CONSERVATOIRE NATIONAL DES ARTS ET METIERS

THÈSE

présentée pour l'obtention du titre de

DOCTEUR DU CNAM

Spécialité : Informatique

par

Fethi JARRAY

RESOLUTION DE PROBLEMES DE TOMOGRAPHIE DISCRETE. APPLICATION A LA PLANIFICATION DE PERSONNEL.

Soutenue **le 23 novembre 2004** devant la commission d'examen :

Mme	Marie-Christine COSTA (CEDRIC-CNAM)	*Directrice de thèse*
M.	Christophe PICOULEAU (CEDRIC-CNAM)	*Co-encadrant de thèse*
Mme	Claire HANEN (LIP6-Univ. Paris 10)	*Rapporteur*
M.	Laurent VUILLON (LAMA-Univ. de Savoie)	*Rapporteur*
M.	Alain DAURAT (LSIIT-Univ. de Strasbourg)	
M.	Dominique de WERRA (EPFL, Suisse)	
M.	Eric JACQUET-LAGREZE (Eurodecision)	
M.	Stéphane NATKIN (CEDRIC-CNAM)	

REMERCIEMENTS

Je commence par remercier ma directrice de thèse Marie-Christine Costa et mon co-encadreur Christophe Picouleau pour m'avoir donné l'occasion d'entrer par la grande porte dans le monde fascinant de la reconstruction d'objets discrets. Leur confiance, leur soutien, leurs conseils et leur disponibilité m'ont beaucoup aidé à accomplir ce travail.

Laurent Vuillon et Claire Hanen ont eu la lourde tâche d'examiner cette thèse. Qu'ils trouvent ici l'expression de ma gratitude. Je remercie également Alain Daurat, Dominique De Werra, Eric Jacquet-Lagrèze et Stéphane Natkin qui ont participé à mon jury.

Je remercie tous les gens que j'ai croisés au laboratoire CEDRIC du CNAM et en particulier les membres de l'équipe Optimisation Combinatoire du CEDRIC.

Je ne peux manquer l'occasion d'adresser un remerciement particulier à toute ma famille : mon père, ma mère, mes frères, mes sœurs, mes neveux et mes nièces.

Enfin, un remerciement chaleureux à tous mes amis pour leur encouragement et tous les bons moments passés avec eux.

Table des matières

Table des figures

Liste des tableaux

Glossaire des problèmes

$MB(H,V)$:	Reconstruction d'une matrice respectant les projections orthogonales (H,V).
$M4adj(H)$:	Reconstruction d'une matrice respectant la projection horizontale et la contrainte de 4-adjacence.
$M4adj(H,V)$:	Reconstruction d'une matrice respectant les projections orthogonales et la contrainte de 4-adjacence.
$M6adj(H,V)$:	Reconstruction d'une matrice respectant les projections orthogonales et la contrainte de 6-adjacence.
$M8adj(H,V)$:	Reconstruction d'une matrice respectant les projections orthogonales et la contrainte de 8-adjacence.
$M2.4adj(H,V)$:	Reconstruction d'une matrice $2 \times n$ respectant les projections orthogonales et la contrainte de 4-adjacence.
$M3.4adj(H,V)$:	Reconstruction d'une matrice $3 \times n$ respectant les projections orthogonales et la contrainte de 4-adjacence.
$MAP(p,q)$:	Reconstruction d'une matrice (p,q) alternée périodique.
TkC :	Reconstruction d'un tableau k-coloré.
$PacDH(H,V)$:	Pavage de dominos horizontaux.
$PavDUH(H,V)$:	Pavage par des dominos unicolorés horizontaux.
$PacDBH(H,V)$:	Packing d'un tableau de dominos bicolorés horizontaux.
$PavDBH(H,V)$:	Pavage par des dominos bicolorés horizontaux.
$PavDBO(H,V)$:	Pavage d'un tableau par des dominos bicolorés orientés.
$PacBHMin(H,V)$:	Packing de b-barres horizontales selon les projections orthogonales et un écart minimal.
$PacBHMax(V)$:	Packing de b-barres horizontales respectant la projection verticale et un écart maximal.

$PacBHMax(H = c, V)$: Packing de b-barres horizontales selon la projection verticale, une projection horizontale constante et un écart maximal.

$PacBHMax(H)$: Packing de b-barres horizontales respectant la projection horizontale et un écart maximal.

$PacBHMaxu(H, V)$: Packing de b-barres horizontales respectant les projections orthogonales et un écart maximal unitaire.

$Pac23BH(H, V)$: Packing de dominos et triminos respectant les projection orthogonales des dominos et des triminos.

$PacHMax(H)$: Packing de barres horizontales respectant la projection horizontale et un écart maximal.

$PacHMax(V)$: Packing de barres horizontales respectant la projection verticale et un écart maximal.

$PacHC(H, V)$: Packing de barres horizontales respectant les projections orthogonales et un écart fixe.

$PacHMax(H, V)$: Packing de barres horizontales respectant les projections orthogonales et un écart maximal.

PCV : Problème de construction de vacations de travail. Il consiste à déterminer, pour une journée de travail, les horaires à effectuer par les employés pour couvrir la charge.

PCG : Problème de construction de grilles de travail. Il consiste à élaborer un emploi du temps grossier sur l'horizon de planification. Il y a deux sous-problèmes PPR et PGE.

PPR : Problème de placement de jours de repos. Il consiste à affecter les jours de travail et les jours de repos aux employés.

PGE : Problème de construction de grilles étiquetées. Il consiste à affecter les jours de repos et les étiquettes aux employés. Une étiquette est un ensemble de vacations ayant les mêmes heures de début et de fin.

PCC : Problème de construction de cycles. Il consiste à élaborer un emploi du temps détaillé sur tout l'horizon de planification.

$P23R$: Problème de planification de personnel avec deux ou trois jours de repos hebdomadaires par employé.

$P34R$: Problème de planification de personnel avec trois ou quatre jours de repos hebdomadaires par employé.

Introduction

La tomographie discrète constitue un thème émergeant en recherche opérationnelle. Ses applications sont nombreuses, notamment en médecine (imagerie médicale), en traitement d'images, en planification de personnel et en cristallographie. Cette discipline consiste à reconstruire un sous-ensemble de $\mathbb{Z}\star\mathbb{Z}$ ou plus généralement de $\mathbb{Z}^n$ à partir d'un ensemble de projections. Les sous-ensembles que l'on cherche à reconstruire peuvent correspondre à des images monochromatiques, des images en couleur, des emplois du temps, des cristaux ou bien des objets mathématiques tels que des pavages ou des packings. Les problèmes traités peuvent être formulés de la manière suivante : étant donnés $H = (h_1, \ldots, h_m)$ et $V = (v_1, \ldots, v_n)$ deux vecteurs à coordonnées entières positives, est-il possible de reconstruire un tableau $m \times n$ des éléments qui respecte les projections (H, V) ? Le plus souvent, les projections donnent le nombre d'éléments dans chacune des lignes et des colonnes. Selon la nature du problème, les éléments peuvent être des barres, des dominos, des couleurs ou bien tout simplement 0 et 1 comme dans le cas de la reconstruction d'une matrice binaire.

Cette thèse se situe dans ce cadre de reconstruction d'objets discrets. Elle est composée de quatre parties. La première partie traite essentiellement de la reconstruction des matrices binaires sous différentes contraintes (chapitres 1, 2 et 3). La deuxième partie (chapitre 4) aborde la reconstruction de tableaux colorés. Dans la troisième partie qui comprend les chapitres 5 et 6, on cherche à placer des barres et des dominos dans des tableaux. La quatrième partie (chapitre 7) a pour objectif de résoudre efficacement une classe des problèmes de planification de personnel à l'aide de modèles de tomographie discrète et de flot.

Le chapitre 1 dresse l'état de l'art concernant la tomographie discrète et introduit un certain nombre de notions fondamentales en reconstruction d'objets. On y trouvera en particulier les principaux résultats de reconstruction de matrices binaires constamment utilisées dans la suite de cette thèse. Ce tour d'horizon de différents problèmes de reconstruction vise à montrer dans quels cas les problèmes sont bien résolus en terme de taille des instances traitées et dans quels cas ils ne le sont pas.

Le chapitre 2 traite le problème de reconstruction de matrices binaires sous différentes contraintes "d'adjacence" que nous définissons au début du chapitre. Deux cellules dites "adjacentes" ne peuvent pas prendre simultanément la valeur 1. Contrairement à la reconstruction de matrices binaires, aucun auteur, à notre connaissance, ne s'est intéressé à ce problème qui a cependant des applications importantes dans plusieurs domaines tels que la physique statistique. Notre recherche vise à déterminer la complexité de ces problèmes.

Le chapitre 3 est consacré au problème de reconstruction de matrices binaires sous des contraintes de (p, q) périodicité alternée. Dans ce problème, les cellules (i, j) et $(i+p, j+q)$ ne peuvent pas prendre la même valeur. Ce travail est une extension des travaux de Del Lungo, Frosini, Nivat et Vuillon portant sur la reconstruction de matrices périodiques, dans lequel nous remplaçons la contrainte de périodicité par une contrainte de périodicité alternée.

Le chapitre 4 est voué à l'étude de la reconstruction de tableaux colorés, qui constitue une généralisation du problème classique de reconstruction de matrices binaires. Nous proposons deux formulations mathématiques pour le problème avec trois couleurs et nous étudions différents cas polynomiaux pour mieux appréhender sa difficulté. Nous concevons également une heuristique basée sur la programmation linéaire pour reconstruire des images avec niveaux de gris à partir de leurs projections.

Nous étudions au chapitre 5 les problèmes de pavage et "packing" de dominos colorés. Ces problèmes ont été traités dans la littérature sous l'hypothèse d'une unique projection ou de projections monotones. Nous considérons ici le cas général des projections orthogonales quelconques et nous supposons que les dominos sont horizontaux. Nous proposons des algorithmes polynomiaux pour résoudre certains problèmes.

Le chapitre 6 est réservé à un problème qui, à notre connaissance, n'a jamais été traité par la communauté scientifique. Il s'agit d'un problème de packing de barres horizontales dont une application est la planification de personnel. Le problème est de placer des barres dans un tableau en respectant les projections orthogonales. Nous introduisons des contraintes d'écart entre les barres et des contraintes sur les longueurs des barres. Nous montrons que lorsqu'une seule projection orthogonale est considérée, les problèmes de packing de barres sont le plus souvent polynomiaux.

L'application de la tomographie discrète dans le domaine de la planification de personnel fait l'objet du dernier chapitre. Il s'agit d'affecter des périodes de travail et des périodes de repos à des employés tout en respectant différentes contraintes. Les contraintes principales sont données par le nombre d'employés devant être présents pour chaque période considérée (charge) et le nombre de jours de repos dont doit bénéficier chaque employé au cours de l'horizon de planification. Nous nous intéressons exclusivement dans cette thèse à un groupe d'employés ayant les mêmes qualifications. Nous présentons tout d'abord un rapide état de l'art de ce très vaste domaine qui a suscité une abondante littérature. Dans la plupart des cas étudiés, les auteurs s'intéressent à élaborer des emplois du temps qui permettent de minimiser l'effectif du groupe d'employés pour un nombre de périodes donné.

Notre contribution consiste à établir un emploi du temps nominatif respectant les deux principales contraintes décrites précédemment pour un effectif fixé. Nous considérons différentes contraintes de placement des périodes de repos. La méthode de résolution proposée combine une approche algébrique et des techniques de flot.

Nous présentons à la fin de ce document un tableau résumant les principaux résultats de complexité démontrés dans cette thèse. Dans la conclusion, nous donnons également quelques perspectives de ce travail.

Chapitre 1

Introduction sur la tomographie discrète

Ce chapitre introductif expose le problème de reconstruction d'objets discrets et les motivations à la fois applicatives et théoriques qui ont poussé la communauté scientifique à se pencher sur ce problème. Nous rappelons les principales méthodes et techniques utilisées pour la reconstruction d'objets discrets. La majorité des résultats importants de complexité sont très récents. Nous en ferons une synthèse la plus complète possible.

1.1 Description

La tomographie discrète consiste à reconstruire des objets discrets à partir de leurs projections dans plusieurs directions. Les problèmes peuvent être classés suivant le nombre de directions et les propriétés géométriques des objets (convexité, connexité, périodicité, etc).

Tout au long de la thèse, nous adoptons les définitions et les notations suivantes :

Projections Orthogonales

Etant donnée une matrice binaire A $m \times n$, on définit :

$h_i = \sum_{j=1}^{n} A_{ij}$ la projection horizontale de la ligne i, $i = 1, \ldots, m$.

$H = (h_1, \ldots, h_m)$ la projection horizontale de A.

$v_j = \sum_{i=1}^{m} A_{ij}$ la projection verticale de la colonne j, $j = 1, \ldots, n$.

$V = (v_1, \ldots, v_n)$ la projection verticale de A.

Vecteurs particuliers

Soit T un vecteur de dimension m.

T est unitaire si $t_i = 1$, $i = 1, \ldots m$.

T est constant si $t_1 =, \ldots, = t_m$.

T est croissant si $t_1 \leq t_2 \leq, \ldots, \leq t_m$.

T est décroissant si $t_1 \geq t_2 \geq, \ldots, \geq t_m$.

T est monotone s'il est croissant ou décroissant.

Dominos

Un **domino** est un ensemble de deux cellules adjacentes.

Un domino **horizontal** a ses deux cellules sur la même ligne.

Un domino **vertical** a ses deux cellules sur la même colonne.

Un domino **unicoloré** a ses deux cellules soit noires (domino **noir**), soit blanches (domino **blanc**).

Un domino **bicoloré** a une cellule noire et une cellule blanche.

Un domino **coloré** est un domino unicoloré ou bicoloré.

Un domino horizontal (resp. vertical) **commence** à la colonne (resp. ligne) où se trouve sa cellule la plus à gauche (resp. en haut).

1.2 Reconstruction d'une matrice binaire

Le problème de reconstruction d'une matrice binaire à partir de ses projections orthogonales (H, V), noté $MB(H, V)$, consiste à trouver une matrice binaire de projection horizontale H et de projection verticale V. On note par $\mathcal{R}(H, V)$ la classe des matrices binaires $m \times n$ de projections orthogonales (H, V) (voir Figure 1.1). Lorsque $m = n$, le problème $MB(H, V)$ est équivalent au problème de reconstruction d'un graphe $G(S, E)$ de n sommets $S = (s_i,\ i = 1, \ldots, n)$ à partir des demi-degrés de ses sommets. La matrice d'adjacence de G appartient à $\mathcal{R}(H, V)$. Les demi-degrés extérieurs et intérieurs de chaque sommet s_i valent respectivement $d_i^+ = h_i$ et $d_i^- = v_i$, $i = 1, \ldots, n$. Le problème de décision correspondant, $\mathcal{E} - MB(H, V)$, est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ et $V = (v_1, \ldots, v_n)$ deux vecteurs à coordonnées entières positives.

Question : Existe-t-il une matrice binaire $m \times n$ respectant les projections (H, V) ?

Nous répondons à quatre questions relatives à ce problème : l'existence d'une solution, la reconstruction d'une solution, l'équivalence entre les solutions et l'unicité de la solution.

1.2.1 Existence d'une solution

Le théorème 1 [12] suivant est fondamental et donne, sous l'hypothèse de la décroissance de la projection verticale, des conditions nécessaires et suffisantes pour l'existence d'une matrice binaire à partir de ses projections orthogonales.

Théorème 1 (Ryser):
Si V est décroissant alors $\mathcal{E} - MB(H,V)$ a pour réponse 'oui' si et seulement si :
$\sum_{i=1}^{m} h_i = \sum_{j=1}^{n} v_j$ *et* $\sum_{j=1}^{k} v_j^* \geq \sum_{j=1}^{k} v_j,\ k = 1, \ldots, n-1.$
Avec $v_j^* = |\{i : h_i \geq j,\ i = 1, \ldots, m\}|,\ j = 1, \ldots, n.$
v_j^* *désigne le nombre de lignes de projections horizontales supérieures ou égales à* j.

Notons que ces conditions restent bien nécessaires lorsque la matrice à reconstruire satisfait d'autres contraintes. D'après le théorème 1, $\mathcal{E} - MB(H,V)$ est résolu par un algorithme linéaire. Si la réponse est 'oui', la reconstruction est obtenue par un algorithme polynomial.

1.2.2 Reconstruction d'une solution

Plusieurs algorithmes basés sur le théorème 1 permettent de reconstruire une matrice binaire [8]. Nous présentons un algorithme glouton en $O(mn + \max(m \log m, n \log n))$. A chaque étape (colonne) j, v_j $'1'$ sont placés à la colonne j et sur les lignes disponibles les plus prioritaires. Si le nombre de lignes disponibles est inférieur à v_j alors l'algorithme s'arrête et il n'y a pas de solution au problème. La ligne i est dite disponible à l'étape j lorsque $\alpha_i > 0$. α_i étant le nombre des $'1'$ non encore placés à l'étape j de l'algorithme. Les lignes les plus prioritaires sont celles qui ont les plus grandes valeurs α_i (voir Figure 1.1).

Algorithme A-MB(H,V)

1. $\alpha_i = h_i,\ i = 1, \ldots, m$;
2. pour $j = 1$ **à** n **faire**
  2.1 s'il n'existe pas v_j lignes disponibles alors fin ;
  2.2 placer v_j $'1'$ sur les lignes disponibles les plus prioritaires ;
  2.3 si un $'1'$ est placé sur la ligne i alors $\alpha_i \leftarrow \alpha_i - 1$;
fin faire ;

1.2.3 Equivalence entre les solutions

Définition 1:
Une ***bascule*** *est un ensemble de quatre cellules (i,j), $(i,j+k)$, $(i+h,j)$ et $(i+h,j+k)$ tel que les cellules (i,j) et $(i+h,j+k)$ sont de valeur 1 et les cellules $(i+h,j)$ et $(i,j+k)$ sont de valeur 0. L'opération de bascule consiste à échanger les 1 et 0 pour ces quatre cellules. Les projections orthogonales d'une matrice binaire demeurent inchangées après les opérations de bascules.*

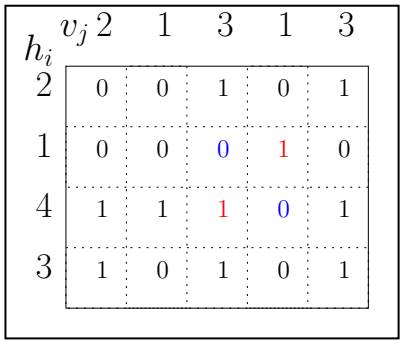


FIG. 1.1 – Reconstruction d'une matrice binaire par l'algorithme A-MB(H,V)

Wang et Zhang [13] ont calculé le nombre de solutions équivalentes. Par exemple, lorsque les projections orthogonales sont unitaires, il existe $n!$ matrices équivalentes. Ryser [12] a montré que les solutions sont équivalentes dans le sens où l'on peut passer d'une solution à une autre par une suite finie de bascules, comme le montre le théorème 2 :

Théorème 2:
Si $A, A' \in \mathcal{R}(H,V)$ alors A se transforme en A' à l'aide d'une suite finie d'opérations de bascules.

De ce théorème, on déduit que les éléments de la classe $\mathcal{R}(H,V)$ sont représentés par un graphe fortement connexe. Les sommets sont associés aux matrices et les arcs traduisent les opérations de bascules de passage d'une solution à une autre.

1.2.4 Unicité de la solution

Nous donnons les définitions relatives aux cellules de valeurs fixées dans toute solution.

Définition 2:
Une cellule (i,j) est 1-invariante si $A(i,j) = 1$ pour toute $A \in \mathcal{R}(H,V)$.
Une cellule (i,j) est 0-invariante si $A(i,j) = 0$ pour toute $A \in \mathcal{R}(H,V)$.
Une cellule est invariante si elle est 1-invariante ou 0-invariante.
Une matrice binaire est invariante (ou unique) si toutes ses cellules sont invariantes.

D'après le théorème 2, une matrice est invariante (solution unique) si et seulement si elle n'a pas de bascules. Par conséquent, tester l'uncité de la matrice, revient à tester si toutes les cellules sont invariantes. Haber [7] a donné les conditions nécessaires et suffisantes pour qu'une cellule soit invariante.

Théorème 3 (Haber):
Supposons que H et V sont monotones. Si la cellule (i,j) est 1-invariante dans $\mathcal{R}(H,V)$ alors il existe deux entiers e et f avec $i \leq e \leq m$ et $j \leq f \leq n$ tels que toute matrice

$A \in \mathcal{R}(H,V)$ a la forme suivante :

$$A = \begin{bmatrix} 1 & A1 \\ A2 & 0 \end{bmatrix} \quad (1)$$

Où :

1 est une matrice $e \times f$ où toutes ses entrées valent 1.

0 est la $(m-e) \times (m-f)$ matrice zéro.

$A1$ est une $e \times (m-f)$ matrice binaire.

$A2$ est une $(m-e) \times f$ matrice binaire.

Un résultat équivalent est déduit pour les cellules 0-invariantes. Pour trouver les cellules invariantes, il faut trier les projections orthogonales et mettre la matrice sous la forme (1). Si une telle forme existe, les cellules 1-invariantes sont les cellules (i,j) avec $1 \leq i \leq e$ et $1 \leq j \leq f$. Les cellules 0-invariantes sont les cellules (i,j) avec $e+1 \leq i \leq m$ et $f+1 \leq j \leq n$. Par conséquent, la détermination de l'ensemble des cellules invariantes est polynomiale et le résultat de l'uncité se déduit comme suit :

Proposition 1:
Le problème de l'unicité associé au problème $MB(H,V)$ est polynomial.

Nous concluons que les problèmes d'existence, d'unicité et de reconstruction d'une matrice binaire sont polynomiaux. Néanmoins, le nombre de solutions admissibles est exponentiel. Pour réduire la taille de l'espace de recherche de solutions, deux approches sont envisageables. La première consiste à augmenter le nombre des projections à satisfaire. Son inconvénient est que le problème de reconstruction est NP-complet pour un nombre de projections supérieur à deux [5]. La deuxième approche consiste à prendre en compte certaines propriétés connues a priori sur l'objet à reconstruire (convexité, connexité, symétrie, périodicité). Cette approche nous semble plus pratique que la première parce qu'on dispose souvent des informations sur la solution. De plus, une solution donnée par la première approche ne respecterait pas forcement ces propriétés malgré le nombre élévé des projections.

