'algorithme suivant pour le problème P_3.

Algorithme 2.4 :

Entrée :
Une instance du problème P_3 ;

Sortie :
Un ensemble de boîte $\mathcal{B}$ définissant une partition de $\mathcal{O}$ et solution du problème P_3 ;

Etape 1 :
Appliquer l'Algorithme 2.1 sur $\mathcal{O}$ pour le problème P_2 ;
Soit $\mathcal{B}'$ l'ensemble des boîtes retourné par l'Algorithme 2.1 ;

Etape 2 :
Pour chaque boîte $B \in \mathcal{B}'$ faire
Si $|B| \leq k$ faire
Inclure B dans $\mathcal{B}$;
Sinon faire
Soit $q = \lfloor \frac{|B|}{k} \rfloor$ et $r = |B| - q \times k$;
Mettre, par paquet de k, $q \times k$ objets de B dans q boîtes et les r objets restants de B dans une boîte ;
Inclure ces boîtes dans $\mathcal{B}$;

Etape 3 :
Retourner l'ensemble des boîtes $\mathcal{B}$;

L'Etape 2 pouvant s'effectuer en temps linéaire, l'Algorithme 2.4 peut s'exécuter en temps polynômial. Analysons donc son résultat.
Soit $B_{ALG_D} = |\mathcal{B}'|$ et $B_{ALG_{D,K}} = |\mathcal{B}|$. De la même manière, soit B_{OPT_D} le nombre optimal de boîtes solutions du problème P_2 à l'Etape 1 et $B_{OPT_{D,K}}$ le nombre optimal de boîtes solutions du problème P_3.
Clairement, nous avons,

$$B_{ALG_{D,K}} = \sum_{j=1}^{B_{ALG_D}} \lceil \frac{|B_j|}{k} \rceil$$

$$B_{ALG_{D,K}} \leq \sum_{j=1}^{B_{ALG_D}} \frac{|B_j|}{k} + B_{ALG_D}$$

$$B_{ALG_{D,K}} \leq \frac{n}{k} + B_{ALG_D}$$

Or $\frac{n}{k} \leq B_{OPT_{D,K}}$, $B_{OPT_D} \leq B_{OPT_{D,K}}$ et $\chi(G) \leq B_{OPT_{D,K}}$, donc

$$B_{OPT_{D,K}} \leq (3+\beta) B_{OPT_{D,K}} \; avec \; \beta \in \mathbb{R}$$

En posant $\alpha = 3 + \beta$ avec $\beta \in \mathbb{R}$, nous obtenons l'α-approximation souhaitée. $\diamondsuit$

Remarque. La proposition 2.5 peut être étendue par une preuve similaire au problème P_3 considérant, non pas des graphes, mais des hypergraphes de compatibilité.

Corollaire 2.4 *Soit B_{ALG} le nombre de boîtes retournées par l'Algorithme 2.4. Soit $G = \overline{G}(\mathcal{O})$. Si G est un graphe parfait alors $B_{ALG} \leq 4B_{OPT}$. En Particulier, le Problème P_3 est 4-approximable pour les graphes d'intervalles en $O(n \log n)$.*

Preuve. Immédiate par le Théorème 1.5, le Corollaire 2.1 et la Proposition 2.5. $\diamondsuit$

Corollaire 2.5 *Soit,*

Etape 1' :
Appliquer l'Algorithme 2.2 sur $\mathcal{O}$ pour le problème P_2 ;
Soit $\mathcal{B}'$ l'ensemble des boîtes retourné par l'Algorithme 2.2 ;

Etape 3' :
Pour chaque objet O de $\mathcal{O}_1$ faire
Mettre O dans une boîte B ;
Compléter en temps polynômial la boîte B par au plus $k-1$ objets de $\mathcal{O}_2$ de couleur $c(O)$;
Inclure B dans $\mathcal{B}$;