1.3 Reconstruction de matrices périodiques

Définition 3:
Soit A une matrice binaire $m \times n$. A est une matrice (p,q) périodique si

$A_{i,j} = A_{i+p,j+q}$ pour $1 \leq i \leq m-p$ et $1 \leq j \leq n-q$ (voir Figure 1.2).

Le problème de reconstruction de matrices binaires périodiques trouve son application dans la reconstruction de structures cristallines. Un cristal est la répétition d'un motif sur un réseau périodique. Le motif peut être simple, composé par un seul atome, comme dans le Fe, Al, Cu, etc, ou composé d'un grand nombre d'atomes comme dans les cristaux moléculaires organiques. Del Lungo et al. [9] ont montré le résultat suivant :

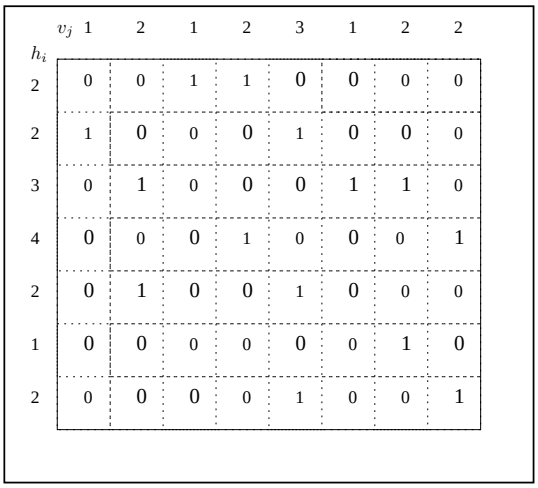


FIG. 1.2 – Matrice (2,3) périodique

Proposition 2:
Le problème de reconstruction d'une matrice $(1,1)$ *périodique est polynomial. De plus, si* m *et* n *sont premiers entre eux alors la solution, si elle existe, est unique.*

Ils ont proposé un algorithme de complexité quadratique. L'idée générale de l'algorithme se base sur la propriété suivante :

Propriété 1:
Toute matrice binaire A, $(1,1)$ *périodique, vérifie les propriétés suivantes :*
Si $h_i = h_{i+1} + 1$ *alors* $A_{i,n} = 1$ *et* $A_{i+1,1} = 0$ *pour* $i = 1, \ldots, m-1$.
Si $h_i = h_{i+1} - 1$ *alors* $A_{i+1,1} = 1$ *et* $A_{i,n} = 0$ *pour* $i = 1, \ldots, m-1$.
Si $v_j = v_{j+1} + 1$ *alors* $A_{m,j} = 1$ *et* $A_{1,j+1} = 0$ *pour* $j = 1, \ldots, n-1$.
Si $v_j = v_{j+1} - 1$ *alors* $A_{1,j+1} = 1$ *et* $A_{m,j} = 0$ *pour* $j = 1, \ldots, n-1$.

Ils ont procédé en deux phases. Dans une phase préliminaire, ils ont déduit de la propriété 1, les valeurs fixées sur les lignes 1 et m et les colonnes 1 et n. Ensuite, la valeur 1 de chaque cellule (i,j) se propage dans l'ensemble $\{(i+k, j+k) | 1 \leq i+k \leq m,\ 1 \leq j+k \leq n, k \in \mathbb{Z}\}$. A l'issue de cette phase, la matrice A est décomposée en deux matrices : la matrice F pour la partie fixe et la matrice A' de projections constantes pour la partie variable ($A = F + A'$). La deuxième phase consiste à reconstruire la matrice A' à l'aide de cycles. Un cycle commence à la cellule (i,j) avec $A'_{i,j} = 1$ et se propage dans A' par une périodicité $(1,1)$. Une solution est construite en $O(mn)$ parce qu'un cycle comporte exactement $PPCM(m,n)$ cellules étant donnée que les projections de A' sont constantes.

Les même auteurs [9] ont généralisé le résultat précédent au cas de la reconstruction d'une matrice $(1,p)$ périodique.

Proposition 3:
Soit p *un entier, le problème de reconstruction d'une matrice binaire* $(1,p)$ *périodique respectant les projections orthogonales est polynomial.*

Pour reconstruire une solution, ils déterminent la partie fixe de la matrice comme dans le cas précédent, puis, ils utilisent une réduction au problème 2-SAT pour déterminer la partie variable de la matrice. Le cas général de ce problème ((p,q) périodique) est non résolu.

1.4 Packing et pavage par des dominos

Dans cette section, nous supposons que l'objet discret à reconstruire est composé de dominos. Le problème de **pavage** par des dominos consiste à couvrir un tableau $m \times n$ par des dominos sans recouvrement en satisfaisant les projections (H, V). Le problème de **packing** de dominos consiste à placer sans recouvrement des dominos dans un tableau en respectant les projections (H, V). Les projections orthogonales sont le nombre de dominos rencontrés dans chacune des lignes et des colonnes.

Le problème de décision, $\mathcal{E}$–$PacDH(H, V)$, correspondant au problème $PacDH(H, V)$ de packing de dominos horizontaux est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ et $V = (v_1, \ldots, v_n)$ deux vecteurs à coordonnées entières positives.
Question : Existe-il un packing de dominos horizontaux respectant les projections (H, V) ?

Le problème $PacDH(H, V)$ est polynomial et l'algorithme A-MB(H,V) s'étend facilement pour le résoudre [11].

La complexité du problème de pavage par des dominos est inconnue. Cependant, lorsqu'au moins une projection est monotone, le problème est polynomial [11].

1.4.1 Pavage par des dominos avec une projection monotone

Supposons que H est monotone. Picouleau [11] a montré que m est pair et que les dominos verticaux commencent seulement sur les lignes impaires (voir Figure 1.3). Ce problème est équivalent au problème $PacDH(H', V)$ avec $h'_i = h_{2i},\ i = 1, \ldots \frac{m}{2}$. Soit S une solution au problème $PacDH(H', V)$. Une solution au problème de pavage est déduite de S de la manière suivante : d'une part, on associe à chaque domino commençant dans la cellule (i, j) dans S deux dominos horizontaux commençant dans les cellules $(2i - 1, j)$ et $(2i, j)$, et d'autre part, on associe à chaque cellule (i, j) non couverte dans S un domino vertical commençant dans la cellule $(2i - 1, j)$.

1.4.2 Pavage par des dominos avec deux projections monotones

Supposons que H et V sont monotones. Picouleau [11] a montré que m et n sont pairs et que le problème de pavage est équivalent au problème $MB(H', V')$ avec $h'_i = h_{2i},\ i = 1, \ldots, \frac{m}{2}$, et $v'_j = v_{2j},\ j = 1, \ldots, \frac{n}{2}$. Pour résoudre le problème de pavage, l'auteur a cherché, tout d'abord, une solution S au problème $MB(H', V')$. Ensuite, il a associé à chaque cellule (i, j) de valeur 1 dans S deux dominos verticaux commençant aux cellules

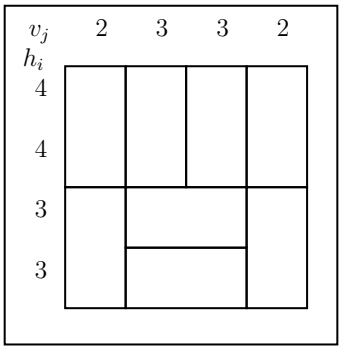


FIG. 1.3 – Pavage par des dominos avec une projection monotone

$(2i-1, 2j-1)$ et $(2i-1, 2j)$. Puis, il a associé à chaque cellule (i, j) de valeur 0 deux dominos horizontaux commençant aux cellules $(2i-1, 2j-1)$ et $(2i, 2j-1)$.

1.5 Pavage par des dominos colorés

On suppose que chaque cellule de valeur 1 est couplée avec une cellule adjacente de valeur 0. Cette paire de cellules sera modélisée par un **domino bicoloré**. Un domino bicoloré est constitué d'une cellule noire associée à la valeur 1 et d'une cellule blanche associée à la valeur 0. Dans ce cas, on parle du problème de pavage par des dominos bicolorés.

On pourrait également envisager que chaque cellule soit couplée avec une cellule adjacente de la même valeur. Cette pair de cellules sera représentée par un domino noir si les deux cellules sont de valeur 1 et par un domino blanc dans le cas contraire. Un tel pavage est appelé problème de pavage par des **dominos unicolorés**. Dans tous les problèmes de pavage par des dominos colorés, les projections donnent le nombre de cellules noires rencontrées dans chacune des lignes et des colonnes.

Le problème de pavage par des dominos colorés se rencontre dans la reconstruction des structures cristallines où un motif (élément) est constitué de deux atomes identiques (domino unicoloré) ou de deux atomes différents (domino bicoloré).

1.5.1 Pavage par des dominos bicolorés

Nous cherchons à paver un tableau $m \times n$ par des dominos bicolorés respectant les projections orthogonales (H, V). Picouleau [10] a résolu le problème relâché de la contrainte de la projection verticale (voir Figure 1.4). Il a retrouvé le théorème suivant :

Théorème 4:
Un tableau $m \times n$ est pavable par des dominos bicolorés respectant la projection horizontale H si et seulement si :
$\sum_{i=1}^{m} h_i = \frac{1}{2}mn$ *et* $max(2\sum_{j=1}^{i-1} h_j + h_i, \sum_{j=1}^{i-1}(n - h_j) + n - h_i) \leq in,\ i = 1, \ldots m.$

Dans le théorème, $\sum_{i=1}^{m} h_i = \frac{mn}{2}$ car le nombre de cellules blanches est égal au nombre

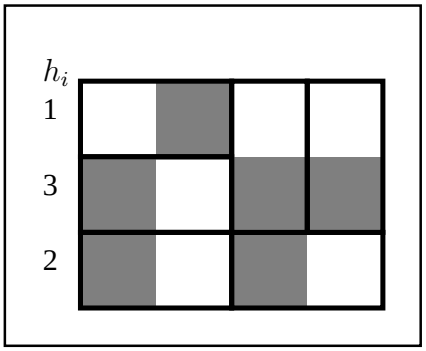


FIG. 1.4 – Pavage par des dominos bicolorés respectant la projection horizontale

de cellules noires et toutes les cellules sont recouvertes. Picouleau [10] propose également un algorithme en temps $O(mn)$ pour trouver une solution s'il en existe une.

1.5.2 Pavage par des dominos unicolorés

Nous cherchons à paver un tableau $m \times n$ par des dominos unicolorés respectant les projections orthogonales (H, V). Comme pour le problème de pavage par des dominos bicolorés, Picouleau [10] s'est intéressé à relâcher la contrainte de la projection verticale (voir Figure 1.5). Il a également montré le théorème suivant :

Théorème 5:
Un tableau $m \times n$ est pavable par des dominos unicolorés respectant la projection horizontale H si et seulement si les conditions i), ii), iii) et iv) sont vérifiées :

i) $h_i \leq n,\ i = 1, \ldots m$;

ii) $\sum_{i=1}^{m} = h_i$ *est pair ;*

iii) si n est pair : si $O_{2i-1} < j < O_{2i}$ *alors* $0 < h_j < n,\ i = 1, \ldots m,\ j = 1, \ldots n$;

iv) si n est impair alors m est pair et :

a) si O_{2i-1} *est pair alors* $h_{O_{2i-1}} < n,\ i = 1, \ldots m$;

b) si O_{2i} *est impair alors* $h_j > 0$ *pour* $j = 1, \ldots n$ *tel que* $O_{2i-1} < j < O_{2i}$.

O_i représente le numéro de la colonne ayant la $i^{ème}$ valeur impaire du vecteur H. Par exemple, pour $H = (1, 2, 4, 6, 8, 7, 8)$, nous avons $O_1 = 1$ et $O_2 = 6$.

Ce théorème montre que l'existence d'une solution est vérifiée en temps linéaire. Picouleau [10] propose aussi un algorithme en temps $O(mn)$ pour reconstruire une solution s'il en existe une.

On conclut que les problèmes de pavage par des dominos colorés sont polynomiaux lorsqu'une seule projection est prise en compte. Pour que le problème admette une solution respectant les deux projections orthogonales, il faut qu'il existe une solution respectant chaque projection et que le problème $MB(H, V)$ admette une solution. Bien évidemment, ces conditions nécessaires ne sont pas toujours suffisantes. Le tableau 1.1 synthétise la complexité des problèmes de pavage par des dominos.

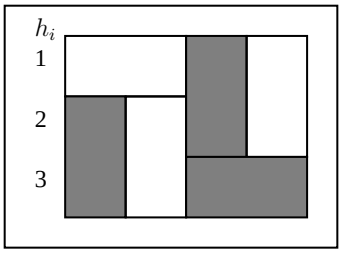


FIG. 1.5 – Pavage par des dominos unicolorés respectant la projection horizontale

Propriétés	**Arbitraire**	H **ou** V
Non coloré	?	?, P si H est monotone [11]
Unicoloré	?	P, [10]
Bicoloré	?	P, [10]

TAB. 1.1 – Complexité des problèmes de pavage par des dominos. P : polynomial

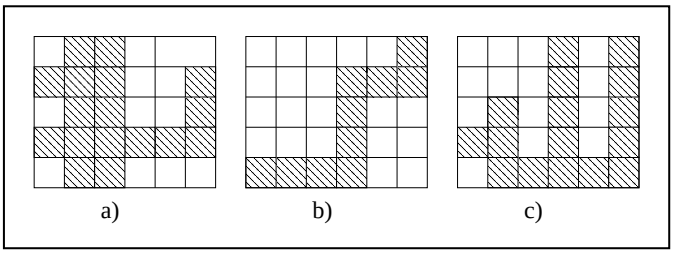


FIG. 1.6 – a) Un polyomino b) Un polyomino h-convexe c) Un polyomino v-convexe

1.6 Reconstruction de polyominos

Dans certaines applications, telle que l'imagerie médicale, on exige des propriétés de connexité selon différentes directions afin d'assurer la continuité de la matière suivant ces directions.

Définition 4:
Soit S une matrice binaire, on définit les propriétés suivantes de S (voir Figure 1.6) :

- **p :** *S est un polyomino si les cellules de valeur* 1 *constituent une partie connexe, c'est-à-dire, toute pair de cellules de valeur* 1 *est liée par un chemin composé de cellules de valeur* 1.
- **h :** *S est h-convexe si les cellules de valeur* 1 *de chaque ligne constituent un ensemble connexe, c'est-à-dire, les cellules de valeur* 1 *ne sont pas séparées par des cellules de valeur* 0.
- **v :** *S est v-convexe si les cellules de valeur* 1 *de chaque colonne constituent un ensemble connexe.*

Barcucci et al. [1] ont montré que le problème de reconstruction d'une matrice binaire devient NP-complet dès que l'on exige les propriétés **(h,v)** ou **p**. Woeginger [14] a montré

que le problème reste toujours NP-complet même si les propriétés **(p, v)**, **(p, h)**, **h** ou **v** sont considérées. Toutefois, le problème redevient polynomial [1] si l'on exige les trois propriétés **(p, h, v)** à la fois.

L'algorithme proposé par Barcucci et al. [1] repose sur trois phases. La première phase consiste à choisir une configuration initiale sur les bords du tableau, c'est-à-dire, sur les lignes 1 et m et les colonnes 1 et n. Le nombre de ces configurations est borné par m^2n^2 car la solution cherchée satisfait les propriétés **(p, h, v)**. Ensuite, pour chaque configuration de départ, une procédure de propagation de contraintes permet de fixer les valeurs de certaines cellules pour respecter les propriétés **(p, h, v)**. A l'issue de cette phase, on ne peut pas conclure l'existence d'une solution car il reste souvent quelques cellules indéterminées. Les valeurs de ces cellules peuvent s'exprimer par des clauses booléennes à deux variables. Enfin, l'éventuel problème 2-SAT est résolu pour trouver une solution s'il en existe une. L'algorithme global est de complexité $O(m^4n^4)$.

Le tableau 1.2 donne une synthèse sur la complexité des problèmes de reconstruction de polyominos :

Propriétés	**Arbitraire**	**Polyomino**
Arbitraire	P, [12]	NP, [14]
h-convex	NP, [1]	NP, [1]
v-convex	NP, [1]	NP, [1]
hv-convex	NP, [14]	P, [1], [2]

TAB. 1.2 – Complexité des problèmes de reconstruction de polyominos

1.7 Reconstruction de tableaux colorés

Le problème de reconstruction de matrices binaires peut être vu comme un problème de reconstruction d'un tableau monocoloré où toutes les cellules de valeur 1 sont colorées en noir et toutes les cellules de valeur 0 sont colorées en blanc.

Le problème de reconstruction d'un tableau k-coloré, TkC, est une généralisation du problème $MB(H, V)$. Chaque cellule est colorée par une couleur parmi un ensemble de k couleurs. Les projections donnent le nombre de cellules de chaque couleur dans chacune des lignes et des colonnes.

Les problèmes TkC trouvent leur application dans des domaines distincts, tels que la cristallographie, l'ordonnancement et la planification de personnel. En effet, une cellule colorée peut représenter un atome, une tâche sur une machine ou une activité.

Notons d la dimension du tableau. Pour les problèmes que nous aborderons dans le chapitre 4, $d = 2$. Notons ξ le nombre d'axes non parallèles de projections. $\xi = 2$ si l'on considère seulement les projections orthogonales.

Gardner et al. [5] ont montré que le problème de reconstruction d'une matrice binaire est NP-complet pour $D \geq 2$ et $\xi \geq 3$. Ce résultat de complexité s'étend aussi au problème TkC car le problème $MB(H, V)$ en est un cas particulier lorsque $k - 2$ couleurs ont des

projections nulles. Gardner et al. [6] ont montré également que le problème TkC est NP-complet pour $k \geq 7$ et $d \geq 2$. Chrobak et Dürr [3] ont démontré que le problème TkC reste NP-complet pour $d = 2$, $\xi = 2$ et $k \geq 4$. Ainsi, le seul cas de complexité inconnue est celui de la reconstruction d'un tableau 3-coloré. Ce problème sera étudié dans le chapitre 4.

Costa et al. [4] ont montré à l'aide de modèles d'emploi du temps que si toutes les couleurs à l'exception de la couleur de fond ont des projections orthogonales unitaires alors le problème TkC est NP-complet pour $n \geq 3$. Si l'on impose de plus que chaque couleur, autre que la couleur de fond, apparaît au plus deux fois dans le tableau alors le problème TkC devient polynomial.

1.8 Conclusion

Ce tour d'horizon nous montre bien qu'il y a peu de résultats dans le domaine de la tomographie discrète. Le résultat principal concerne la résolution du problème $MB(H, V)$. Nous verrons dans les chapitres suivants que $MB(H, V)$ est omniprésent dans tous les problèmes de reconstruction. Nous nous intéresserons particulièrement d'une part, à reconstruire des matrices binaires sous différentes contraintes telles que l'adjacence et la périodicité, et d'autre part, à paver des tableaux par des barres.

Bibliographie

[1] E. Barcucci and A. Del Lungo. Reconstructing convex polyominoes from their horizontal and vertical projections. *Theoretical computer science*, 155(1) :321–347, 1996.

[2] M. Chrobak and C. Dürr. Reconstructing hv-convex polyominoes from orthogonal projections. *Information Processing Letters*, 69 :283–289, 1999.

[3] M. Chrobak and C. Dürr. Reconstructing polyatomic structures from x-rays : Np completness proof for three atoms. *Theoretical computer science*, 259(1) :81–98, 2001.

[4] M.C. Costa, D. de Werra, and C. Picouleau. On some special cases of an image reconstruction. Technical report, Cedric-Cnam, 2002.

[5] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computionnal complexity of reconstructing lattice sets from their x-rays. *Discrete Mathematics*, 202 :45–71, 1999.

[6] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computational complexity of determining polyatomic structures by x-rays. *Theoretical computer science*, 233 :91–106, 2000.

[7] R. M. Haber. Term rank of 0,1 matrices. *Rend. Sem. Mat. Univ. padova*, 30 :24–51, 1960.

[8] A. Kuba and G.T. Hermann. Discrete tomography : a historical overview. In *Discrete Tomography : Foundations, Algorithms and Applications*, pages 3–33. Birkhauser, 1999.

[9] A. Del Lungo, A. Frosini, M. Nivat, and L. Vuillon. Reconstruction under periodicity constraints. *ICALP*, pages 38–56, 2002.

[10] C. Picouleau. Reconstruction of a coloured domino tiling from its projections. Technical report, Cedric-Cnam, 2001.

[11] C. Picouleau. Reconstruction of domino tiling from its two orthogonal projections. *Theoretical computer science*, 255(1) :437–447, 2001.

[12] H.J. Ryser. Combinatorial properties of matrices of zeros and ones. *Canad. J. Math*, 9 :371–377, 1957.

[13] B. Wang and F. Zhang. On the precise number of (0,1)-matrices in u(r,s). *Discrete Mathematics*, 187 :211–220, 1998.

[14] G.J. Woeginger. The reconstruction of polyominoes from their orthogonal projections. *Information Processing Letters*, 77(5-6) :225–229, 2001.

Chapitre 2

Reconstruction de matrices binaires sous contraintes d'adjacence

Dans ce chapitre, nous considérons le problème de reconstruction de matrices binaires sous contraintes d'adjacence. Les matrices à reconstruire doivent respecter les projections orthogonales et deux cellules adjacentes ne peuvent pas prendre simultanément la valeur 1. Selon la définition d'adjacence (voisinage), nous distinguons trois types de contraintes : 4-adjacence, 6-adjacence et 8-adjacence. Nous décrivons le problème et les motivations

qui nous ont encouragé à le traiter. Puis, nous considérons séparément les trois types de contraintes d'adjacence et nous étudions la complexité des problèmes associés.

2.1 Description du problème

Nous rappelons que le problème polynomial *MB(H,V)* consiste à reconstruire une matrice binaire dont H est la projection horizontale et V est la projection verticale (voir [2] ou la section 1.2). Dans ce chapitre, nous imposons des contraintes supplémentaires d'adjacence. Autrement dit, si une cellule est de valeur 1 alors toutes ses voisines sont de valeur 0. Comme dans tout problème de reconstruction, on connaît les projections orthogonales et on cherche à trouver une matrice binaire respectant les projections orthogonales et les contraintes d'adjacence.

2.1.1 Applications

Le problème de reconstruction de matrices binaires avec contraintes d'adjacence est inspiré des modèles des gaz durs (Hard Gas) utilisés en mécanique statistique. Cette discipline consiste à déterminer les propriétés microscopiques de la matière condensée (énergie, densité, entropie, etc). Dans les modèles des gaz durs, deux sites (positions) voisins, ne sont pas occupés simultanément par des particules parce que l'énergie de répulsion entre elles est assez forte. On parlera du modèle des gaz durs carrés (Hard Square Gas) lorsque les particules sont disposées sur un réseau bidimensionnel carré. Toutes les propriétés statistiques des gaz sont déduites de la fonction de répartition qui est elle même déduite de l'énumération de toutes les configurations possibles des particules. Ceci impose d'énumérer les matrices binaires n'ayant pas deux cellules voisines de valeur 1.

2.1.2 Adjacence

Dans un tableau ou une matrice, chaque cellule a deux types de voisines, à savoir, ses 4 voisines selon les axes (lignes ou colonnes) et ses 4 voisines selon les diagonales. Le plus souvent, la relation de proximité entre les voisines axiales est différente de celle entre les voisines diagonales. Par conséquent, nous définissons les relations suivantes d'adjacence entre les cellules d'un tableau (voir Figure 2.1) : deux cellules (i, j) et (i', j') sont dites :

- 2-adjacentes horizontalement si elles sont voisines suivant une ligne,
- 2-adjacentes verticalement si elles sont voisines suivant une colonne,
- 4-adjacentes si elles sont voisines suivant un axe (ligne ou colonne),
- 6-adjacentes si elles sont voisines suivant un axe (ligne ou colonne) ou suivant la première diagonale,
- 8-adjacentes si elles sont voisines suivant un axe ou une diagonale.

En résumé, les cellules adjacentes à une cellule (i, j) sont :

- $(i, j+1)$ et $(i, j-1)$ pour la 2-adjacence horizontale,
- $(i+1, j)$ et $(i-1, j)$ pour la 2-adjacence verticale,
- $(i, j+1), (i, j-1), (i+1, j)$ et $(i-1, j)$ pour la 4-adjacence,

- $(i, j+1), (i, j-1), (i+1, j), (i-1, j), (i+1, j+1)$ et $(i-1, j-1)$ pour la 6-adjacence,
- $(i, j+1), (i, j-1), (i+1, j-1), (i+1, j), (i+1, j+1), (i-1, j), (i-1, j-1)$ et $(i-1, j+1)$ pour la 8-adjacence.

On vérifie facilement que le problème *MB(H,V)* reste polynomial si l'on ajoute la contrainte de **2-adjacence horizontale**. En effet, il s'agit d'un problème de packing de barres horizontales de longueur unitaire avec un écart minimal unitaire (voir section 6.2.1). Nous retrouvons le même résultat de complexité pour le problème *MB(H,V)* avec la contrainte de **2-adjacence verticale**.

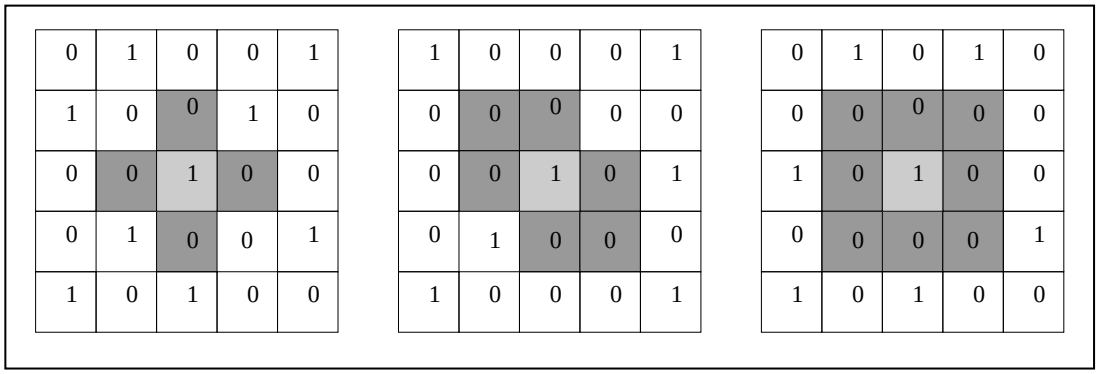


FIG. 2.1 – Contraintes d'adjacence : de gauche à droite, 4-adjacence, 6-adjacence et 8-adjacence

Dans la section suivante, nous imposerons à la fois les contraintes de 2-adjacence horizontale et 2-adjacence verticale, c'est-à-dire, la contrainte de **4-adjacence**.

2.2 Matrice satisfaisant la projection horizontale et la contrainte de 4-adjacence (*M4adj(H)*)

Nous supposons ici que deux cellules 4-adjacentes ne prennent pas simultanément la valeur 1 et nous abandonnons la contrainte de la projection verticale. Le problème de décision correspondant, $\mathcal{E}-M4adj(H)$, est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ un vecteur à coordonnées entières positives et un entier n.
Question : Existe-t-il une matrice binaire $m \times n$ respectant la projection horizontale H et la contrainte de 4-adjacence ?

Nous introduisons les notions de damier et d'escalier qui sont de grande importance pour résoudre le problème *M4adj(H)*.

Définition 5:
On appelle ***damier*** *D tout tableau binaire $m \times n$ dont les $'0'$ et les $'1'$ sont alternés. On note $D(i, j)$ la valeur de la cellule (i, j) dans le damier (voir Figure 2.2).*

Lorsque n est pair, un damier admet $\frac{n}{2}$ $'1'$ par ligne. Si n est impair, un damier admet $\frac{n+1}{2}$ ou $\frac{n-1}{2}$ $'1'$ par ligne. Un damier est alors la matrice binaire la plus dense qui respecte la contrainte de 4-adjacence.

1	0	1	0	1
0	1	0	1	0
1	0	1	0	1
0	1	0	1	0
1	0	1	0	1

FIG. 2.2 – Damier avec $D(1,1) = 1$

Définition 6:
Soit g une fonction entière croissante définie de l'ensemble $E \subset \mathbb{N}$ vers $\mathbb{N}$. On appelle ***escalier*** *de g sur l'intervalle $[a,b] \subset E$ et on le note $E_g(a,b)$, l'ensemble des cellules :*
$\{(i,j),\ i = a,\dots,b,\ j = g(i-1), g(i-1)+1,\dots,g(i)-1, g(i)\}$.

La figure 2.3 représente l'escalier $E_g(2,4)$ de la fonction g définie sur l'ensemble $\{1,\dots,5\}$ avec $g(1) = 1$, $g(2) = 3$, $g(3) = 5$, $g(4) = 6$, $g(5) = 7$.

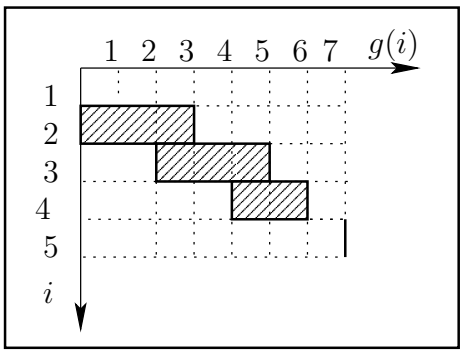


FIG. 2.3 – Escalier $E_g(2,4)$ hachuré

Le résultat suivant donne des conditions nécessaires d'existence de solution pour *M4adj(H)* :

Proposition 4:
Pour que $\mathcal{E}-M4adj(H)$ ait pour réponse 'oui' il faut que $2h_i - 1 \leq n,\ i = 1,\dots,m$.

Preuve:
Cette condition est nécessaire car sur chaque ligne i, il y a h_i cellules de valeur 1 et au moins $h_i - 1$ cellules de valeur 0 pour respecter la contrainte de 4-adjacence, soit au moins $2h_i - 1$ cellules. ■

Nous remarquons que **cette condition est suffisante lorsque n est pair**. Une solution est reconstruite en considérant un damier D et en mettant à zéro $(\frac{n}{2} - h_i)$ cellules quelconques de valeur 1 dans chaque ligne i. Nous obtenons $\frac{n}{2} - (\frac{n}{2} - h_i) = h_i$ cellules de valeur 1. Dans la suite nous supposons que la condition est satisfaite et nous nous intéressons au cas n impair.

Avant de traiter le cas n impair, il est indispensable d'introduire les notations et les définitions qui suivent (voir Figure 2.4). Deux lignes i_1 et i_2 sont dites **adjacentes** si

$|i_1 - i_2| = 1$. Une ligne i est dite **saturée** si $h_i = \frac{n+1}{2}$. Dans une ligne saturée, les $'1'$ occupent toutes les colonnes impaires. On en déduit que s'il existe deux lignes saturées adjacentes alors le problème *M4adj(H)* n'admet pas de solution. Deux lignes saturées i_1 et i_2 sont dites **consécutives** lorsqu'elles ne sont pas séparées par une autre ligne saturée. Toute ligne située entre deux lignes saturées consécutives est dite **intermédiaire**. Notons par $P(i_1, i_2)$ le sous-problème du *M4adj(H)* correspondant aux lignes $i_1, \ldots, i_2$. Une solution à $P(i_1, i_2)$ respecte les projections horizontales des lignes intermédiaires et la contrainte de 4-adjacence. $k = i_2 - i_1 - 1$ est le nombre de lignes intermédiaires situées entre les lignes saturées i_1 et i_2.

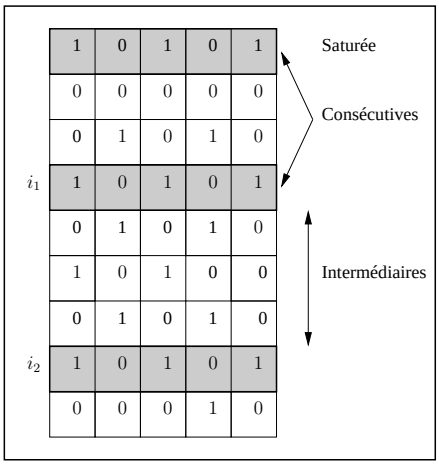


FIG. 2.4 – Lignes saturées, consécutives, intermédiaires

Deux cas se présentent selon la parité de k.

Proposition 5:
Si les lignes i_1 et i_2 sont saturées consécutives séparées par un nombre impair de lignes alors $P(i_1, i_2)$ admet une solution si et seulement si $2h_i - 1 \leq n,\ i = i_1, \ldots, i_2$.

Preuve:
D'après la proposition 4, $2h_i - 1 \leq n,\ i_1 = 1, \ldots, i_2$ est une condition nécessaire. Elle est aussi suffisante : une solution est reconstruite en plaçant un damier D sur les lignes $i_1, \ldots, i_2$ avec $D(i_1, 1) = D(i_2, 1) = 1$ et en gardant h_i $'1'$ dans la ligne i et en mettant les autres à zéro. ■

Lorsque k est pair, il est impossible de placer un damier avec $D(i_1, 1) = D(i_2, 1) = 1$. Soit $f1$ une fonction entière définie sur l'ensemble $\{i = i_1, \ldots, i_2\}$ vers $\{1, \ldots, n\}$ qui à i associe $f1(i)$ avec

$$\begin{cases} f1(i_1) = 1 \\ f1(i_1 + 1) = 1 + 2(\frac{n-1}{2} - h_{i_1+1}) \\ f1(i) = min(n, f1(i-1) + 1 + 2(\frac{n-1}{2} - h_i)) \quad i = i_1 + 2, \ldots, i_2 \end{cases}$$

$f1$ est bien définie et croissante car $h_i \leq \frac{n-1}{2}$ pour les lignes intermédiaires. Soit i_0 tel que $f1(i_0) = n$ et $f1(i) < n$ pour $i < i_0$. L'escalier $E_{f1}(i_1 + 1, i_0)$ couvre alors chaque ligne

i de la colonne $f1(i-1)$ à la colonne $f1(i)$ (voir Figure 2.5). Nous proposons l'algorithme polynomial $A-P(i_1,i_2)$ pour reconstruire une solution. Le principe de cet algorithme consiste à scinder l'ensemble des cellules en deux demi-damiers séparés par un ensemble de cellules de valeur 0, dit **escalier**.

Algorithme $A-P(i_1,i_2)$

1. Calculer la fonction $f1$ et l'escalier $E_{f1}(i_1+1,i_0)$;
2. les cellules situées sur $E_{f1}(i_1+1,i_0)$ prennent la valeur 0 ;
3. les cellules situées au-dessus de $E_{f1}(i_1+1,i_0)$ prennent la valeur 1 si elles correspondent à des lignes et des colonnes de même parité ;
4. les cellules situées au-dessous de $E_{f1}(i_1+1,i_0)$ prennent la valeur 1 si elles correspondent à des lignes et des colonnes de parité différente ;
5. pour $i_0 \leq i \leq i_2$, garder h_i $'1'$ sur la ligne i et mettre les autres à 0.

Proposition 6:
Si les lignes i_1 et i_2 sont saturées consécutives séparées par un nombre k pair de lignes alors le problème $P(i_1,i_2)$ admet une solution si et seulement si :

i) $2h_i - 1 \leq n,\ i = i_1, \ldots, i_2$;
ii) $\sum_{i=i_1+1}^{i_2-1} h_i \leq (k-1)\frac{n+1}{2} - \frac{k}{2}$.

Preuve:
Sans perte de généralité, nous supposons que i_1 est impair. D'après la proposition 4, la condition i) est nécessaire. Montrons que la condition ii) l'est aussi. Notons qu'il y a $(\frac{n+1}{2}-1)$ colonnes paires et $\frac{n+1}{2}$ colonnes impaires car n est impair. Sur les lignes intermédiaires, il y a au plus $(\frac{k}{2}-1)$ $'1'$ dans les colonnes impaires et il y en a au plus $\frac{k}{2}$ dans les colonnes paires. Il en découle que le nombre maximal des $'1'$ dans les lignes intermédiaires est $(\frac{n+1}{2}-1)\frac{k}{2} + \frac{n+1}{2}(\frac{k}{2}-1) = \frac{n+1}{2}(k-1) - \frac{k}{2}$. Pour qu'il existe une solution, il faut que ce nombre maximal soit supérieur au nombre des $'1'$ à placer sur ces lignes. D'où, $\sum_{i=i_1+1}^{i_2-1} h_i \leq (k-1)\frac{n+1}{2} - \frac{k}{2}$.

Inversement, montrons que si ces conditions sont satisfaites alors $A-P(i_1,i_2)$ fournit une solution à $P(i_1,i_2)$.

Nous constatons que d'après $f1$, $f1(i_0) = n$, $f1(i)$ et i sont de parité différente pour $i_1 < i < i_0$ et $f1(i) < n$ pour $i < i_0$.

- *L'algorithme $A-P(i_1,i_2)$ place $(\frac{n+1}{2})$ $'1'$ sur la ligne i_1 puisqu'elle est impaire et située au-dessus de l'escalier.*
- *Montrons par l'absurde que $i_0 < i_2$. Si $i_0 \geq i_2$ alors $f1(i) = f1(i-1)+1+2(\frac{n-1}{2}-h_i)$ pour $i = i_1+2$ à i_2-1 et $n > f1(i_2-1) = k+2\sum_{i=i_1+1}^{i_2-1}(\frac{n-1}{2}-h_i)$, ce qui est absurde car d'après ii), $k+2\sum_{i=i_1+1}^{i_2-1}(\frac{n-1}{2}-h_i) \geq n+1$. L'algorithme place alors $(\frac{n+1}{2})$ $'1'$ sur la ligne i_2 parce qu'elle est paire et située au-dessous de l'escalier $(i_0 < i_2)$.*
- *Sur chaque ligne i impaire avec $i_1 < i < i_0$, l'algorithme place $(\frac{f1(i-1)-1}{2})$ $'1'$ dans les colonnes paires situées à gauche de la colonne $f1(i-1)$. Il place également $(\frac{n-f1(i)+1}{2})$ $'1'$ dans les colonnes impaires situées à droite de la colonne $f1(i)$, soit au total $(\frac{n-1-2\frac{n-1}{2}+2h_i}{2} = h_i)$ $'1'$ placés sur la ligne i.*

- *Avec le même raisonnement, on démontre que l'algorithme* $A-P(i_1, i_2)$ *respecte les projections horizontales des lignes paires situées avant la ligne* i_0.
- *Supposons que* i_0 *est impair. A l'étape 4, l'algorithme place* $(\frac{f1(i_0-1)-1}{2})$ *'1' sur les colonnes paires situées à gauche de la colonne* $f1(i_0-1)$. *On a* $2h_{i_0} \leq f1(i_0-1)$ *car d'après* $f1$, $f1(i_0) = n \leq f1(i_0-1)+1+2(\frac{n-1}{2}-h_{i_0})$. *D'où,* $h_{i_0} \leq \frac{f1(i_0-1)-1}{2}$ *car* $f1(i_0-1)$ *est impair et* $2h_{i_0}$ *est pair. Par conséquent, à l'étape 5, l'algorithme garde* h_{i_0} *'1' et met* $(\frac{f1(i_0-1)-1}{2} - h_{i_0})$ *cellules à 0 sur la ligne* i_0. *On retrouve le même résultat lorsque* i_0 *est pair.*
- *L'algorithme respecte également les projections horizontales des lignes situées entre* i_0 *et* $i_2 - 1$ *car elles ne sont pas saturées.*

La contrainte de 4-adjacence est aussi respectée car les cellules de l'escalier valent 0. ■

L'utilisation des escaliers pour la résolution du problème $P(i_1, i_2)$ est illustrée par la figure 2.5 : à gauche $\frac{n-1}{2} > h_i$ pour certaines lignes intermédiaires, à droite $\frac{n-1}{2} = h_i$ pour toutes les lignes intermédiaires.

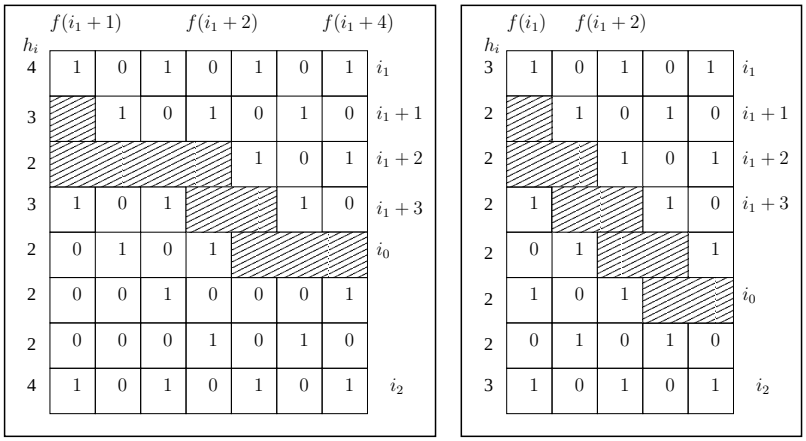


FIG. 2.5 – Algorithme $A - P(i_1, i_2)$: $E_{f1}(i_1+1, i_0)$ hachuré

La proposition suivante récapitule tous les cas traités précédemment et donne les conditions nécessaires et suffisantes d'existence d'une solution pour le problème *M4adj(H)* :

Proposition 7:

$\mathcal{E}-M4adj(H)$ *a pour réponse 'oui' si et seulement si :*

i) $2h_i - 1 \leq n,\ i = 1, \ldots, m$;

ii) pour tout couple de lignes i_1 *et* i_2 *saturées consécutives séparées par un nombre* k *pair de lignes :* $\sum_{i=i_1+1}^{i_2-1} h_i \leq (k-1)\frac{n+1}{2} - \frac{k}{2}$.

Preuve:

Notons par $i_1, \ldots, i_k$ *l'ensemble des indices des lignes saturées consécutives. D'après les propositions 4 et 6, les conditions i) et ii) sont nécessaires. Si elles sont satisfaites, une solution est reconstruite par l'algorithme A-M4adj(H)) :*

Algorithme A-M4adj(H)

1. Placer un damier sur les lignes $1, \ldots, i_1$ *avec* $D(i_1, 1) = 1$ *;*
2. placer un damier sur les lignes $i_k, \ldots, m$ *avec* $D(i_k, 1) = 1$ *;*
3. pour tout couple de lignes i *et* i' *saturées consécutives séparées par un nombre impair de lignes, placer un damier sur les lignes* $i, \ldots, i'$ *avec* $D(i, 1) = D(i', 1) = 1$ *;*
4. pour toute ligne i *d'un damier, garder* h_i *'1' et mettre les autres cellules à 0 ;*
5. pour tout couple de lignes i *et* i' *saturées consécutives séparées par un nombre pair de lignes, résoudre* $P(i, i')$ *par* $A-P(i_1, i_2)$ *;*

■

Nous démontrons de la même façon que le problème de reconstruction d'une matrice binaire respectant la projection verticale et la contrainte de 4-adjacence est polynomial.

2.3 Matrice respectant les projections orthogonales et la contrainte de 4-adjacence (*M4adj(H, V)*)

Nous nous proposons maintenant d'aller au-delà de la relaxation d'une projection présentée ci-dessus et de considérer les deux projections orthogonales. Le problème de décision correspondant, $\mathcal{E}-M4adj(H, V)$, est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ et $V = (v_1, \ldots, v_n)$ deux vecteurs à coordonnées entières positives.
Question : Existe-t-il une matrice binaire $m \times n$ respectant les projections (H, V) et la contrainte de 4-adjacence ?