Soit, dans l'Algorithme 2.2, l'Etape 3 remplacée par l'Etape 3' et dans l'Algorithme 2.4, l'Etape 1 remplacée par l'Etape 1'. Soit B_{ALG} le nombre de boîtes retournées par ce dernier et $G = \overline{G}(\mathcal{O})$.
Si G est colorable en $\chi_{ALG}(G)$ couleurs tel que $\chi_{ALG}(G) \leq \beta\chi(G)$ avec $\beta \in \mathbb{R}$ alors $B_{ALG} \leq (\frac{11}{4} + \beta)B_{OPT}$.
Si G est un graphe parfait alors $B_{ALG} \leq \frac{15}{4}B_{OPT}$. En particulier, le Problème P_3 est $\frac{15}{4}$-approximable pour les graphes d'intervalles en $O(n \log n)$.

Preuve. Similaire aux preuves de la Proposition 2.5 et du Corollaire 2.4. $\diamondsuit$

Nous allons maintenant voir comment encore améliorer les facteurs d'approximation établis par les algorithmes précédents. Voici, comme précedemment, un algorithme se basant sur l'Algorithme 2.1.

Algorithme 2.5 :

Entrée :
Une instance du problème P_3 ;

Sortie :
Un ensemble de boîte $\mathcal{B}$ définissant une partition de $\mathcal{O}$ et solution du problème P_3 ;

Etape 1 :
Colorier G par l'algorithme A en $\chi_{ALG}(G)$ couleurs tel que $\chi_{ALG}(G) \leq \beta\chi(G)$ avec $\beta \in \mathbb{R}$;

Etape 2 :
Pour chaque stable S_i défini par les objets de couleur i faire
Appliquer l'Algorithme Packing pour le problème BP sur l'ensemble des objets de S_i ;
Soit $\mathcal{B}'$ l'ensemble des objets retourné par celui-ci ;
Pour chaque boîte $B \in \mathcal{B}'$ faire
Si $|B| \leq k$ faire
Inclure B (*) dans $\mathcal{B}$;
Sinon faire
Marquer les $|B| - k$ plus petit objets de B ;
Mettre les objets non-marqués (**) de B dans une boîte et l'inclure dans $\mathcal{B}$;
Tant qu'il reste au moins k objets marqués faire
Disposer ceux-ci par paquets de k dans des boîtes et les inclure dans $\mathcal{B}$;
Mettre les objets restants dans une boîte et l'inclure dans $\mathcal{B}$;

Etape 3 :
Retourner l'ensemble des boîtes $\mathcal{B}$;

Proposition 2.6 *L'Algorithme 2.5 est un algorithme d'approximation polynômial pour le problème P_3 donnant une α-approximation tel que*

$$\alpha \leq \frac{3k-2}{k} + \beta \ \ avec \ \ \beta \in \mathbb{R}$$

Preuve. L'Etape 2 pouvant s'effectuer en $O(n \log n)$, l'algorithme peut s'exécuter en temps polynômial. Quant à sa validité, elle est basée sur le lemme suivant.

Lemme 2.1 *Les objets marqués ont une taille strictement inférieure à $\frac{D}{k}$.*

Preuve du Lemme 2.1. Supposons qu'une boîte contienne k' objets, $k' > k$. Il existe nécessairement $k' - k$ objets de taille strictement inférieure à $\frac{D}{k}$. En effet, si ce n'était pas le cas, cela signifirait que $k+1$ objets seraient de taille supérieure ou égale à $\frac{D}{k}$ et donc que la boîte contenant ces objets aurait une taille supérieure ou égale à $\frac{k+1}{k}D > D$, ce qui contredit l'hypothèse selon laquelle la somme des tailles des objets d'une boîte doit être inférieure ou égale à D. $\diamondsuit$

Grâce au Lemme 2.1, nous pouvons affirmer que notre algorithme est valide. Démontrons maintenant la borne énoncée dans la proposition.
Soit :

- B_{i_1} le nombre de boîtes dont les objets sont de couleur i et sont marqués ou non-marqués (**),
- B_{i_2} le nombre de boîtes (*) de couleur i,
- $|B_{i_1}|$ le nombre d'objets de l'ensemble des boîtes dont les objets sont de couleur i et sont marqués ou non-marqués (**),
- $|B_{i_2}|$ le nombre d'objets de l'ensemble des boîtes (*) de couleur i,
- $n_i = |B_{i_1}| + |B_{i_2}|$.