La complexité de *M4adj(H,V)* est à notre connaissance inconnue. Cependant, nous avons dégagé des conditions nécessaires et des conditions suffisantes d'existence de solution, comme le montre le résultat suivant :

Proposition 8:
Soient $H^p = (h_2, h_4, \ldots, h_{2\lfloor \frac{m}{2} \rfloor})$, $H^i = (h_1, h_3, \ldots, h_{2\lceil \frac{m}{2} \rceil - 1})$, $V^p = (v_2, v_4, \ldots, v_{2\lfloor \frac{n}{2} \rfloor})$ *et* $V^i = (v_1, v_3, \ldots, v_{2\lceil \frac{n}{2} \rceil - 1})$. *Chacune de deux conditions suivantes est suffisante pour que M4adj(H,V) admette une solution :*

i) les problèmes $MB(H^p, V^p)$ *et* $MB(H^i, V^i)$ *admettent des solutions,*

ii) les problèmes $MB(H^p, V^i)$ *et* $MB(H^i, V^p)$ *admettent des solutions.*

Preuve:
Montrons que la condition i) est suffisante. Soient S^p *et* S^i *deux solutions respectivement aux problèmes* $MB(H^p, V^p)$ *et* $MB(H^i, V^i)$. *Si la condition i) est satisfaite alors une solution,* S, *au problème M4adj(H,V) est construite de la manière suivante :*

$$S(i,j) = \begin{cases} S^p(\frac{i}{2}, \frac{j}{2}) & si \quad i, j \ pairs \\ S^i(\frac{i+1}{2}, \frac{j+1}{2}) & si \quad i, j \ impairs \\ 0 & sinon \end{cases}$$

$S(i,j)$ est la valeur de la cellule (i,j) dans S. S respecte toutes les projections car S^p satisfait les projections des lignes et colonnes paires et S^i satisfait les projections des lignes et colonnes impaires. Elle respecte aussi la contrainte de 4-adjacence car si $S(i,j) = 1$ alors i et j sont de même parité et les quatre cellules $(i,j+1), (i,j-1), (i+1,j)$ et $(i-1,j)$ sont de valeur 0. On démontre de même que la deuxième condition est suffisante. ■

Néanmoins, nous remarquons que ces conditions ne sont pas nécessaires (voir Figure 2.6). Prenons $H = (3,1,3)$ et $V = (2,1,1,1,1,1)$, *M4adj(H,V)* admet une solution alors que le problème $MB(H^i, V^i)$ où $H^i = (3,3)$ et $V^i = (2,1,1)$ n'admet pas de solution (les sommes des projections orthogonales ne sont pas égales).

v_j / h_i	2	1	1	1	1	1
3	1	0	0	1	0	1
1	0	1	0	0	0	0
3	1	0	1	0	1	0

FIG. 2.6 – Contre exemple

Concernant, les conditions nécessaires, on peut énoncer le résultat suivant :

Proposition 9:
Pour que $\mathcal{E} - M4adj(H,V)$ ait pour réponse 'oui', il faut que les problèmes MB(H,V), M4adj(H), M4adj(V), MB(H,V) avec contrainte de 2-adjacence horizontale et MB(H,V) avec contrainte de 2-adjacence verticale admettent des solutions.

La preuve est évidente parce que toute solution au problème *M4adj(H,V)* est une solution à chacun des autres problèmes.

Devant la difficulté de résoudre le problème *M4adj(H,V)* pour m et n quelconques, nous traiterons deux sous-problèmes avec deux lignes ($m = 2$) et trois lignes ($m = 3$).

2.4 Matrice $2 \times n$ respectant les projections orthogonales et la contrainte de 4-adjacence (*M2.4adj(H,V)*)

Nous supposons ici que $m = 2$. Les colonnes se décomposent en un ensemble de p blocs $B^1 B^2 \dots B^i \dots B^p$ où B^i est un ensemble maximal de colonnes consécutives de projection unitaire (voir Figure 2.7). Toute colonne située entre deux blocs consécutifs est de projection nulle. La longueur l^i du bloc B^i est le nombre de ses colonnes. Un bloc est dit pair ou impair selon que sa longueur est paire ou impaire.

Dans tous les blocs, les $'1'$ et les $'0'$ sont alternés sur les deux lignes pour respecter la contrainte de 4-adjacence. Si B^i est pair, $\lfloor \frac{l^i}{2} \rfloor$ $'1'$ sont placés sur chacune des lignes. Sinon, $(\lfloor \frac{l^i}{2} \rfloor + 1)$ $'1'$ sont placés sur une ligne et $\lfloor \frac{l^i}{2} \rfloor$ $'1'$ sont placés sur l'autre ligne. Donc chaque

ligne contient au moins $\lfloor \frac{l^i}{2} \rfloor$ '1' et au plus $\lceil \frac{l^i}{2} \rceil$ '1' sur le bloc B^i.

Une solution à *M2.4adj(H,V)* est donnée par l'algorithme A-M2.4adj(H,V).

Algorithme A-M2.4adj(H,V)

1. Sur chaque bloc B^i pair, placer $\lfloor \frac{l^i}{2} \rfloor$ '1' sur chaque ligne ;
2. sélectionner $h_1 - \sum_{i=1}^{p} \lfloor \frac{l^i}{2} \rfloor$ blocs impairs. Sur chaque bloc B^i sélectionné, placer $(\lfloor \frac{l^i}{2} \rfloor + 1)$ '1' sur la première ligne et $\lfloor \frac{l^i}{2} \rfloor$ '1' sur la deuxième ligne ;
3. sur les autres blocs impairs, placer $(\lfloor \frac{l^i}{2} \rfloor + 1)$ '1' sur la deuxième ligne et $\lfloor \frac{l^i}{2} \rfloor$ '1' sur la première ligne.

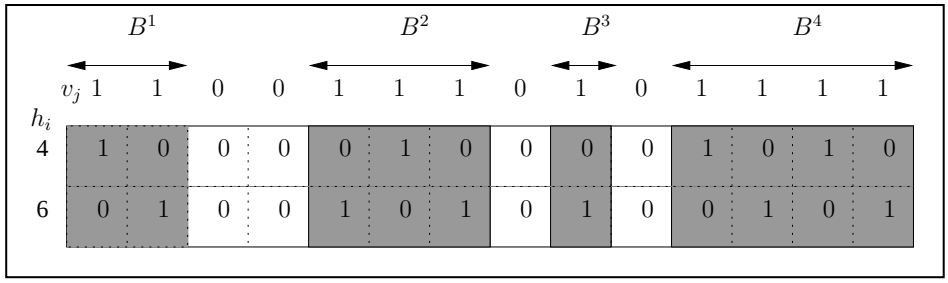


FIG. 2.7 – Un exemple de décomposition en blocs du problème *M2.4adj(H,V)*

Proposition 10:
M2.4adj(H,V) est polynomial et l'algorithme A-M2.4adj(H,V) en donne une solution s'il en existe une.

Preuve:
D'après la figure 2.7, la contrainte de 4-adjacence est respectée. Montrons que les projections horizontales de deux lignes le sont aussi. Notons par $\mathcal{E}$ *un ensemble arbitraire de* $h_1 - \sum_{i=1}^{p} \lfloor \frac{l^i}{2} \rfloor$ *blocs impairs et par* $\overline{\mathcal{E}}$ *son complémentaire dans les blocs impairs.*

L'égalité entre les projections verticales et horizontales implique que :

$h_1 + h_2 = \sum_{i=1}^{p} l^i = \sum_{i,B^i \ pair} 2\lfloor \frac{l^i}{2} \rfloor + \sum_{i,B^i \ impair} (2\lfloor \frac{l^i}{2} \rfloor + 1) = 2\sum_{i=1}^{p} \lfloor \frac{l^i}{2} \rfloor + \sum_{i,B^i \ impair} 1.$

D'où $|\mathcal{E}| + |\overline{\mathcal{E}}| = h_1 + h_2 - 2\sum_{i=1}^{p} \lfloor \frac{l^i}{2} \rfloor$ *= le nombre de blocs impairs. On en déduit que* $|\overline{\mathcal{E}}| = h_2 - \sum_{i=1}^{p} \lfloor \frac{l^i}{2} \rfloor$ *car* $|\mathcal{E}| = h_1 - \sum_{i=1}^{p} \lfloor \frac{l^i}{2} \rfloor$.

Sur la première ligne, l'algorithme place $\sum_{i,B^i \ pair} \lfloor \frac{l^i}{2} \rfloor$ '1' *sur les blocs pairs,* $\sum_{i \in \mathcal{E}} (\lfloor \frac{l^i}{2} \rfloor + 1)$ '1' *sur l'ensemble* $\mathcal{E}$ *et* $\sum_{i \in \overline{\mathcal{E}}} \lfloor \frac{l^i}{2} \rfloor$ '1' *sur l'ensemble* $\overline{\mathcal{E}}$*, soit au total* $\sum_{i=1}^{p} \lfloor \frac{l^i}{2} \rfloor + |\mathcal{E}| = h_1$ *cellules de valeur* 1.

Sur la deuxième ligne, l'algorithme place $\sum_{i,B^i \ pair} \lfloor \frac{l^i}{2} \rfloor$ '1' *sur les blocs pairs,* $\sum_{i \in \mathcal{E}} \lfloor \frac{l^i}{2} \rfloor$ '1' *sur l'ensemble* $\mathcal{E}$ *et* $\sum_{i \in \overline{\mathcal{E}}} (\lfloor \frac{l^i}{2} \rfloor + 1)$ '1' *sur l'ensemble* $\overline{\mathcal{E}}$*, soit au total* $\sum_{i=1}^{p} \lfloor \frac{l^i}{2} \rfloor + |\overline{\mathcal{E}}| = h_2$ *cellules de valeur* 1. ■

2.5 Matrice $3 \times n$ respectant les projections orthogonales et la contrainte de 4-adjacence (*M3.4adj(H,V)*)

Nous nous plaçons ici dans le cas $m = 3$. Avant d'étudier le cas général avec des projections verticales quelconques (0, 1 ou 2) nous avons besoin de traiter trois problèmes particulier P1, P2 et P3.

2.5.1 P1 : Matrice $3 \times k$ respectant les projections orthogonales (H, V), $V = 1$ et la contrainte de 4-adjacence

Nous supposons que $V = (1, \ldots, 1)$ et $H = (h_1, h_2, h_3)$. Pour que $P1$ admette une solution il faut que $h_1 + h_2 + h_3 = k$ et $h_i \leq \lceil \frac{k}{2} \rceil,\ i = 1, 2, 3$. La première condition traduit l'égalité entre la somme des projections horizontales et la somme des projections verticales. La deuxième condition est nécessaire car chaque ligne contient au plus $\lceil \frac{k}{2} \rceil$ $'1'$ pour respecter la contrainte de 4-adjacence. Supposons que ces conditions sont satisfaites et construisons une solution. *Si l'une des projections horizontales est nulle alors l'algorithme A-M2.4adj(H,V) trouve une solution.* Sinon, deux cas se présentent :

cas 1) $h_1 = \lfloor \frac{k}{2} \rfloor + 1$ **ou** $h_3 = \lfloor \frac{k}{2} \rfloor + 1$. Supposons que $h_1 = \lfloor \frac{k}{2} \rfloor + 1$.

Dans ce cas k est impair car d'après la condition ii) $h_1 \leq \lceil \frac{k}{2} \rceil$. h_1 est alors égal au nombre de colonnes impaires et $h_2 + h_3 = \lfloor \frac{k}{2} \rfloor$ est égal au nombre de colonnes paires. Une solution est reconstruite comme suit :

- sur la première ligne, placer h_1 $'1'$ sur les colonnes impaires ;
- sur la deuxième ligne, placer h_2 $'1'$ sur les h_2 premières colonnes paires ;
- sur la troisième ligne, placer h_3 $'1'$ sur les h_3 dernières colonnes paires.

cas 2) si $h_1 \leq \lfloor \frac{k}{2} \rfloor$ **et** $h_3 \leq \lfloor \frac{k}{2} \rfloor$. Supposons que $h_3 \leq h_1$.

Une solution est donnée par l'algorithme A-P1(H,V) ci-dessous (voir Figure 2.8).

Algorithme A-P1(H,V)

1. Si $h_2 = 1$
   1.1 sur la première ligne, placer h_1 $'1'$ sur les colonnes paires ;
   1.2 affecter 1 à la cellule $(2, 1)$;
   1.3 sur la troisième ligne, placer h_3 $'1'$ sur les colonnes impaires appartenant à $[3, k]$;
2. Si $h_2 > 1$
   2.1 sur la ligne 1, placer des $'1'$ sur les colonnes paires appartenant aux intervalles $[2, 2h_1 + 2h_2 - 2\lfloor \frac{k}{2} \rfloor - 2]$ et $[2h_2, 2\lfloor \frac{k}{2} \rfloor]$;
   2.2 sur la ligne 2, placer des $'1'$ sur les colonnes impaires appartenant à $[1, 2h_2 - 1]$;
   2.3 sur la ligne 3, placer des $'1'$ sur les colonnes paires appartenant à $[2h_1 + 2h_2 - 2\lfloor \frac{k}{2} \rfloor, 2h_2 - 2]$ et sur les colonnes impaires dans $[2h_2 + 1, 2\lceil \frac{k}{2} \rceil - 1]$.

Proposition 11:
Si $h_1 \leq \lfloor \frac{k}{2} \rfloor$, $h_2 \leq \lceil \frac{k}{2} \rceil$, $h_3 \leq \lfloor \frac{k}{2} \rfloor$, $h_1 + h_2 + h_3 = k$ *et* $h_3 \leq h_1$ *alors l'algorithme A-P1(H,V) donne une solution à P1.*

Preuve:
Nous rappelons que les trois projections sont non nulles. Autrement dit l'algorithme A-M2.4adj(H,V) trouve une solution.

Lorsque $h_2 = 1$, *il est évident que les projections orthogonales et la contrainte d'adjacence sont respectées car* $h_1 = \lceil \frac{k-1}{2} \rceil$ *et* $h_3 = \lfloor \frac{k-1}{2} \rfloor$. *En effet,* $h_1 + h_2 = k - 1$, $h_1 \leq \lfloor \frac{k}{2} \rfloor$, $h_3 \leq \lfloor \frac{k}{2} \rfloor$ *et* $h_3 \leq h_1$ *impliquent que* $\frac{k-1}{2} \leq h_1 \leq \lfloor \frac{k}{2} \rfloor$. *D'où,* $h_1 = \lceil \frac{k-1}{2} \rceil$ *et* $h_3 = \lfloor \frac{k-1}{2} \rfloor$ *car* $h_1 + h_2 = k - 1$.

Supposons que $h_2 > 1$. *Montrons tout d'abord que* $2 \leq 2h_1 + 2h_2 - 2\lfloor \frac{k}{2} \rfloor - 2 \leq 2h_2$.

Par hypothèse, $h_1 + h_2 + h_3 = k = \lfloor \frac{k}{2} \rfloor + \lceil \frac{k}{2} \rceil \Leftrightarrow h_1 + h_2 - \lfloor \frac{k}{2} \rfloor - 1 = \lceil \frac{k}{2} \rceil - 1 - h_3$.

Par conséquent, si $\lceil \frac{k}{2} \rceil - 1 \leq h_3 \leq \lfloor \frac{k}{2} \rfloor$ *alors* $h_3 = \lfloor \frac{k}{2} \rfloor = \lceil \frac{k}{2} \rceil - 1 = h_1$ ($h_3 \leq h_1$) *et* $h_2 = 1$, *d'où une contradiction. On a également* $h_1 + h_2 - \lfloor \frac{k}{2} \rfloor - 1 < h_2$ *car* $h_1 < \lfloor \frac{k}{2} \rfloor + 1$.

D'après la figure 2.8, l'algorithme A-P1(H,V) respecte les projections verticales et la contrainte de 4-adjacence sur les colonnes. Montrons que l'algorithme respecte aussi les projections horizontales et la contrainte de 4-adjacence sur les lignes.

Concernant la première ligne, la projection horizontale est satisfaite car les deux intervalles $[2, 2h_1 + 2h_2 - 2\lfloor \frac{k}{2} \rfloor - 2]$ *et* $[2h_2, 2\lfloor \frac{k}{2} \rfloor]$ *contiennent* h_1 *colonnes paires :*

$(h_1 + h_2 - \lfloor \frac{k}{2} \rfloor - 1) + (\lfloor \frac{k}{2} \rfloor - h_2 + 1) = h_1$. *La contrainte de 4-adjacence est respectée parce que les cellules* $(1, 2h_1 + 2h_2 - 2\lfloor \frac{k}{2} \rfloor - 2)$ *et* $(1, 2h_2)$ *sont séparées par au moins une colonne :* $2h_2 - 2h_1 - 2h_2 + 2\lfloor \frac{k}{2} \rfloor + 2 = 2 + 2(\lfloor \frac{k}{2} \rfloor - h_1) \geq 2$.

Concernant la troisième ligne, la projection horizontale est satisfaite car, d'une part $(\lfloor \frac{k}{2} \rfloor - h_1)$ *'1' sont placés sur les colonnes paires, et d' autre part,* $(\lceil \frac{k}{2} \rceil - h_2)$ *'1' sont placés sur les colonnes impaires, soit au total* $(\lfloor \frac{k}{2} \rfloor - h_1 + \lceil \frac{k}{2} \rceil - h_2) = (k - h_1 - h_2 = h_3)$ *'1'. La contrainte de 4-adjacence est également respectée parce que les cellules* $(3, 2h_2 - 2)$ *et* $(3, 2h_2 + 1)$ *sont séparées par deux colonnes. Il est immédiat de vérifier que la projection horizontale et la contrainte de 4-adjacence sont respectées sur la deuxième ligne.* ■

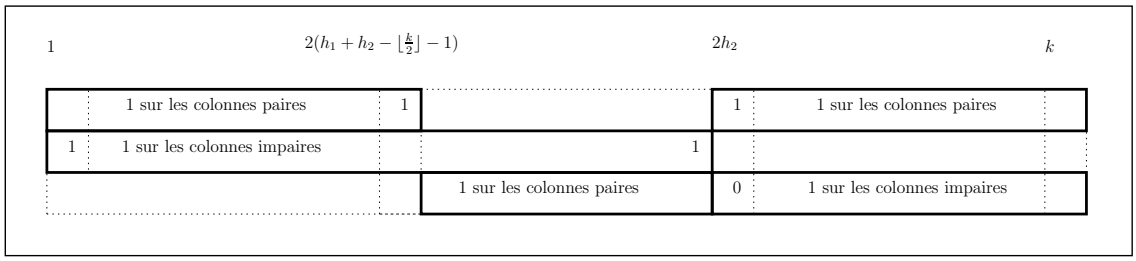


FIG. 2.8 – Schéma de l'algorithme A-P1(H,V)

On en déduit le résultat suivant :

Proposition 12:
P1 admet une solution si et seulement si :

i) $h_1 + h_2 + h_3 = k$;

ii) $h_i \leq \lceil \frac{k}{2} \rceil$, $i = 1, 2, 3$.

2.5.2 P2 : Matrice A $3 \times k$ respectant les projections orthogonales (H, V), $V = 1$ et la contrainte de 4-adjacence avec $A(2,1) = 1$

Nous supposons ici que $V = (1, \ldots, 1)$ et que $A(2,1) = 1$. De façon symétrique, on pourrait considérer $A(2,k) = 1$ au lieu de $A(2,1) = 1$.

Proposition 13:
P2 admet une solution si et seulement si :
i) $h_1 + h_2 + h_3 = k$, $h_2 \leq \lceil \frac{k}{2} \rceil$;
ii) $h_i \leq \lfloor \frac{k}{2} \rfloor$, $i = 1, 3$.

Preuve:
$h_1 + h_2 + h_3 = k$ *et* $h_2 \leq \lceil \frac{k}{2} \rceil$ *sont évidents (voir P1).* $h_i \leq \lfloor \frac{k}{2} \rfloor$, $i = 1, 3$ *car* $A(1,1) = A(1,3) = 0$. *Si ces conditions sont vérifiées alors on est bien dans le cas 2 du sous-problème P1 et l'algorithme A-P1(H,V) trouve une solution.* ■

2.5.3 P3 : Matrice A $3 \times k$ respectant les projections orthogonales $H, V = 1$ et la contrainte de 4-adjacence avec $A(2,1) = A(2,k) = 1$

Nous supposons que $V = (1, \ldots, 1)$ et $A(2,1) = A(2,k) = 1$.

Proposition 14:
P3 admet une solution si et seulement si :
i) $h_2 \leq \lceil \frac{k}{2} \rceil$;
ii) $h_i \leq \lfloor \frac{k-1}{2} \rfloor$, $i = 1, 3$.

Preuve:
i) est évidente (voir P2 ou P1).
Notons P^{k-1} *la restriction de P3 sur les* $(k-1)$ *premières colonnes.* P^{k-1} *est un problème P2 et respecte la projection horizontale* $(h_1, h_2 - 1, h_3)$. *D'après la proposition 13,* P^{k-1} *admet une solution si et seulement si* $h_1 + h_2 - 1 + h_3 = k - 1$, $h_2 - 1 \leq \lceil \frac{k-1}{2} \rceil$ *et* $h_i \leq \lfloor \frac{k-1}{2} \rfloor$ *pour i=1,3. D'où ii) est aussi nécessaire.*

Réciproquement, si ces conditions sont satisfaites, deux cas se présentent selon h_2 :

1) Si $h_2 = \lfloor \frac{k}{2} \rfloor + 1$ *alors P3 admet une solution et l'algorithme A-P1(H,V) trouve une solution (voir cas 2 de P1). Dans ce cas on a forcément k impair.*

2) Si $h_2 \leq \lfloor \frac{k}{2} \rfloor$ *alors considérons* S^{k-1} *la solution donnée par l'algorithme A-P1(H,V) au problème* P^{k-1}. *Une solution S à P3 est formée par* S^{k-1}, *la cellule* $(2,k)$ *de valeur 1 et les cellules* $(1,k)$ *et* $(3,k)$ *de valeur 0. Pour respecter la contrainte de 4-adjacence entre les colonnes k et* $k-1$, *il faut montrer que dans* S^{k-1}, *la cellule* $(2, k-1)$ *est de valeur 0 (voir Figure 2.9). D'après l'algorithme A-P1(H,V), la cellule* $(2, k-1)$ *de* S^{k-1} *est de valeur 1 si et seulement si* $(k-1)$ *est impair et* $h_2 - 1 = \lceil \frac{k-1}{2} \rceil$. *Donc k est pair et* $h_2 = \frac{k}{2} + 1$, *d'où une contradiction car* $h_2 \leq \frac{k}{2}$. ■

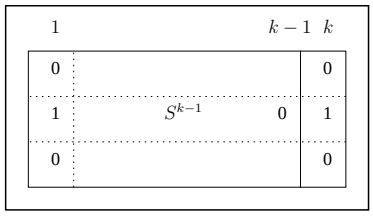


FIG. 2.9 – S et S^{k-1}

2.5.4 Problème général : *M3.4adj(H,V)*

La séquence $(v_1, \ldots, v_n)$ des projections verticales se décompose en une suite de blocs $B_1^0 B_1^1 B_1^2 B_2^0 B_2^1 B_2^2 \ldots B_i^0 B_i^1 B_i^2 \ldots B_p^0 B_p^1 B_p^2$ où B_i^0 (resp. B_i^1 et B_i^2) est une séquence maximale de colonnes consécutives de projection 0 (resp. 1 et 2). Désignons par $l_i^j,\ j = 0, 1, 2,\ i = 1, \ldots, p$ la longueur du bloc B_i^j. Nous avons $0 \leq l_i^j \leq n,\ j = 0, 1, 2$ et $l_i^2 \leq 1$ pour $i = 1, \ldots, p$, car si $v_j = 2$ alors les $'1'$ sont placés sur la première et la troisième lignes. Ce qui est impossible sur deux colonnes adjacentes : s'il existe i tel que $l_i^2 \geq 2$ alors le problème n'a pas de solution.