Nous aurons ainsi,

$$B_{i_{ALG}} = \lceil \frac{|B_{i_1}|}{k} \rceil + B_{i_2}$$

$$B_{i_{ALG}} \leq \frac{|B_{i_1}|}{k} + B_{i_2} + 1$$

Or $|B_{i_1}| = n_i + |B_{i_2}|$, donc

$$B_{i_{ALG}} \leq \frac{n_i}{k} + \frac{k-1}{k} B_{i_2} + 1$$

Ensuite $B_{i_2} \leq |\mathcal{B}'| - 1$ (sinon $B_{i_{ALG}} \leq 2B_{i_{FRACT}} + 1$) et $|\mathcal{B}'| \leq 2B_{i_{FRACT}} + 1$, par conséquent

$$B_{i_{ALG}} \leq \frac{n_i}{k} + 2\frac{k-1}{k} B_{i_{FRACT}} + 1$$

Ainsi, en sommant sur l'ensemble des couleurs,

$$B_{ALG} \leq \frac{n}{k} + 2\frac{k-1}{k} B_{FRACT} + \chi_{ALG}(G)$$

Et comme $\frac{n}{k} \leq B_{OPT}$, $B_{FRACT} \leq B_{OPT}$ et $\chi(G) \leq B_{OPT}$, nous obtenons

$$B_{ALG} \leq (\frac{3k-2}{k} + \beta) B_{OPT} \ avec \ \beta \in \mathbb{R}$$

En posant $\alpha = \frac{3k-2}{k} + \beta$ avec $\beta \in \mathbb{R}$, nous obtenons l'α-approximation souhaitée. $\diamondsuit$

Corollaire 2.6 *Soit B_{ALG} le nombre de boîtes retournées par l'Algorithme 2.5. Soit $G = \overline{G}(\mathcal{O})$.*
Si G est un graphe parfait alors $B_{ALG} \leq \frac{4k-2}{k} B_{OPT}$. En particulier, le Problème P_3 est $\frac{4k-2}{k}$-approximable

pour les graphes d'intervalles en $O(n \log n)$.

Preuve. Immédiate par le Théorème 1.5 et la Proposition 2.6. $\diamondsuit$

Remarque. Il est possible de reproduire cet technique d'approximation en se basant non pas sur l'Algorithme 2.1 mais sur l'Algorithme 2.2, ainsi l'α-approximation obtenue sera tel que $\alpha = \frac{11k-3}{4k} + \beta$ avec $\beta \in \mathbb{R}$. Notons que cette approximation sera meilleure que celle obtenue par l'Algorithme 2.5 pour $k \geq 5$.

Conclusion

Nous venons de voir, à travers ce travail de recherche, que la problématique de la planification d'horaires de travail, même modélisée simplement, restait un problème difficile. Les problème P_1, P_2 et P_3 pour les graphes d'intervalles (correspondants respectivement aux problèmes MP_1, MP_2 et MP_3) sont en effet tous trois $\mathcal{NP}$-complets. Cependant, notre étude a montré que la structure d'un ensemble d'intervalles offrait de bonnes propriétés pour la mise au point d'algorithmes d'approximation polynômiaux aux facteurs d'approximation α raisonnables ($\alpha \leq 4$ quelque soit le problème).

Malgré tout, cette première étude est loin d'être exhaustive. En effet, beaucoup de questions restent poser autour de ces problèmes. Il reste notamment à étudier l'existence de schémas d'approximation polynômiaux, à rechercher des bornes inférieures et à mettre au point des algorithmes améliorant les bornes supérieures existantes. Un autre point d'interrogation majeur reste la complexité du problème P_1 pour $k = 3$. Enfin, ce travail met en perspective d'autres modèles ou bien certaines variantes du problème de planification dont nous allons donner quelques exemples pour terminer.