Il faut mettre des $'0'$ sur les colonnes de projection nulle. Deux blocs sont voisins ou adjacents s'ils admettent deux colonnes adjacentes. Pour chaque bloc B_i^1 de projection unitaire, trois cas (classes) se présentent selon les blocs adjacents (voir Figure 2.10).

$C1$: B_i^1 n'est adjacent à aucun bloc de projection 2 : $l_i^2 = 0$ et si $i \neq 1$ alors $l_i^0 \neq 0$.

$C2$: B_i^1 est adjacent à un seul bloc de projection 2 : soit ($i > 1$, $l_i^0 = 0$ et $l_i^2 = 0$), soit ($i > 1$, $l_i^0 \neq 0$ et $l_i^2 \neq 0$), soit ($i = 1$ et $l_i^2 \neq 0$).

$C3$: B_i^1 est adjacent à deux blocs de projection 2 : $i > 1$, $l_i^0 = 0$ et $l_i^2 \neq 0$.

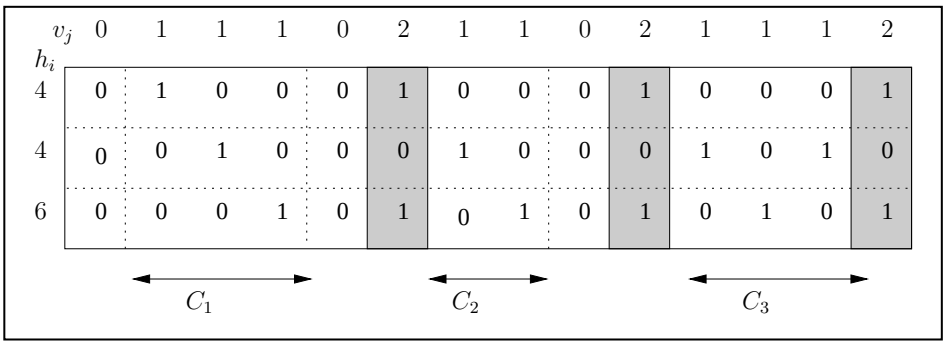


FIG. 2.10 – Classes des blocs B_i^1

Nous avons dégagé des conditions nécessaires et suffisantes d'existence de solution pour chaque problème $Pi,\ i = 1, 2, 3$ définis en 2.5.1, 2.5.2 et 2.5.3. Le problème Pi modéliserait le problème de reconstruction d'un bloc de type Ci si l'on connaissait les projections horizontales restreintes à ce bloc. En effet, d'une part, la cellule $(2, 1)$ est de valeur 1 dans un bloc de type $C2$ car il est adjacent à un seul bloc de projection 2, et d'autre part, les cellules $(2, 1)$ et $(2, k)$ sont de valeur 1 dans un bloc de type $C3$ de longueur k car il est adjacent à deux blocs de projection 2. Notre problème est donc de

déterminer ces projections horizontales restreintes. Mais ces projections sont liées : pour chaque ligne, leur somme est égale à sa projection horizontale dans *M3.4adj(H,V)*.

Nous proposons une méthode polynomiale en deux étapes pour résoudre *M3.4adj(H,V)*. La première étape consiste à calculer les projections horizontales des blocs, c'est-à-dire, la projection horizontale de chaque ligne par bloc. Ceci permet de décomposer le problème initial en sous-problèmes Pi. La deuxième étape consiste à résoudre les problèmes Pi dont les projections verticales sont unitaires et les projections horizontales sont données par la première étape. Chaque bloc B_j^2 de longueur non nulle a obligatoirement un $'1'$ sur la première et la troisième ligne.

La première étape consiste à calculer le nombre des $'1'$ sur chaque ligne et sur chaque bloc B_j^1 de longueur non nulle à travers la recherche d'un flot maximal dans un graphe biparti augmenté $G(\mathcal{X}, \mathcal{Y}, \mathcal{E})$ (voir Figure 2.11). $\mathcal{X}$= $(ligne\,1, ligne\,2, ligne\,3)$ représente les trois lignes. $\mathcal{Y}$ représente les blocs B_j^1 de longueur non nulle. Sur la figure 2.11, de façon à illustrer tous les cas, le bloc B_1^1 est de type $C1$, le bloc B_3^1 est de type $C2$ et le bloc B_5^1 est de type $C3$. Ajoutons deux sommets S et T au graphe G. S est connecté à la deuxième ligne par un arc de capacité h_2 et à la première (resp. troisième) ligne par un arc de capacité $h_1 - d$ (resp. $h_3 - d$) où d est le nombre de blocs B_j^2 de longueurs non nulles. Chaque bloc B_j^1 est relié au sommet T par un arc de capacité égale à sa longueur. L'idée est d'associer la valeur du flot sur l'arc (ligne i, B_j^1) au nombre de $'1'$ sur la ligne i et sur le bloc B_j^1. Les capacités des arcs de la forme (ligne i, B_j^1) sont déduites des conditions nécessaires et suffisantes trouvées précédemment pour l'existence d'une solution pour les sous-problèmes P1, P2 et P3. Nous rappelons que ces conditions portent seulement sur les projections horizontales.

La projection horizontale de la ligne i est satisfaite si et seulement si le flot maximal dans G sature l'arc (S, ligne i). D'après les sous-sections précedentes, les problèmes de reconstructions restreints aux blocs B_j^1 admettent une solution si et seulement si le flot maximal sature les arcs (B_j^1, T) et les capcités des arcs (ligne i, B_j^1) sont respectées. D'où, la première étape admet une solution si et seulement si le flot maximal sature les arcs d'entrée et de sortie ; sa valeur est égale à $\sum_{i=1}^{p} l_i^1 = h_1 + h_2 + h_3 - 2d$.

La deuxième étape consiste à reconstruire une solution sur chaque bloc B_j^1, c'est-à-dire, résoudre les sous-problèmes $P1$, $P2$ et $P3$. Donc si la première étape admet une solution alors la deuxième étape admet aussi une solution car les capacités des arcs (ligne i, B_j^1) traduisent les conditions nécessaires et suffisantes d'existence de solution pour $P1$, $P2$ et $P3$.

On déduit de ce qui précède que le problème *M3.4adj(H,V)* admet une solution si et seulement si la valeur du flot maximal dans G est $\sum_{i=1}^{p} l_i^1 = h_1 + h_2 + h_3 - 2d$. L'algorithme A-M3.4adj(H,V) résout le problème.

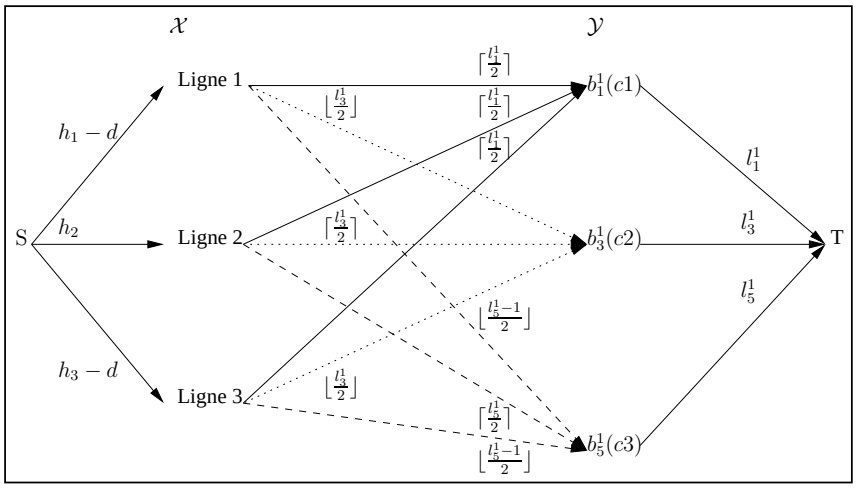


FIG. 2.11 – Reconstruction d'une matrice binaire $3 \times n$ avec 4-adjacence

Algorithme A-M3.4adj(H,V)

1. Décomposer les colonnes en blocs ;
2. pour les blocs B_j^2, mettre un $'1'$ sur les lignes 1 et 2 ;
3. chercher un flot maximal dans le graphe $G(\mathcal{X}, \mathcal{Y}, \mathcal{E})$;
4. si le flot maximal ne sature pas tous les arcs d'entrée et de sortie alors il n'existe pas de solution ;
5. pour chaque bloc B_j^1, résoudre le sous-problème associé où les projections horizontales sont les valeurs du flot maximal sur les arcs (ligne i, B_j^1).

On en déduit la proposition suivante :

Proposition 15:
Le problème M3.4adj(H,V) est polynomial et l'algorithme A-M3.4adj(H,V) en trouve une solution s'il en existe une.

La figure 2.12 représente un problème *M3.4adj(H,V)* et le flot maximal associé. On en déduit que les vecteurs $(1,1,1)$, $(0,1,1)$ et $(0,2,1)$ sont respectivement les projections horizontales des blocs de type $C1$, $C2$ et $C3$.

2.5.5 Conclusion

En conclusion, nous avons montré que le poblème de reconstruction d'une matrice binaire sous la contrainte de 4-adjacence est polynomial lorsque le nombre de lignes ou de colonnes est inférieur à 3. Pour $m > 3$, le problème de reconstruction d'une matrice binaire $m \times n$ sous contrainte de 4-adjacence est de complexité inconnue. Même pour $m = 4$, la technique de décomposition en trois sous-problèmes liés par un modèle de flot ne peut pas s'appliquer. En effet, si la projection verticale d'une colonne vaut 2 alors les deux $'1'$ sont soit sur la première et la troisième lignes, soit sur la deuxième et la quatrième lignes. De plus, les blocs de projection 2 ne sont plus de longueur inférieure ou égale à 1.

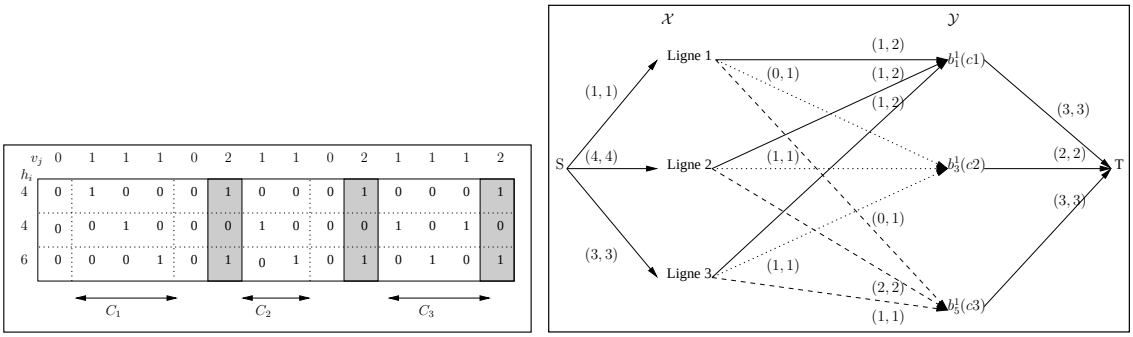
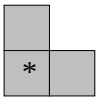


FIG. 2.12 – Exemple de flot maximal pour *M3.4adj(H,V)*

2.6 Matrice respectant les projections orthogonales et la contrainte de 6-adjacence (*M6adj(H,V)*)

Nous supposons ici que deux cellules 6-adjacentes ne prennent pas simultanément la valeur 1 (voir Figure 2.1). Nous montrons la NP-complétude de ce problème à partir de la réduction d'un problème de packing de L-paveurs (voir chapitre 1 pour les problèmes de packing).

On appelle **L-paveur** la forme géométrique suivante : 

Le symbole '*' indique le centre du paveur. Le problème de packing de L-paveurs consiste à placer sans recouvrement des L-paveurs dans un tableau en respectant les projections (H, V). h_i donne le nombre de centres des L-paveurs dans la ligne i. v_j donne le nombre de centres des L-paveurs dans la colonne j.

Proposition 16:
M6adj(H,V) est NP-complet.

Preuve:
Soient Q une instance du problème de packing de L-paveurs respectant les projections (H, V) et P une instance du problème M6adj(H,V) respectant également (H, V). Montrons que Q et P sont équivalentes. Supposons que Q admet une solution $S1$. On obtient une solution pour P en remplaçant les centres des L-paveurs dans $S1$ par des $'1'$ et les autres cellules par des $'0'$. De même, supposons que P admet une solution $S2$. On obtient une solution pour Q en remplaçant toute cellule (i, j) de valeur 1 et les cellules adjacentes $(i, j+1)$ et $(i-1, j)$ par un L-paveur. La contrainte de 6-adjacence dans P assure que les L-paveurs ne se recouvrent pas (voir Figure 2.13). Le problème M6adj(H,V) est ainsi NP-complet car le problème de packing de L-paveurs est NP-complet [1]. ■

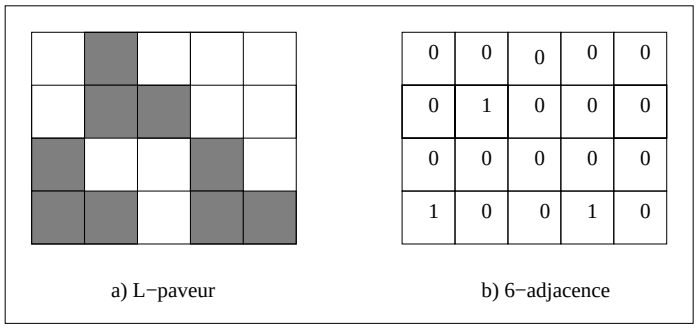
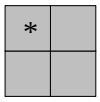


FIG. 2.13 – Equivalence entre 6-adjacence et L-paveurs

2.7 Matrice respectant les projections orthogonales et la contrainte de 8-adjacence (*M8adj(H,V)*)

Nous considérons ici la contrainte de 8-adjacence, c'est-à-dire, deux cellules 8-adjacentes n'ont pas simultanément la valeur 1 (voir Figure 2.1). Nous montrons que ce problème est au moins aussi difficile que le problème de reconstruction de tableaux 3-colorés en passant par le problème de packing de carrés 2×2.

On appelle paveur **carré** 2×2 la forme géométrique suivante :  Le symbole '*' indique le centre du carré.

Le problème de packing de carrés 2×2 consiste à placer sans recouvrement des carrés 2×2 dans un tableau en satisfaisant les projections (H, V). h_i est le nombre des centres des carrés dans la ligne i. v_j est le nombre des centres des carrés dans la colonne j.

Proposition 17:
M8adj(H,V) est au moins aussi difficile que le problème de reconstruction d'un tableau 3-coloré.

Preuve:
Soient Q une instance du problème de packing de carrés 2×2 d'un tableau respectant les projections (H, V) et P une instance du problème M8adj(H,V) respectant également les projections (H, V). Montrons que Q et P sont équivalentes. Supposons que Q admet une solution $S1$. Une solution pour P est obtenue en remplaçant les centres des carrés 2×2 dans $S1$ par des $'1'$ et les autres cellules par des $'0'$. De même, supposons que P admet une solution $S2$. Si l'on remplace chaque cellule (i, j) de valeur 1 et les cellules adjacentes $(i, j+1)$,$(i+1, j)$ et $(i+1, j+1)$ par un carré 2×2, on obtient une solution pour Q. La contrainte de 8-adjacence dans P assure que les carrés 2×2 ne se recouvrent pas dans Q (voir Figure 2.14). M8adj(H,V) est alors au moins aussi difficile que le problème de reconstruction d'un tableau 3-coloré car ce dernier est plus facile que le problème de packing de carrés 2×2 [1]. ■

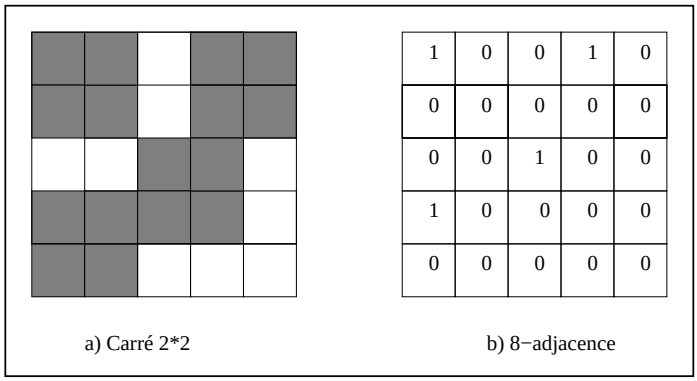


FIG. 2.14 – Equivalence entre 8-adjacence et carrés 2×2

2.8 Conclusion

Nous avons étudié, dans ce chapitre, le problème de reconstruction de matrices binaires sous contraintes d'adjacence. Nous avons considéré trois problèmes qui se distinguent par la contrainte d'adjacence : *M4adj(H)*, *M6adj(H,V)* et *M8adj(H,V)*. Nous avons montré que *M4adj(H)* est polynomial, mais lorsque les deux projections sont prises en compte, le problème *M4adj(H,V)* est de complexité inconnue pour un nombre de lignes $m > 3$ et polynomial pour $m \leq 3$. Le problème de reconstruction avec contrainte de 6-adjacence est NP-complet. Le problème de reconstruction avec contrainte de 8-adjacence est au moins aussi difficile que le problème de reconstruction d'un tableau 3-coloré. Néanmoins, ce travail reste à poursuivre. Nous envisageons d'étudier davantage la complexité du problème *M4adj(H,V)* pour $m \geq 4$. Les techniques de damiers et d'escaliers utilisées pour résoudre le problème *M4adj(H)* ne sembleraient pas adéquates pour *M4adj(H,V)* car elles relâchent une projection.

Bibliographie

[1] M. Chrobak, P. Couperus, C. Dürr, and G. Woeginger. A note on tiling under tomographic constraints. *Theoretical Computer Science*, 290 :2125–2136, 2003.

[2] H.J. Ryser. Combinatorial properties of matrices of zeros and ones. *Canad. J. Math*, 9 :371–377, 1957.

Chapitre 3

Reconstruction de matrices périodiques

Ce chapitre se situe dans le cadre de la reconstruction de matrices binaires sous contraintes de périodicité alternée (voir [1] pour les matrices périodiques). A est une matrice (p, q) alternée périodique si $A_{i,j} + A_{i+p,j+q} = 1$ pour $i = 1, \ldots, m - p$ et $j = 1, \ldots, n - q$, c'est-à-dire, les cellules (i, j) et $(i + p, j + q)$ ont des valeurs opposées. Nous dégageons les propriétés des matrices binaires alternées périodiques et nous résolvons les cas $(p, q) = (1, 1)$ et $(p, q) = (0, q)$. Ce problème trouve une de ses applications dans la reconstruction de structures cristallines.

3.1 Matrices (p,q) alternées périodiques (*MAP(p,q)*)

Définition 7:
Soit A une matrice binaire $m \times n$. A est une matrice (p, q) alternée périodique si
$A_{i,j} + A_{i+p,j+q} = 1$ *pour* $i = 1, \ldots, m-p$ *et* $j = 1, \ldots, n-q$.

Nous notons (H, V) les projections orthogonales d'une matrice binaire alternée périodiques. Nous avons la propriété suivante :

Propriété 2:
Toute matrice (p,q) alternée périodique vérifie :
i) $|h_i + h_{i+p} - n| \leq q,\ i = 1, \ldots, m-p,$
ii) $|v_j + v_{j+q} - m| \leq p,\ j = 1, \ldots, n-q.$

Preuve:
Supposons que A est une matrice (p,q) alternée périodique.
Par définition $h_i = \sum_{j=1}^{n} A_{i,j} = \sum_{j=1}^{n-q} A_{i,j} + \sum_{j=n-q+1}^{n} A_{i,j}$.
La (p, q) périodicité alternée permet d'établir que :
$h_i = \sum_{j=1+q}^{n}(1 - A_{i+p,j}) + \sum_{j=n-q+1}^{n} A_{i,j} = n - h_{i+p} - \sum_{j=1}^{q}(1 - A_{i+p,j}) + \sum_{j=n-q+1}^{n} A_{i,j}$
pour $1 \leq i \leq m-p$,
d'où $|h_i + h_{i+p} - n| = |\sum_{j=n-q+1}^{n} A_{i,j} - \sum_{j=1}^{q}(1 - A_{i+p,j})| \leq q.$ ■
De manière analogue pour la projection verticale, on obtient :
$|v_j + v_{j+q} - m| \leq p$ *pour* $1 \leq j \leq n-q$.

Nous supposons dans la suite que les projections orthogonales (H, V) vérifient cette propriété. Pour tirer profit de cette propriété dans la reconstruction des matrices alternées périodiques, nous introduisons les **boxes** définies de la façon suivante (voir Figure 3.1) :

$RHbox_i$ est l'ensemble des cellules $\{(i, n-q+1), \ldots, (i, n)\}$
$LHbox_i$ est l'ensemble des cellules $\{(i, 1), \ldots, (i, q)\}$
$UVbox_j$ est l'ensemble des cellules $\{(1, j), \ldots, (p, j)\}$
$DVbox_j$ est l'ensemble des cellules $\{(m-p+1, j), \ldots, (m, j)\}$
Appelons le poids d'une boxe le nombre de ses cellules de valeur 1.

Remarque : La somme des poids des boxes $RHbox_i$ et $LHbox_{i+p}$ vaut $h_i + h_{i+p} - n + q$. En effet, d'après la propriété de la (p, q) périodicité alternée, pour les lignes i et $i+p$, la somme des valeurs des cellules en dehors des boxes $RHbox_i$ et $LHbox_{i+p}$ vaut $n - q$. De même, on déduit que la somme des poids des boxes $DVbox_j$ et $UVbox_{j+q}$ vaut $v_j + v_{j+q} - m + p$.

Nous allons maintenant étudier le cas particulier $(p, q) = (1, 1)$.

3.2 Matrices (1,1) alternées périodiques *(MAP(1,1))*

Définition 8:
Etant donnée une matrice binaire A, la cellule (i, j) appartient au bord de A si $i = 1$ ou $i = m$ ou $j = 1$ ou $j = n$.

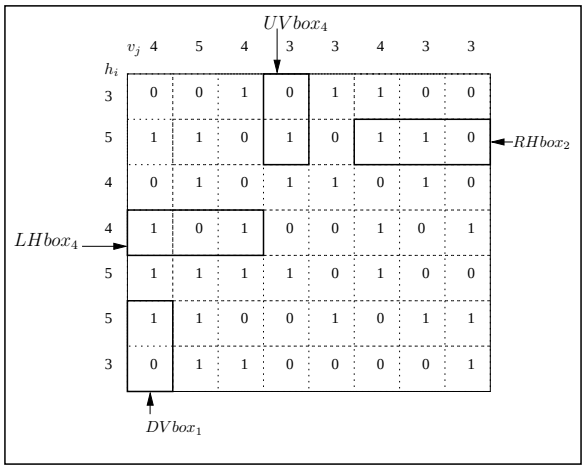


FIG. 3.1 – Matrice (2,3) alternée périodique et ses boxes

Ici, chaque boxe est réduite à une cellule, la remarque précédente s'écrit comme suit : Pour $1 \leq i \leq m-1$:

- si $h_i + h_{i+1} = n+1$ alors $A(i,n) = A(i+1,1) = 1$ (les deux boxes sont de poids 1),
- si $h_i + h_{i+1} = n-1$ alors $A(i,n) = A(i+1,1) = 0$ (les deux boxes sont de poids 0),
- si $h_i + h_{i+1} = n$ alors $A(i,n) + A(i+1,1) = 1$ (une boxe est de poids 1, l'autre est de poids 0).

Pour $1 \leq j \leq n-1$:

- si $v_j + v_{j+1} = m+1$ alors $A(m,j) = 1$ et $A(1,j+1) = 1$,
- si $v_j + v_{j+1} = m-1$ alors $A(m,j) = 0$ et $A(1,j+1) = 0$,
- si $v_j + v_{j+1} = m$ alors $A(m,j) + A(1,j+1) = 1$.

Ainsi les cellules du bord ont soit leurs valeurs déterminées par la valeur $h_i + h_{i+1}$ (ou $v_j + v_{j+1}$) ou bien sont couplées de sorte que $A(i,n) + A(i+1,1) = 1$ ou $A(m,j) + A(1,j+1) = 1$.

Définition 9:
Deux cellules (i,j) *et* (i',j') *sont voisines si* $(i',j') = (i+1 \bmod m, j+1 \bmod n)$.

Cette relation de voisinage définit le graphe $G(X,E)$. X représente l'ensemble des cellules de A. E représente la relation de voisinage entre les cellules. Le graphe G est alors une collection de cycles car chaque cellule a exactement deux voisines. Un cycle passant par la cellule (i,j) est l'ensemble des cellules $\{((i+k) \bmod m, (j+k) \bmod n), k \in \mathbb{Z}\}$. Par convention un cycle c_j commence à la cellule $(1,j)$ où j est la plus petite ordonnée de ses cellules, c'est-à-dire, par la cellule le plus en haut à gauche (voir Figure 3.2). Remarquons que la détermination de la valeur d'une cellule d'un cycle permet de déterminer l'ensemble des valeurs des cellules du cycle.

Proposition 18:
La longueur des cycles est $PPCM(m,n)$.

Preuve:
Nous avons $(i+k) \bmod m = i$ *si et seulement* k *est un multiple de* m. *De même,* $j+k \bmod n = j$ *si et seulement si* k *est un multiple de* n. *Ainsi* $(i,j) = ((i+k) \, mod \, m, (j+k) \bmod n)$ *si et seulement si* k *est un multiple de* $PPCM(m,n)$ *et la longueur de tout cycle est* $PPCM(m,n)$. ■

Un cycle alterné est un cycle où les valeurs des cellules sont alternativement 1 et 0. Nous remarquons qu'un cycle alterné est nécessairement de longueur paire. Nous définissons d'une manière identique une chaîne alternée.

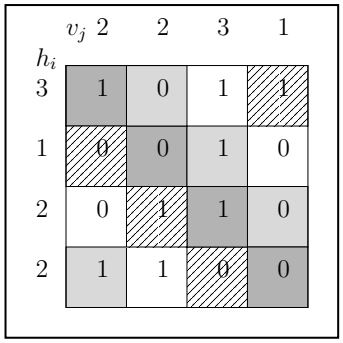


FIG. 3.2 – Matrice (1,1) alternée périodique et ses cycles

Reconstruction d'une solution

On détermine, tout d'abord, les cellules du bord couplées et celles dont la valeur est fixée. Pour chaque cellule (i,j) de valeur fixée, la valeur des cellules du cycle passant par (i,j) est déterminée. A cette étape, on vérifie la cohérence, c'est-à-dire, s'il n'existe pas une cellule qui prend à la fois les valeurs 0 et 1. Si une incohérence se dégage alors il n'existe pas de solution. Si toutes les valeurs de A sont fixées alors le problème est résolu.

Supposons que dans A, il existe des boxes de valeur non déterminée. Ces boxes indéterminées sont deux à deux couplées à l'exception des cellules $(1,1)$ et (m,n). Donc la partie non déterminée de A correspond à une collection de cycles alternés et une chaîne alternée. Cette chaîne a pour origine la cellule $(1,1)$ et pour extrémité la cellule (m,n). Notons x_j la valeur de la cellule $(1,j)$ où le cycle alterné c_j commence.

Pour chaque cycle alterné c_j, on détermine le nombre de ses cellules de valeur 1 par ligne et par colonne selon la valeur de x_j (voir Tab.3.1). Les valeurs a, a', b et b' peuvent être considérées comme les projections des cycles alternés sur les lignes et les colonnes.

	$x_j = 1$	$x_j = 0$
Ligne impaire	a	a'
Ligne paire	a'	a
Colonne j' : $j' = j \bmod 2$	b	b'
Colonne j' : $j' = 1 + j \bmod 2$	b'	b

	m pair	m impair	n pair	n impair
a	$\frac{PPCM(m,n)}{m}$	$\lceil \frac{PPCM(m,n)}{2m} \rceil$	-	-
a'	0	$\lfloor \frac{PPCM(m,n)}{2m} \rfloor$	-	-
b	-	-	$\frac{PPCM(m,n)}{n}$	$\lceil \frac{PPCM(m,n)}{2n} \rceil$
b'	-	-	0	$\lfloor \frac{PPCM(m,n)}{2n} \rfloor$

TAB. 3.1 – Projections des cycles et valeurs de a, a', b et b'

En notant h'_i (resp. v'_j) le nombre de $'1'$ restant à placer sur la ligne $i,\ i = 1, \ldots, m$, (resp. colonne $j,\ j = 1, \ldots, n$) et (H', V') les vecteurs correspondant, pour qu'une matrice $(1,1)$ alternée périodique satisfaisant (H, V) existe, il faut que l'ensemble des cycles alternés satisfassent les projections (H', V').

Puisque les projections orthogonales de chaque cycle c_j ne dépendent que de la valeur de x_j et de la parité de j, nous introduisons les variables $x = \sum_{j,\,pair} x_j$ et $y = \sum_{j,\,impair} x_j$. Notons par I_p (resp. I_i) le nombre de cycles alternés c_j avec j pair (resp. impair). Les variables x et y sont alors liées par le système linéaire suivant :

$$\mathcal{S} \begin{cases} ax + ay + a'(I_p - x) + a'(I_i - y) = h'_1 & (1) \\ a'x + a'y + a(I_p - x) + a(I_i - y) = h'_2 & (2) \\ b'x + by + b(I_p - x) + b'(I_i - y) = v'_1 & (3) \\ bx + b'y + b'(I_p - x) + b(I_i - y) = v'_2 & (4) \\ x \leq I_p, y \leq I_i \end{cases}$$

Il suffit de considérer uniquement les deux premières lignes et les deux premières colonnes car pour x_j donné, les projections du cycle c_j ne dépendent que de la parité des lignes et des colonnes.

La contrainte (1) garantit la satisfaction de la projection horizontale des lignes impaires et la contrainte (2) assure celle des lignes paires. De même, les contraintes (3) et (4) assurent la satisfaction des projections verticales des colonnes.

Nous déduisons de $\mathcal{S}$ qu'une condition nécessaire d'existence d'une matrice (1,1) alternée périodique est : $h'_1 + h'_2 = (a + a')I_p + (a + a')I_i = \frac{PPCM(m,n)}{m}(I_p + I_i)$ et $v'_1 + v'_2 = (b + b')I_p + (b + b')I_i = \frac{PPCM(m,n)}{n}(I_p + I_i)$. En effet, un cycle alterné admet $\frac{PPCM(m,n)}{m}$ $'1'$ sur les deux premières lignes de A et $\frac{PPCM(m,n)}{n}$ $'1'$ sur les deux premières colonnes. Nous supposons que ces conditions sont bien satisfaites. Le système $\mathcal{S}$ peut être simplifié comme suit :

$$\mathcal{S}' \quad \begin{cases} (a - a')(x + y) = h'_1 - a'I_p - a'I_i & (1') \\ (b' - b)(x - y) = v'_1 - bI_p - b'I_i & (2') \\ x \leq I_p, y \leq I_i \end{cases}$$

Trois cas se présentent selon les valeurs de a, a', b et b' :

- Si $a \neq a'$ et $b \neq b'$, $\mathcal{S}'$ admet une solution unique $x = \frac{(h'_1 - a'I_p - a'I_i)(b-b') - (v'_1 - bI_p - b'I_i)(a-a')}{2(a-a')(b-b')}$ et $y = \frac{(v'_1 - bI_p - b'I_i)(a-a') + (h'_1 - a'I_p - a'I_i)(b-b')}{2(a-a')(b-b')}$ si x et y sont entiers avec $x \leq I_p$ et $y \leq I_i$.
- Si $a = a'$, nous avons m impair (voir Tab 3.1). Si n est impair, nous avons $PPCM(m,n)$ qui est impair et donc $\lceil \frac{PPCM(m,n)}{2n} \rceil = \lfloor \frac{PPCM(m,n)}{2n} \rfloor + 1$, donc d'après le tableau 3.1, on obtient $a \neq a'$, d'où une contradiction. Nécessairement n est pair, alors $b \neq b'$ et $b' = 0$ (voir Tab. 3.1). Dans ce cas, l'équation (1') est $h'_1 = a(I_p + I_i)$ qui est satisfaite d'après les définitions de I_p, I_i et a. Le système $\mathcal{S}'$ se réduit à : $y - x = \frac{v'_1}{b} - I_p$ avec $x \leq I_p$ et $y \leq I_i$. Puisque $b' = 0$, v'_1 et v'_2 sont des multiples de b (voir Tab. 3.1). L'équation $y - x = \frac{v'_1}{b} - I_p$ admet les solutions suivantes :
  $(\frac{v'_2}{b} - k, I_i - k)$ si $\frac{v'_2}{b} \leq I_p$ avec $0 \leq k \leq min(I_i, \frac{v'_2}{b})$, k entier.

$(I_p - k, \frac{v'_1}{b} - k)$ si $\frac{v'_2}{b} > I_p$ avec $0 \leq k \leq min(I_p, \frac{v'_1}{b})$, k entier.

- Si $b = b'$. Par un raisonnement identique au précédent, $\mathcal{S}'$ admet les solutions suivantes :
  $(\frac{h'_1}{a} - k, k)$ si $\frac{h'_1}{a} \leq I_p$ avec $0 \leq k \leq min(I_i, \frac{h'_1}{a})$, k entier.
  $(I_p - k, I_i - \frac{h'_2}{a} + k)$ si $\frac{h'_1}{a} > I_p$ avec $0 \leq k \leq min(I_p, \frac{h'_2}{a})$, k entier.

Lorsque le système $\mathcal{S}'$ admet une solution, les valeurs x_j sont obtenues en affectant la valeur 1 à x variables x_j telles que j est pair et en affectant la valeur 1 à y variables x_j telles que j est impair. La valeur 0 est affectée aux variables x_j restantes.

Une solution au problème *MAP(1,1)* est alors donnée par l'algorithme A-MAP(1,1).

Algorithme A-MAP(1,1)

1. Tester la cohérence des projections orthogonales et calculer (H', V') ;
2. Calculer (x, y) une solution de $\mathcal{S}'$;
3. En déduire les valeurs des cellules de chaque cycle c_j ;

Avant d'aller plus loin, illustrons ce qui précède par un exemple simple.

Nous cherchons une matrice (1,1) alternée périodique respectant les projections $H = (3, 1, 2, 2)$ et $V = (2, 2, 3, 1)$ (voir Figure 3.3). Les cellules (boxes) $(2, 4)$ et $(3, 1)$ ont une valeur égale à 0 car $h_2 + h_3 - n + q = 0$. Les contraintes de périodicité alternée font que les cellules $(1, 3)$ et $(4, 2)$ ont une valeur égale à 1. On obtient $H' = (2, 1, 2, 1)$ et $V' = (2, 1, 2, 1)$. Les cycles à valeur non déterminée de A sont c_1, c_2 et c_4 de longueur $PPCM(4, 4) = 4$. Nous avons $I_p = 2$, $I_i = 1$, $a = b = 1$ et $a' = b' = 0$.

Le système $\mathcal{S}'$ a pour solution $(x, y) = (1, 1)$. Donc $x_1 = 1$ et $x_2 + x_4 = 1$, on choisit arbitrairement $x_2 = 0$ et $x_4 = 1$, c'est-à-dire, le cycle c_2 commence avec une cellule de valeur 0 et les cycles c_1 et c_4 commencent avec des cellules de valeur 1. La solution obtenue est donnée dans la figure 3.3. Si l'on avait choisi $x_2 = 1$ et $x_4 = 0$, on aurait obtenu une solution équivalente respectant les mêmes projections orthogonales.

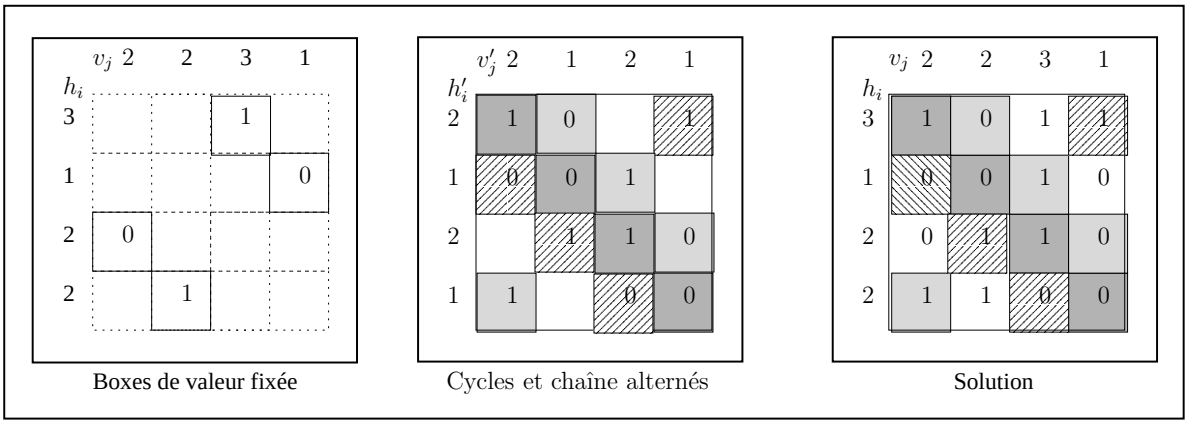


FIG. 3.3 – Résolution d'un problème $MAP(1, 1)$

Proposition 19:
La complexité du problème $MAP(1, 1)$ est $O(mn)$.

Preuve:
Il faut $O(m+n)$ opérations pour trouver les cellules de valeur fixée et les cellules couplées sur le bord de la matrice. L'ensemble des cycles partitionnent la matrice. Ainsi l'ensemble des opérations de mise à jour des valeurs de la matrice a pour complexité $O(mn)$. La résolution de $\mathcal{S}'$ se faisant en temps constant, nous avons la complexité $O(mn)$ pour l'algorithme A-MAP(1,1). ■

Nous abordons maintenant un autre cas polynomial de reconstruction de matrices alternées périodiques.

3.3 Matrices (0,q) alternées périodiques *(MAP(0,q))*

Dans cette section, nous reconstruisons des matrices $(0, q)$ alternées périodiques.
Remarque : En vertu de la $(0, q)$ périodicité alternée, il y a q $'1'$ et q $'0'$ par colonne dans chaque bloc de $2p$ lignes successives.

Nous distinguons deux cas selon les valeurs de m et q (voir Figure 3.4) :

3.3.1 Cas 1 : $m = 2kq + r,\ r \leq q$

L'ensemble des lignes est composé par un bloc de r lignes $(1, \ldots, r)$ suivi par k blocs de $2q$ lignes. Notons $v'_j,\ j = 1, \ldots, n$, la projection de la colonne j sur le bloc de r lignes $(1, \ldots, r)$. D'après la remarque précédente, nous obtenons $v'_j = v_j - kq,\ j = 1, \ldots, n$. Une solution est reconstruite comme suit :

1. sur le bloc de r lignes $(1, \ldots, r)$, reconstruire une matrice binaire $r \times n$ respectant les projections horizontales $h_1, \ldots, h_r$ et les projections verticales $v'_j,\ j = 1, \ldots, n$,
2. pour chaque ligne $i,\ i = r + 1, \ldots, q$, affecter à 1 la valeur de h_i cellules,
3. translater par $(0, q)$ périodicité alternée la valeur de q premières lignes.

3.3.2 Cas 2 : $m = (2k + 1)q + r,\ r \leq q$

L'ensemble des lignes est composé par un bloc de r lignes $(1, \ldots, r)$, un bloc de $q - r$ lignes $(r + 1, \ldots, q)$, un autre bloc de r lignes $(q + 1, \ldots, q + r)$ suivis par k blocs de $2q$ lignes. Notons $v'_j,\ j = 1, \ldots, n$, la projection de la colonne j sur le bloc de $q - r$ lignes $(r + 1, \ldots, q)$. Il y a r cellules de valeur 1 et r cellules de valeur 0 par colonne dans les deux blocs de r lignes $(1, \ldots, r)$ et $(q+1, \ldots, q+r)$. D'une manière analogue au premier cas, nous retrouvons $v'_j = v_j - kq - r,\ j = 1, \ldots, n$. Une solution est reconstruite comme suit :

1. sur le bloc de $q-r$ lignes $(r+1, \ldots, q)$, reconstruire une matrice $(q-r) \times n$ respectant les projections horizontales $h_{r+1}, \ldots, h_q$ et les projections verticales $v'_j,\ j = 1, \ldots, n$,
2. pour chaque ligne $i,\ i = 1, \ldots, r$, affecter à 1 la valeur de h_i cellules,
3. translater par $(0, q)$ périodicité alternée la valeur de q premières lignes.

On déduit le résultat suivant de ce qui précède.

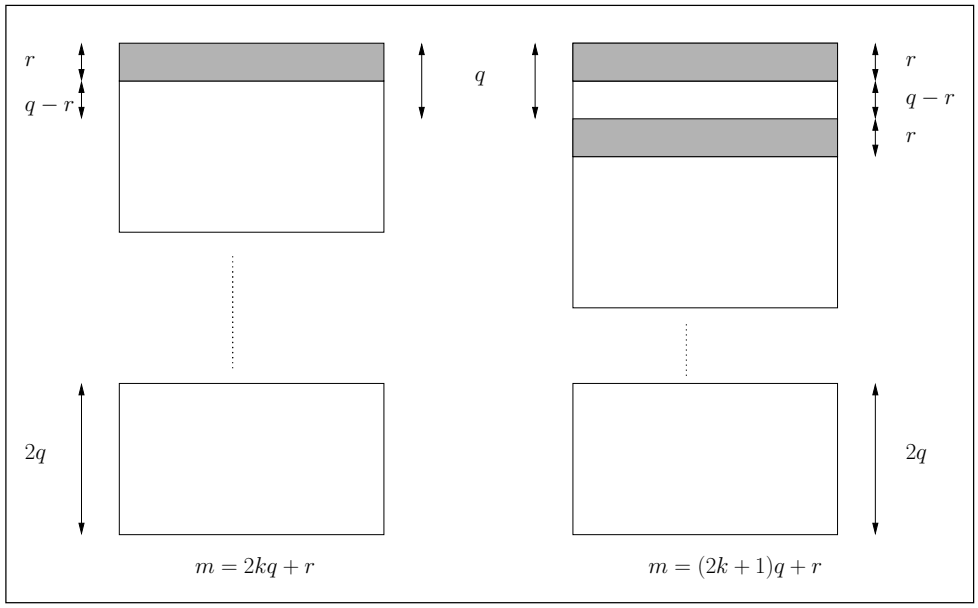


FIG. 3.4 – Matrices (0,q) alternées périodiques

Proposition 20:
Le problème MAP(0,q) est polynomial.

3.4 Conclusion

Dans ce chapitre, nous avons traité le problème de reconstruction de matrices binaires alternées périodiques. Nous avons établi des propriétés générales de ces matrices et nous avons résolu deux sous-problèmes polynomiaux. Cette étude est, à notre connaissance, parmi les premières dans ce domaine et elle ouvre la voie à l'étude de nouveaux problèmes.

L'algorithme A-MAP(1,1) ne se généralise pas au cas général car les boxes ne sont plus réduites à des cellules. Parmi ceux-ci, la complexité du problème *MAP(1,q)* est encore inconnue.

Bibliographie

[1] A. Del Lungo, A. Frosini, M. Nivat, and L. Vuillon. Reconstruction under periodicity constraints. *ICALP*, pages 38–56, 2002.

Chapitre 4

Reconstruction de tableaux colorés

L'objet de ce chapitre est d'étudier la complexité du problème $T3C$ et de proposer une heuristique pour la reconstruction d'images numériques à partir de leurs projections.

Dans la première partie, nous décrivons le problème $T3C$ et montrons la grande utilité de le résoudre. Nous proposons aussi deux formulations par des programmes mathématiques en 0 et 1 qui diffèrent essentiellement par la fonction objectif à prendre en compte. Dans la deuxième partie, nous abordons des sous-problèmes polynomiaux qui couvrent une grande partie des applications possibles.

Dans la troisième partie, nous traitons le problème de reconstruction d'images numériques avec niveaux de gris. Nous formulons ce problème à l'aide de la programmation linéaire en nombres entiers (PLNE). Pour une reconstruction approchée d'images, nous concevons une heuristique basée sur la relaxation continue du modèle en PLNE.

4.1 Reconstruction de tableaux 3-colorés (T3C)

Etant donnés quatre vecteurs $H^a = (h_1^a, \ldots, h_m^a)$, $V^a = (v_1^a, \ldots, v_n^a)$, $H^b = (h_1^b, \ldots, h_m^b)$ et $V^b = (v_1^b, \ldots, v_n^b)$, le problème de reconstruction d'un tableau 3-coloré, noté $T3C$,

consiste à colorier les cellules d'un tableau $m \times n$ par trois couleurs : la couleur de fond, la couleur a ou la couleur b. Par abus de langage, on dit que la cellule est non colorée lorsqu'elle est colorée par le fond. Les cellules colorées par a et b vérifient respectivement les projections (H^a, V^a) et (H^b, V^b) (voir Figure 4.1). Les projections donnent le nombre de cellules de couleurs a et b se trouvant dans chacune des lignes et des colonnes. On ne tient pas compte des projections de la couleur de fond parce qu'elles sont satisfaites lorsque celles de a et b le sont.