Un quatrième problème, modélisant parfaitement la planification d'horaires de travail, peut être en effet défini comme suit.

Problème $\mathbf{MP_4}$:

Soit n tâches T_i, $i \in \{1, \ldots, n\}$ tel que $T_i = (d_i, f_i)$ avec $d_i \in \mathbb{N}$ date de début de T_i et $f_i \in \mathbb{N}$ date de fin de T_i. Partitionner ces n tâches en le minimum de vacations tel que les tâches appartenant à une vacation ne se chevauchent pas deux-à-deux et que chaque vacation ait une durée totale (= date de fin de sa dernière tâche − date de début de sa première) au plus égale à D, $D \in \mathbb{R}$.

Ce problème correspond à la coloration d'un graphe $G = (V, E_1 \cup E_2)$ où $G_1 = (V, E_1)$ est un graphe d'intervalles et $G_2 = (V, E_2)$ est un graphe de comparabilité. Il existe donc une 2-approximation immédiate du problème par les colorations successives de G_1 et de G_2 mais la complexité de ce problème reste à étudier précisemment.

Une autre approche serait l'étude des problèmes MP_1, MP_2 et MP_3 dans leur version "on-line" qui serait telle que les tâches apparaîtraient dans l'ordre défini par leurs indices de date de début. Dans ce cas-ci, nous pouvons déjà affirmer que certains algorithmes, basés sur la coloration "on-line" d'un graphe d'intervalles (pouvant se faire par l'Algorithme de coloration d'un ensemble d'intervalles présenté Chapitre 1 Section 1) et l'utilisation de méthodes "on-line" de packing tirées de l'Algorithme Packing (Chapitre 2 Section 2) que nous avons présenté, seraient respectivement 2-compétitif, 3-compétitif et 4-compétitif pour les versions "on-line" des problèmes MP_1, MP_2 et MP_3.

Enfin, le dernier exemple est le problème de replanification. En effet, une fois la planification des horaires effectuée, l'avance ou le retard de certains travaux peuvent perturber l'emploi du temps (par exemple, faits communs pour les employés des aérogares, les retards d'avions) et ainsi conduire à la nécessité de replanifier certains horaires selon certaines règles. La problématique de replanification ferait donc apparaître certains types de problèmes où se mèleraient méthodes d'approximation et analyses

probabilistes : problèmes intéressants tant du point de vue théorique qu'expérimental.

Références

[1] M.G. ANDREWS, M.J. ATALLAH, D.Z. CHEN, et D.T. LEE, *Parallel Algorithms for Maximum Matching in Complements of Interval Graphs and a Related Problems*, Algorithmica, Springer-Verlag (2000), 263–289.

[2] C. BERGE, *Hypergraphes*, Gauthier-Villars, 1987.

[3] H.L. BOADLANDER et K. JANSEN, *Restrictions of Graph Partitions Problems. Part I*, Elsevier, Theoretical Computer Science 148 (1995), 93–109.

[4] T. CORMEN, C. LEISERSON, et R.RIVEST, *Introduction à l'Algorithmique*, Dunod, 1994.

[5] J. EDMONDS, *Maximum Matching and a Polyedron with 0,1 vertices*, J. Research N.B.S. 69 B (1965), 125–130.

[6] M.R. GAREY et D.S. JOHNSON, *Computers and Intractability : a guide to the theory of NP-Completeness*, Freeman & Co, 1979.

[7] M.C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Computer Science and Applied Mathematics, Academic Press, London, 1980.

[8] E.G. COFFMAN Jr., M.R. GAREY, et D.S. JOHNSON, *Approximation Algorithms for Bin-Packing : a Survey*, PWS Publishing, Boston, 1996.

[9] R.M KARP, *Reduciblity among Combinatorial Problems*, Miller and Thatcher, Complexity of Computer Computations (1972), 85–104.

[10] A. PARTOUCHE, *Planification d'horaires de travail*, Thèse de Doctorat, Université Paris-Dauphine, 1998.