D'une manière générale, on définit le problème TkC de reconstruction de tableaux k-colorés où chaque cellule est colorée par une couleur parmi un ensemble de k couleurs. Ce problème est NP-complet pour $k \geq 4$ ([2] et [3]), de complexité inconnue pour $k = 3$ et polynômial pour $k = 2$ [4].

La détermination de la complexité du problème $T3C$ est de grande importance parce qu'il est plus facile que plusieurs problèmes non résolus. Nous citons, le problème de pavage par des carrées 2×2 [1], le problème de reconstruction de matrices binaires avec contrainte de 8-adjacence (voir section 2.7), le problème $T3C$ avec au moins deux cellules colorées adjacentes (voir section 4.2.5), etc. D'autres problèmes non résolus ont le même degré de complexité que $T3C$, par exemple, le problème de packing de dominos et triminos (voir section 6.3.1).

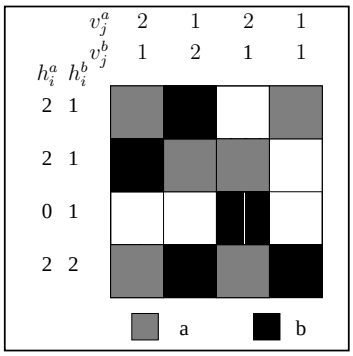


FIG. 4.1 – Exemple d'un tableau 3-coloré

Les notations suivantes sont adoptées tout au long de ce chapitre.

Tableaux et matrices

- $H = H^a + H^b = (h_1^a + h_1^b, \ldots, h_m^a + h_m^b)$: projection horizontale agrégée.
- $MB(H, V)$: problème de reconstruction d'une matrice binaire respectant (H, V).
- TkC : problème de reconstruction d'un tableau k-coloré.
- $S(i, j)$: la valeur de la cellule (i, j) si S est une matrice binaire ou la couleur de la cellule (i, j) si S est un tableau 3-coloré.
- Sa : matrice solution au problème $MB(H^a, V^a)$.
- Sb : matrice solution au problème $MB(H^b, V^b)$.
- $S = S_a \oplus S_b$: tableau 3-coloré tel que la cellule $S(i, j)$ est de couleur a (resp. b) si $S_a(i, j) = 1$ (resp. $S_b(i, j) = 1$). La cellule (i, j) est colorée par le fond du tableau si $S_a(i, j) = S_b(i, j) = 0$.

Cellules colorées

- Cellule non colorée : cellule colorée par le fond du tableau.
- Cellule colorée : cellule colorée par a ou b.
- Cellule bicolorée : cellule colorée par a et b. On dit aussi qu'il y a un **conflit**.

Transformations entre tableaux

- Cellule invariante : cellule colorée par la même couleur dans toute solution (voir section 1.2.3).
- Tableaux équivalents : tableaux ayant les mêmes projections pour toutes les couleurs.
- Bascule : sous-tableau $\begin{array}{|c|c|}\hline x & y \\ \hline y & x \\ \hline \end{array}$ où x et y sont deux couleurs différentes.
- Opération de bascule : transformation qui consiste à remplacer la bascule $\begin{array}{|c|c|}\hline x & y \\ \hline y & x \\ \hline \end{array}$ par la bascule $\begin{array}{|c|c|}\hline y & x \\ \hline x & y \\ \hline \end{array}$. Les projections orthogonales de toutes les couleurs demeurent inchangées après les opérations de bascules. Une opération de bascule est un cas particulier des permutations.
- Permutation : transformation permettant de passer d'un tableau 3-coloré avec conflits à un autre tableau 3-coloré avec moins de conflits. Il existe trois types de permutations :

$$\begin{array}{|c|c|}\hline ab & 0 \\ \hline 0 & a \\ \hline \end{array} \Longrightarrow \begin{array}{|c|c|}\hline b & a \\ \hline a & 0 \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline ab & 0 \\ \hline 0 & b \\ \hline \end{array} \Longrightarrow \begin{array}{|c|c|}\hline a & b \\ \hline b & 0 \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline ab & 0 \\ \hline 0 & ab \\ \hline \end{array} \Longrightarrow \begin{array}{|c|c|}\hline b & a \\ \hline a & b \\ \hline \end{array} \Longleftrightarrow \begin{array}{|c|c|}\hline a & b \\ \hline b & a \\ \hline \end{array}$$

Toutes les matrices intervenant dans ce chapitre sont de taille $m \times n$. Nous supposons connus les théorèmes classiques sur l'addition des vecteurs et sur la multiplication d'un vecteur par un scalaire.

Notons que les problèmes $MB(H, V)$ et $T2C$ sont équivalents car les cellules colorées s'assimilent aux $'1'$ et les cellules non colorées s'assimilent aux $'0'$.

Commençons par établir des conditions nécessaires d'existence de solution pour $T3C$.

Proposition 21:

Les conditions suivantes sont nécessaires pour que le problème $T3C$ admette une solution :

i) $h_i^a + h_i^b \leq n,\ i = 1, \ldots, m,$

ii) $v_j^a + v_j^b \leq m,\ j = 1, \ldots, n,$

iii) $MB(H^a, V^a)$, $MB(H^b, V^b)$ *et* $MB(H^a + H^b, V^a + V^b)$ *admettent des solutions,*

iv) il n'existe pas de cellules à la fois ***invariantes*** *pour a et b.*

Preuve:

i) Sur chaque ligne i, il y a h_i^a *cellules de couleur a et* h_i^b *cellules de couleur b, soit* $h_i^a + h_i^b$ *cellules colorées . De même ii) est nécessaire.*

iii) Toute solution satisfait, entre autres, les projections de chaque couleur et les projections agrégées.
iv) S'il existe une cellule à la fois invariante pour a et b, alors il existera un conflit. Cette condition peut être testée en temps polynomial car le théorème 3 du chapitre 1 permet de déterminer en temps polynomial les cellules invariantes pour le problème $MB(H,V)$. ■

4.1.1 Formulation du problème $T3C$

Nous proposons deux modèles mathématiques en 0 et 1 pour modéliser le problème $T3C$. Le premier modèle, $Q1$, cherche une solution avec un nombre minimal de conflits. Lorsqu'une cellule est colorée par les deux couleurs, on dit qu'il y a **un conflit** dans cette cellule. Le deuxième modèle, $Q2$, est obtenu à partir de $Q1$ en relâchant la contrainte de la projection verticale de la couleur b et en cherchant à la satisfaire au mieux. La fonction objectif à minimiser est la différence entre la projection verticale de la solution trouvée et la projection souhaitée.

Nous introduisons deux variables bivalentes : x_{ij} qui vaut 1 si et seulement si la cellule (i,j) est colorée par a et y_{ij} qui vaut 1 si et seulement si la cellule (i,j) est colorée par b :

$$x_{ij} = \begin{cases} 1 & \text{si la cellule } (i,j) \text{ est colorée par a} \\ 0 & \textit{sinon} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si la cellule } (i,j) \text{ est colorée par b} \\ 0 & \textit{sinon} \end{cases}$$

Le premier modèle se formalise par le programme mathématique ($Q1$) qui consiste à minimiser une fonction quadratique **non convexe** soumise à des contraintes **linéaires**.

$$Q1 \begin{cases} min\ F1 = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} y_{ij} & & \\ \sum_{j=1}^{n} x_{ij} = h_i^a & i = 1, \ldots, m & (1) \\ \sum_{j=1}^{n} y_{ij} = h_i^b & i = 1, \ldots, m & (2) \\ \sum_{i=1}^{m} x_{ij} = v_j^a & j = 1, \ldots, n & (3) \\ \sum_{i=1}^{m} y_{ij} = v_j^b & j = 1, \ldots, n & (4) \\ x_{ij},\ y_{ij} \in \{0,1\} & i = 1, \ldots, m,\ j = 1, \ldots, n & \end{cases}$$

Dans ce modèle, on autorise que les cellules soient bicolorées et on cherche à minimiser le nombre de conflits dans le tableau. Les contraintes (1) et (3) garantissent la satisfaction des projections orthogonales de la couleur a et les contraintes (2) et (4) celles de b. Il est évident que le problème $T3C$ admet une solution si et seulement si la valeur de la fonction objectif à l'optimum est nulle, c'est-à-dire, il n'existe pas de conflits.

Une autre formulation possible ($Q2$) consiste à relâcher la contrainte (4) du $Q1$ et à en tenir compte dans la fonction objectif et à ajouter une contrainte d'exclusivité (4') sur les couleurs (une cellule est non colorée ou bien colorée par a ou b). Ainsi, on cherche à minimiser la somme des différences absolues entre les projections exigées v_j^b et les projections trouvées $\sum_{i=1}^{m} y_{ij}$.

$$Q2 \begin{cases} min\ F2 = \sum_{j=1}^{n} |\sum_{i=1}^{m} y_{ij} - v_j^b| & \\ \sum_{j=1}^{n} x_{ij} = h_i^a & i = 1, \ldots, m \quad (1) \\ \sum_{j=1}^{n} y_{ij} = h_i^b & i = 1, \ldots, m \quad (2) \\ \sum_{i=1}^{m} x_{ij} = v_j^a & j = 1, \ldots, n \quad (3) \\ x_{ij} + y_{ij} \leq 1 & i = 1, \ldots, m,\ j = 1, \ldots, n \quad (4') \\ x_{ij},\ y_{ij} \in \{0, 1\} & i = 1, \ldots, m,\ j = 1, \ldots, n \end{cases}$$

Notons $F1^*$ et $F2^*$ la valeur de la solution optimale respectivement de $Q1$ et $Q2$.

Proposition 22:
$F2^* \leq 2F1^*$.

Preuve:
Soit $S1^*$ une solution optimale pour $Q1$. On construit une solution $S2$ admissible pour $Q2$ avec $F2(S2) \leq 2F1^*$ de la manière suivante :

$S2 = S1^*$, $F2(S2) = 0$;
tant qu'il existe un conflit dans (i,j) faire
- *chercher une cellule (i, j') non colorée dans $S2$;*
- *colorer par b la cellule (i, j') dans $S2$;*
- *mettre à jour $S2$;*

fin tant que ;

A la sortie de la boucle "tant que", $S2$ ne contient plus de conflits car s'il y avait un conflit dans la cellule (i, j) alors il existerait une colonne j' telle que la cellule (i, j') ne serait pas colorée ($h_i^a + h_i^b \leq n$). $S2$ est bien une solution de $Q2$ car elle viole uniquement la contrainte de la projection verticale de b. La levée de chaque conflit fait augmenter $F2(S2)$ de moins de deux unités, d'où $F2(S2) \leq 2F1(S1^*) = 2F1^*$. ■

Nous établissons l'existence d'une solution pour le problème $T3C$ à partir de la relaxation continue de $Q1$.

4.1.2 Relaxation continue de Q1

Nous transformons le modèle $Q1$ en un modèle continu $\overline{Q1}$ en relâchant la contrainte d'intégrité sur les variables x_{ij} et y_{ij}.

$$\overline{Q1} \begin{cases} min \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} y_{ij} & \\ \sum_{j=1}^{n} x_{ij} = h_i^a & i = 1, \ldots, m \\ \sum_{j=1}^{n} y_{ij} = h_i^b & i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} = v_j^a & j = 1, \ldots, n \\ \sum_{i=1}^{m} y_{ij} = v_j^b & j = 1, \ldots, n \\ x_{ij},\ y_{ij} \in [0, 1] & i = 1, \ldots, m,\ j = 1, \ldots, n \end{cases}$$

L'utilité de cette relaxation provient du résultat suivant :

Proposition 23:
Si $\overline{Q1}$ admet une solution optimale $S1(x_{ij}, y_{ij})$ de valeur nulle, alors $S1$ se transforme polynomialement en une solution au problème $T3C$.

Preuve:
Notons, tout d'abord, que si le problème $T3C$ admet une solution alors les modèles $Q1$ et $\overline{Q1}$ ont une valeur nulle à l'optimum.

Supposons que $\overline{Q1}$ admet une solution optimale, $S1$, de valeur nulle. L'idée globale de la preuve est d'extraire de $S1$ deux solutions Sa et Sb respectivement aux problèmes $MB(H^a, V^a)$ et $MB(H^b, V^b)$ avec $Sa(i,j) + Sb(i,j) \leq 1$ pour toute cellule (i,j). Le tableau 3-coloré $S = Sa \oplus Sb$ sera une solution au problème $T3C$.

Pour cela, à la couleur a, faisons correspondre un graphe biparti $Ga(X, Y, E)$ (voir Figure 4.2) où $X = \{l_i,\ i = 1, \ldots, m\}$ représente les lignes et $Y = \{c_j,\ j = 1, \ldots, n\}$ représente les colonnes. Ajoutons à $Ga(X, Y, E)$ une source S et un puits T. La source S est reliée à chaque sommet l_i par un arc de capacité h_i^a. Le puits T est relié à chaque sommet c_j par un arc de capacité v_j^a. Les sommets l_i et c_j seront reliés par un arc de capacité unitaire si $x_{ij} \neq 0$ pour $i = 1, \ldots, m$ et $j = 1, \ldots, n$. Le graphe Ga admet un flot maximal fractionnaire de valeur $\sum_{i=1}^{m} h_i^a$ dont le flot sur chaque arc (l_i, c_j) est x_{ij} car $S1$ respecte les projections de la couleur a. Il résulte du théorème des valeurs entières que le flot maximal, ϕ_a, de S à T est de valeur $\sum_{i=1}^{m} h_i^a$ et que toutes ses composantes sont entières car la capacité des arcs de Ga est entière. Soit Sa la matrice binaire associée au flot ϕ_a avec $Sa(i,j) = 1$ si et seulement si $\phi_a(l_i, c_j) = 1$. Sa satisfait les projections orthogonales (H^a, V^a) car ϕ_a sature les arcs reliant S et $l_i,\ i = 1, \ldots, m$, et les arcs reliant T et $c_j,\ j = 1, \ldots, n$.

De même, nous faisons correspondre à la couleur b un graphe Gb ayant les mêmes sommets que Ga. Les sommets S et l_i sont reliés par un arc de capacité h_i^b. Les sommets c_j et T sont reliés par un arc de capacité v_j^b. Si $y_{ij} \neq 0$ alors un arc de capacité unitaire relie l_i et c_j. Selon la démarche précédente, on retrouve une matrice (solution) Sb respectant les projections de la couleur b.

Les graphes Ga et Gb n'ont pas d'arcs de la forme (l_i, c_j) en commun car par hypothèse $x_{ij}y_{ij} = 0$ pour toute cellule (i,j). Il en découle qu'il n'existe pas une cellule (i,j) avec $Sa(i,j) = Sb(i,j) = 1$. Le tableau 3-coloré $S = S_a \oplus S_b$ est bien une solution à $T3C$. ■

Illustrons par un exemple la démarche à suivre pour reconstruire, à l'aide du modèle $\overline{Q1}$, un tableau 3-coloré respectant les projections (H^a, V^a) pour a et (H^b, V^b) pour b. Prenons $H^a = (1,1,1,2), V^a = (0,1,2,1,1), H^b = (1,1,1,0)$ et $V^b = (2,0,0,1,0)$.

Nous cherchons tout d'abord, une solution $S1(x_{ij}, y_{ij})$ au modèle $\overline{Q1}$ (voir Tableau 4.1). Ensuite, nous associons le graphe Ga à la solution $S1$ (voir Figure 4.2). La recherche d'un flot maximal dans Ga nous donne une solution Sa respectant les projections de la couleur a. Puis, nous construisons le graphe Gb et cherchons une solution Sb respectant les projections de la couleur b. Enfin, nous additionnons les deux solutions précédentes pour trouver une solution globale $S = Sa \oplus Sb$.

En prenant $H^b = 0$ et $V^b = 0$, on déduit le résultat suivant : Toute solution fraction-

$S1(x_{ij}, y_{ij})$

$\underline{\frac{1}{3}}$	$\frac{1}{2}$		$\underline{\frac{2}{3}}$	$\frac{1}{2}$
$\underline{1}$		1		
$\underline{\frac{2}{3}}$	$\frac{1}{4}$	$\frac{1}{2}$	$\underline{\frac{1}{3}}$	$\frac{1}{4}$
	$\frac{1}{4}$	$\frac{1}{2}$	1	$\frac{1}{4}$

S_a

	a			
		a		
		a		
			a	a

S_b

			b	
b				
b				

$S = S_a \oplus S_b$

	a		b	
b		a		
b		a		
			a	a

TAB. 4.1 – Résolution par $\overline{Q1}$

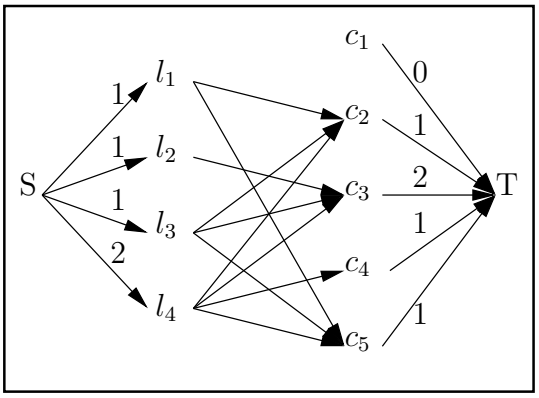


FIG. 4.2 – Le graphe Ga

naire du problème $T2C$ ou $MB(H, V)$ se transforme polynomialement en une solution entière.

4.2 Cas particuliers

L'inconvénient de l'approche précédente est que le programme $\overline{Q1}$ est de fonction objectif non convexe et on ne sait pas le résoudre en temps polynomial. Vu cette difficulté, nous nous limitons à étudier quelques sous-problèmes particuiers avant de proposer une heuristique dans la dernière partie.

4.2.1 $T3C$ avec une projection agrégée (P1)

On cherche à reconstruire un tableau 3-coloré à partir de la projection verticale de chaque couleur, la projection horizontale agrégée et la liste des couleurs admissibles par ligne. Le problème de décision correspondant, $\mathcal{E} - P1$, est le suivant :

Données : $H \in N^m, V^a, V^b \in N^n$ trois vecteurs à coordonnées entières positives et H^* une liste de dimension m.
Question : Existe-t-il un tableau $m \times n$ respectant V^a et V^b pour les projections verticales de a et b, H pour la projection horizontale agrégée et H^* pour la liste des couleurs admissibles par ligne ?

Concernant la polynomialité de $P1$, nous énonçons le résultat suivant :

Proposition 24:
Le problème $P1$ est polynomial.

Preuve:
Le problème $P1$ est aussi un problème de recherche d'un flot maximal dans un graphe biparti augmenté $G(L, L^*, A, B, E)$ (voir Figure 4.3). $L = \{l_i,\ i = 1, \ldots, m\}$ représente la projection horizontale agrégée ou les lignes. L^* est l'ensemble de lignes ou les deux couleurs a et b sont admissibles. $A = \{a_j,\ j = 1, \ldots, n\}$ représente la projection verticale de la couleur a et $B = \{b_j,\ j = 1, \ldots, n\}$ représente la projection verticale de la couleur b. Ajoutons à G deux sommets S et T. S est relié à l_i par un arc de capacité h_i pour $i = 1, \ldots, m$. T est relié à a_j et à b_j par des arcs des capacités respectives v_j^a et v_j^b pour $j = 1, \ldots, n$. l_i est lié à l_i^* par un arc de capacité unitaire si $i \in L^*$. Lorsque $i \notin L^*$, l_i est relié à a_j (resp. b_j) par un arc de capacité unitaire si $a \in h_i^*$ (resp. $b \in h_i^*$). De même l_i^* est lié aux sommets a_j et b_j par des arcs de capacité unitaire. Le problème $P1$ admet une solution si et seulement si la valeur du flot maximal dans G est égale à $\sum_{i=1}^{m} h_i = \sum_{j=1}^{n} v_j^a + \sum_{j=1}^{n} v_j^b$.

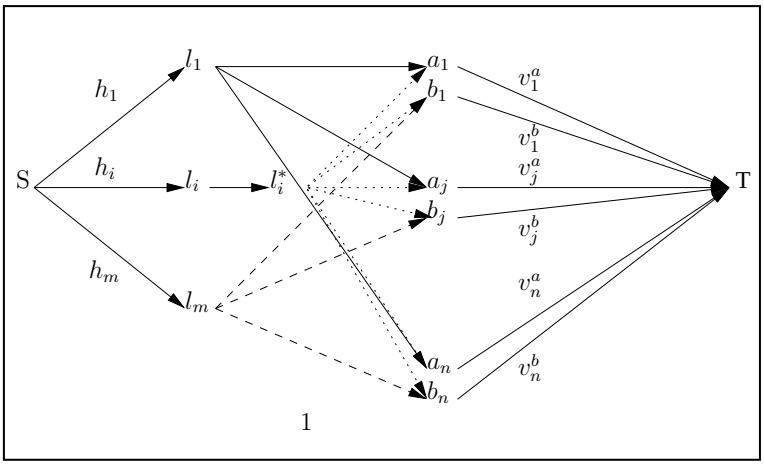


FIG. 4.3 – $T2C$ avec avec une projection horizontale agrégée

Dans l'exemple illustré par la figure 4.3, sur la ligne 1 (resp. m), les cellules peuvent être colorées seulement par a (resp. b). Les deux couleurs sont admissibles sur la ligne i : $h_1^* = \{a\}$, $h_i^* = \{a, b\}$ et $h_m^* = \{b\}$. ■

Ce résultat se généralise facilement aux problèmes TkC avec une projection horizontale ou verticale agrégée.

4.2.2 $T3C$ avec des projections bornées (P2)

Nous supposons que dans toute direction, le nombre de cellules colorées par chaque couleur est borné par c : $h_i^a \leq c,\ h_i^b \leq c,\ i = 1, \ldots, m$, et $v_j^a \leq c,\ v_j^b \leq c,\ j = 1, \ldots, n$.

Notons :

C_i : l'ensemble de colonnes coupant la ligne i en une cellule colorée.

L_j : l'ensemble de lignes coupant la colonne j en une cellule colorée.

Pour résoudre le problème $P2$, nous proposons l'algorithme suivant :

Algorithme A-P2

1. Fin $\longleftarrow$ faux ;
2. chercher une solution Sa à $MB(H^a, V^a)$;
3. chercher une solution Sb à $MB(H^b, V^b)$;
4. $S \longleftarrow S_a \oplus S_b$;
5. tant que fin est faux faire
  5.1 chercher un conflit (i, j), s'il n'en existe plus alors fin $\longleftarrow$ vrai ;
  5.2 s'il existe une cellule colorée (i', j') avec $i' \notin L_j$ et $j' \notin C_i$
    alors effectuer la permutation $(i, j), (i, j'), (i', j), (i', j')$;
  5.3 sinon, résoudre par un Branch and Bound le problème $T3C$ et fin $\longleftarrow$ vrai ;
fin tant que ;

La complexité de l'algorithme A-P2 est donnée par le résultat suivant :

Proposition 25:
Pour c fixé, le problème P2 est polynomial et l'algorithme A-P2 en fournit une solution s'il en existe une.

Preuve:
Au cours de l'étape 5.2*, le nombre de conflits diminue d'au moins une unité. Montrons que s'il existe un conflit dans la cellule* (i, j) *et pourtant on passe à l'étape* 5.3 *alors* $m \leq 2c(2c-1)$ *et* $n \leq 2c(2c-1)$*. En effet, on a* $|L_j| \leq v_j^a + v_j^b - 1 \leq 2c - 1$ *car il y a un conflit dans* (i, j)*. Le nombre de cellules colorées sur les lignes* L_j *est inférieur à* $2c(2c-1)$ *car une ligne contient au plus* $2c$ *cellules colorées et* $|L_j| \leq 2c - 1$ *(voir Figure 4.4). On en déduit que* $n \leq 2c(2c-1)$ *car une colonne admet au moins une cellule colorée sur les lignes* L_j*. Autrement dit, il y aura encore une permutation à effectuer à l'étape 5.2. De même, on démontre que* $m \leq 2c(2c-1)$*.*

La taille du sous-problème $T3C$ *à résoudre à l'étape* 5.3 *est donc indépendante de* m *et* n*. Constatons que si* $m > 2c(2c-1)$ *et* $n > 2c(2c-1)$ *alors l'algorithme A-P2 converge sans passer par l'étape* 5.3*.* ■

Dans le cas général : si $m > 2C_{max}(2C_{max} - 1)$ et $n > 2C_{max}(2C_{max} - 1)$ avec $C_{max} = max(h_i^a, h_i^b, v_j^a, v_j^b,\ i = 1, \ldots, m,\ j = 1, \ldots, n)$ alors le problème $T3C$ est polynomial.

4.2.3 $T3C$ avec un facteur de multiplication (P3)

Supposons que dans toute direction, le nombre total de cellules colorées est un multiple du nombre de cellules de couleur a avec le même facteur de multiplication pour les deux directions. Le problème $P3$ trouve son application essentiellement en cristallographie où on connaît souvent la proportion des cristaux. Le problème de décision correspondant, $\mathcal{E} - P3$, est défini de la façon suivante :

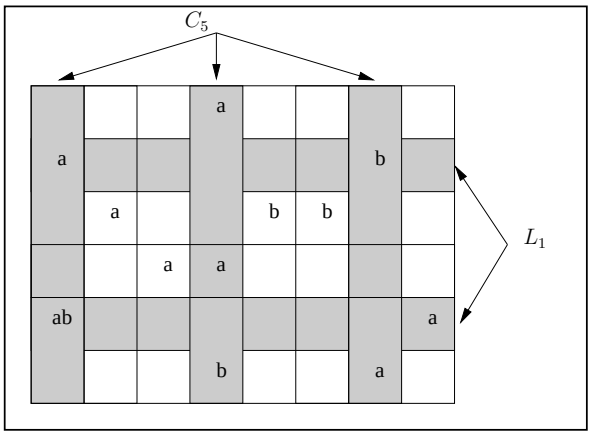


FIG. 4.4 – $T3C$ avec des projections bornées

Données : c un entier, $H \in N^m$ et $V \in N^n$ deux vecteurs à coordonnées entières positives multiples de c.
Question : Existe-t-il un tableau $m \times n$ 3-coloré respectant les projections orthogonales $(\frac{H}{c}, \frac{V}{c})$ pour a et $(H - \frac{H}{c}, V - \frac{V}{c})$ pour b ?

Le cas $c = 2$ correspond à des projections égales pour les deux couleurs. La complexité du problème P3 est donnée par la proposition suivante :

Proposition 26:
Le problème P3 est polynomial.

Preuve:
Nous proposons une preuve constructive permettant de construire une solution au problème P3. Soit A une matrice solution au problème $MB(H, V)$. L'idée de la preuve consiste à extraire de A, une solution Sa respectant les projections de a. Pour cela, associons à A un graphe biparti $G(X, Y, E)$ où $X = \{l_i,\ i = 1, \ldots, m\}$ représente les lignes et $Y = \{c_j,\ j = 1, \ldots, n\}$ représente les colonnes (voir Figure 4.5). Ajoutons à G une source S à laquelle on relie chaque sommet l_i par un arc de capacité h_i. Ajoutons également un puits T auquel on relie chaque sommet c_j par un arc de capacité v_j. Si $A(i, j) = 1$ alors on ajoute un arc de capacité unitaire du sommet l_i au sommet c_j. Le flot maximal dans G est de valeur $\sum_{i=1}^{m} h_i$ car A est une solution au problème $MB(H, V)$.

Pour trouver une solution Sa qui respecte les projections de a, nous considérons le graphe $G'(X, Y, E)$. G' a les mêmes sommets et les mêmes arcs que G sauf que les arcs entrant aux sommets l_i sont de capacité $\frac{h_i}{c}$ et les arcs sortant des sommets c_j sont de capacité $\frac{v_j}{c}$ pour $j = 1, \ldots, n$. Les autres arcs gardent la même capacité. La capacité d'une coupe dans G' est supérieure ou égale à sa capacité dans G divisée par c. La valeur du flot maximal dans G' est $\frac{\sum_{i=1}^{m} h_i}{c}$ parce que tout flot dans G' est de valeur inférieure ou égale à $\frac{\sum_{i=1}^{m} h_i}{c}$. A ce flot maximal entier dans G', nous associons une matrice binaire Sa : $S_a(i, j) = 1$ si le flot maximal dans G' sature l'arc (l_i, c_j). Sa satisfait les projections orthogonales de la couleur a parce que le flot maximal dans G' sature les arcs sortant du

sommet S et les arcs entrant au sommet T.

Pour reconstruire une solution à P3, il suffit de colorer par a les cellules (i, j) vérifiant $A(i, j) = 1$ et $S_a(i, j) = 1$ et colorer par b les cellules (i, j) vérifiant $A(i, j) = 1$ et $S_a(i, j) = 0$. ■

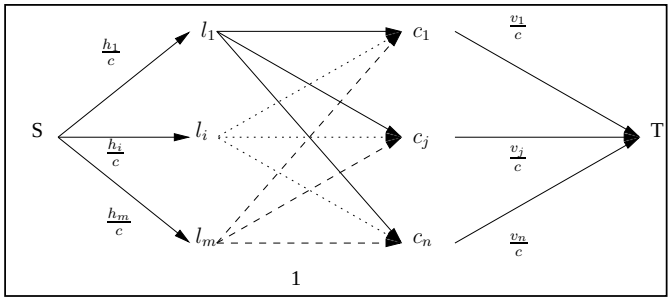


FIG. 4.5 – $T3C$ avec un facteur de multiplication : Graphe G'

Notons que cette preuve ne serait pas valable si les projections orthogonales n'avaient pas le même facteur de multiplication car S_a ne respecterait pas forcément les projections de a.

Détaillons à l'aide d'un exemple la procédure de reconstruction d'une solution au problème $P3$. Prenons $c = 2$, $H = (4, 4, 2, 2)$ et $V = (2, 2, 4, 4)$.

Cherchons, tout d'abord, une solution A à $MB(H, V)$ (voir Tableau 4.2). Ensuite, associons à A le graphe G et en déduisons le graphe G' (voir Figure 4.6). Puis, déterminons un flot maximal dans G' et reconstruisons la matrice Sa. Enfin, combinons Sa et A pour trouver une solution S à $P3$.

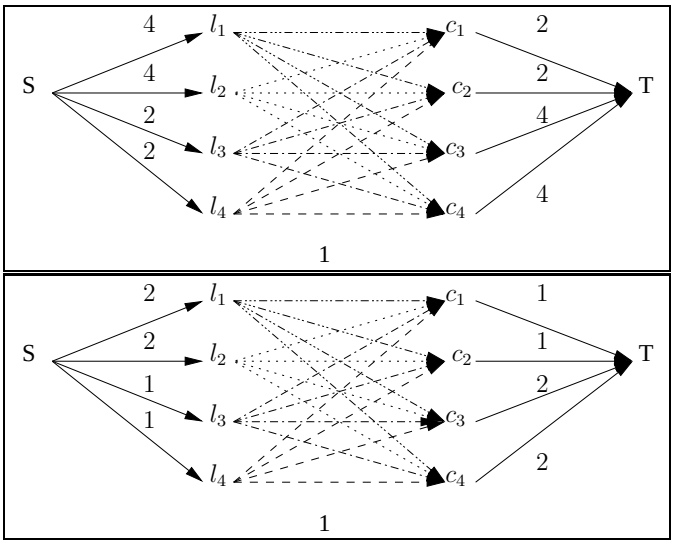


FIG. 4.6 – Exemple de P3 : G en haut et G' en bas

A

1	1	1	1
1	1	1	1
0	0	1	1
0	0	1	1

S_a

0	1	0	1
1	0	1	0
0	0	1	0
0	0	0	1

S

b	a	b	a
a	b	a	b
0	0	a	b
0	0	b	a

TAB. 4.2 – Exemple de P3 (A, S_a et S)

4.2.4 $T3C$ avec une projection relâchée (P4)

Nous relâchons la contrainte de la projection verticale de la couleur b. Nous cherchons une solution satisfaisant les projections (H^a, V^a) pour a et la projection horizontale H^b pour b. Ce problème permet de modéliser certains problèmes d'emplois du temps avec un jour de repos et deux activités a et b. On connaît le nombre d'employés qui exercent chacune des activités par jour. Pour chaque employé, on fixe à l'avance le nombre de jours où il effectue l'activité a.

Proposition 27:
Le problème $P4$ est polynomial.

Preuve:
D'après la proposition 21, il est nécessaire que $h_i^a + h_i^b \leq n,\ i = 1, \ldots, m$, et que le problème $MB(H^a, V^a)$ admette une solution. Ces conditions sont aussi suffisantes : une solution au problème $P4$ est reconstruite en cherchant tout d'abord une solution Sa à $MB(H^a, V^a)$. Ensuite, pour chaque ligne $i,\ i = 1, \ldots, m$, on colore par b (h_i^b) cellules parmi les cellules non colorées par a. ■

4.2.5 $T3C$ avec des cellules colorées adjacentes (P5)

Nous imposons que les cellules colorées ne soient pas isolées, c'est-à-dire, que toute cellule colorée ait au moins une cellule horizontale adjacente colorée. Ce problème est inspiré du problème d'emploi du temps dans lequel les jours de travail ne sont pas isolés, c'est-à-dire, au moins deux jours de travail consécutifs. Le problème de décision correspondant, $\mathcal{E} - P5$, est défini de la façon suivante :

Données : $H^a, H^b \in N^m$ et $V^a, V^b \in N^n$ quatre vecteurs à coordonnées entières positives.
Question : Existe-t-il un tableau $m \times n$ 3-coloré respectant les projections orthogonales (H^a, V^a) pour a et (H^b, V^b) pour b sans cellules colorées isolées (horizontalement) ?

Concernant la complexité du problème $P5$, on peut énoncer le résultat suivant :

Proposition 28:
Le problème $P5$ est au moins aussi difficile que le problème $T3C$ (sans la contrainte d'adjacence des cellules colorées).

Preuve:
Nous démontrons le résultat à l'aide d'une réduction du problème T3C au problème P5. Soit Q une instance du problème T3C respectant les projections (H'^a, V'^a) et (H'^b, V'^b) :

– *h'^a_i (resp. h'^b_i) : le nombre de cellules de couleur a (resp. b) sur la ligne i.*

– *v'^a_j (resp. v'^b_j) : le nombre de cellules de couleur a (resp. b) sur la colonne j.*

Nous définissons une instance P du problème P5 de taille $m \times 5n$ respectant les projections (H^a, V^a) et (H^b, V^b) (voir Figure 4.7) avec :

- $h^a_i = h'^a_i + 2n,\ i = 1, \ldots, m,$
- $h^b_i = h'^b_i + 2n,\ i = 1, \ldots, m,$
- $v^a_{5j-1} = v^a_{5j} = 0,\ v^b_{5j-1} = v^b_{5j} = m,\ j = 1, \ldots, n,$
- $v^a_{5j-4} = v^a_{5j-3} = m,\ v^b_{5j-4} = v^b_{5j-3} = 0,\ j = 1, \ldots, n,$
- $v^a_{5j-2} = v'^a_j,\ v^b_{5j-2} = v'^b_j,\ j = 1, \ldots, n.$

P et Q sont équivalentes car toute solution de P se transforme polynomialement en une solution de Q et inversement. En effet, dans P, les colonnes $5j-4$ et $5j-3$ sont entièrement colorées par a, les colonnes $5j-1$ et $5j$ sont entièrement colorées par b. Les colonnes $5j-2$ de P et j de Q ont les mêmes projections. Ainsi s'achève la preuve étant donnée que P est un cas particulier du problème P5. ■

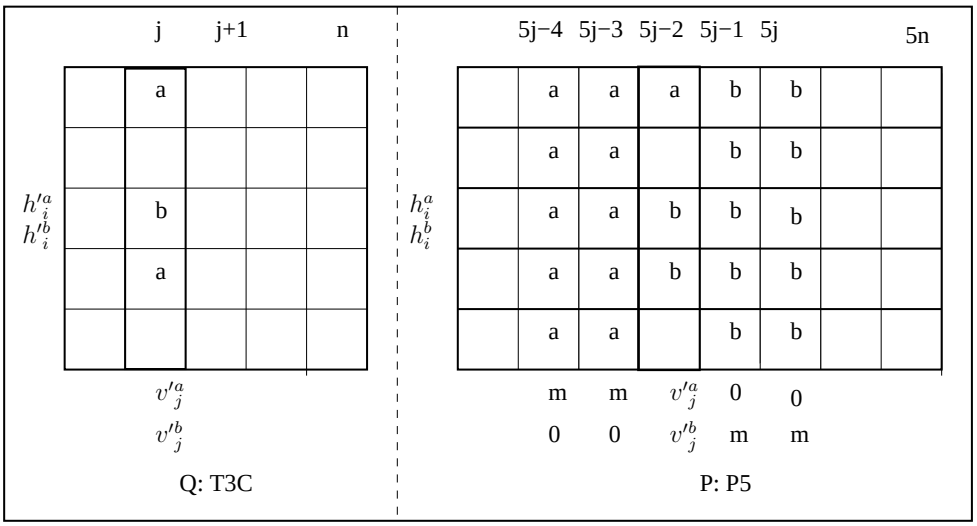


FIG. 4.7 – *T3C* avec des cellules colorées adjacentes

4.2.6 *T3C* avec les produits des projections bornés (P6)

Nous supposons que si dans une direction les projections de a et b sont non nulles alors leur somme est inférieure à trois, c'est-à-dire, $h^a_i h^b_i \leq 2,\ i = 1, \ldots, m$, et $v^a_j v^b_j \leq 2,\ j = 1, \ldots, n$.

Afin de trouver une solution, nous proposons l'algorithme A-P6. Il procède par énumération en temps polynomial sur tous les cas possibles.

Algorithme A-P6

Chercher une solution Sa à $MB(H^a, V^a)$ et une solution Sb à $MB(H^b, V^b)$;
$S \longleftarrow S_a \oplus S_b$;
tant qu'il existe un conflit (i, j) **dans** S **faire**

1. **si** $h_i^a = h_i^b = v_j^a = v_j^b = 1$: chercher une cellule colorée (k, l) avec $k \neq i$ et $l \neq j$ et effectuer la permutation $(i, j), (i, l), (k, j), (k, l)$;
2. **si** $h_i^a = 2$ **et** $h_i^b = v_j^a = v_j^b = 1$: chercher une cellule colorée (k, l) telle que les cellules (k, j) et (i, l) ne sont pas colorées et effectuer la permutation $(i, j), (i, l), (k, j), (k, l)$;
3. **si** $h_i^a = v_j^b = 2$ **et** $h_i^b = v_j^a = 1$: chercher une cellule $(i, l1)$ colorée par a et une cellule $(k1, j)$ colorée par b. S'il existe une cellule colorée $(k2, l2)$ avec $k2 \neq i, k1$ et $l2 \neq j, l1$ alors effectuer la permutation $(i, j), (i, l2), (k, j), (k, l2)$. Sinon :
   (a) **si la cellule** $(k1, l1)$ **est non colorée** :
      i. s'il existe une cellule $(k1, l2)$ colorée par a avec $l2 \neq l1, j$ alors effectuer les permutations $(i, l1), (i, l2), (k1, l1), (k1, l2)$ et $(i, j), (i, l1), (k2, j), (k2, l1)$;
      ii. s'il existe une cellule $(k2, l1)$ colorée par b avec $k2 \neq i, k1$ alors effectuer les permutations $(k1, j), (k1, l1), (k2, j), (k2, l1)$ et $(i, j), (i, l2), (k1, j), (k1, l2)$;
      iii. sinon, le problème n'admet pas de solution ;
   (b) **si la cellule** $(k1, l1)$ **est colorée** alors le problème n'admet pas de solution ;
4. **si** $h_i^a = v_j^a = 2$ **et** $h_i^b = v_j^b = 1$: chercher deux cellules $(i, l1)$ et $(k1, j)$ colorées par a. S'il existe une cellule colorée $(k2, l2)$ avec $k2 \neq i, k1$ et $l2 \neq j, l1$ alors effectuer la permutation $(i, j), (i, l2), (k, j), (k, l2)$. Sinon :
   (a) **si la cellule** $(k1, l1)$ **est non colorée** :
      i. s'il existe une cellule $(k1, l2)$ colorée par a avec $l2 \neq l1, j$ alors effectuer les permutations $(i, l1), (i, l2), (k1, l1), (k1, l2)$ et $(i, j), (i, l1), (k2, j), (k2, l1)$;
      ii. s'il existe une cellule $(k2, l1)$ colorée par a avec $k_2 \neq i, k1$ alors effectuer les permutations $(k1, j), (k1, l1), (k2, j), (k2, l1)$ et $(i, j), (i, l2), (k1, j), (k1, l2)$;
      iii. si les deux cellules $(k1, l2)$ et $(k2, l1)$ sont colorées par b alors effectuer les permutations $(k2, l1), (k2, l2), (k1, l1), (k1, l2)$ et $(i, j), (i, l2), (k2, j), (k2, l2)$;
   (b) **si la cellule** $(k1, l1)$ **est colorée** alors il n'existe pas de solution ;

fin tant que ;

Proposition 29:
Le problème P6 est polynomial et l'algorithme A-P6 en fournit une solution s'il en existe une.

Preuve:
Supposons que toutes les conditions de la proposition 21 sont satisfaites. L'algorithme A-P6 est polynomial car à chaque étape de la boucle tant que le nombre de conflits diminue d'au moins une unité. Montrons qu'il donne toujours une solution s'il en existe une et que les cellules intervenant existent (voir la figure 4.8 pour les différents cas).

- ***Si*** $h_i^a = h_i^b = v_j^a = v_j^b = 1$ *alors il existe une cellule* (k, l) *colorée car d'après la proposition 21, on a* $m \geq v_j^a + v_j^b = 2$ *et* $n \geq h_i^a + h_i^b = 2$.
- ***Si*** $h_i^a = 2$ ***et*** $h_i^b = v_j^a = v_j^b = 1$ *alors il existe une cellule* $(i, l1)$ *colorée par* a *parce que* $h_i^a = 2$. *Il existe également une cellule colorée* (k, l) *avec* $k \neq i$ *et* $l \neq j, l1$ *car* $m \geq v_j^a + v_j^b = 2$ *et* $n \geq h_i^a + h_i^b = 3$.
- ***Si*** $h_i^a = v_j^b = 2$ ***et*** $h_i^b = v_j^a = 1$ *alors il existe une cellule* $(i, l1)$ *colorée par* a *et une cellule* $(k1, j)$ *colorée par* b *car* $h_i^a = 2$ *et* $v_j^b = 2$.
  S'il n'existe pas une cellule colorée $(k2, l2)$ *avec* $k2 \neq i, k1$ *et* $l2 \neq j, l1$ *alors :*
  - *Chaque colonne (resp. ligne) à l'exception de* $l1$ *et* j *(resp.* $k1$ *et* i*) admet une seule cellule colorée placée sur la ligne* $k1$ *(resp. colonne* $l1$*).*
  - *Il existe au moins deux cellules colorées* $(k2, l1)$ *et* $(k1, l2)$ *parce que* $m \geq 3$ *et* $n \geq 3$. *Deux cas se présentent suivant la coloration de la cellule* $(k1, l1)$.

  Si ***la cellule*** $(k1, l1)$ ***est non colorée*** *et il n'existe ni de cellule* $(k1, l2)$ *colorée par* a *avec* $l2 \neq l1, j$ *ni de cellule* $(k2, l1)$ *colorée par* b *avec* $k2 \neq i, k1$ *alors le problème n'admet pas de solution. En effet, sur la ligne* $k1$ *(resp. la colonne* $l1$*), il n'existe pas de cellules colorées par* a *(resp.* b*) et la cellule* (i, j) *est forcement bicolorée.*
  Si ***la cellule*** $(k1, l1)$ ***est colorée*** *alors on a* $h_{k1}^a + h_{k1}^b = n$ *et* $v_{l1}^a + v_{l1}^b = m$. *Le problème P6 n'admet pas de solution car même le problème* $MB(H^a + H^b, V^a + V^b)$ *n'admet pas de solution (voir le cas 3.b de la figure 4.8).*
- ***Si*** $h_i^a = v_j^a = 2$ ***et*** $h_i^b = v_j^b = 1$ *alors il existe deux cellules* $(i, l1)$ *et* $(k1, j)$ *colorées par* a *parce que* $h_i^a = v_j^a = 2$. *S'il n'existe pas de cellule colorée* $(k2, l2)$ *avec* $k2 \neq i, k1$ *et* $l2 \neq j, l1$ *alors il existe au moins deux cellules colorées* $(k2, l1)$ *et* $(k1, l2)$ *car* $m \geq 3$ *et* $n \geq 3$. *Si la cellule* $(k1, l1)$ *est colorée alors il n'existe pas de solution (voir le cas 3.b de la figure 4.8).* ■

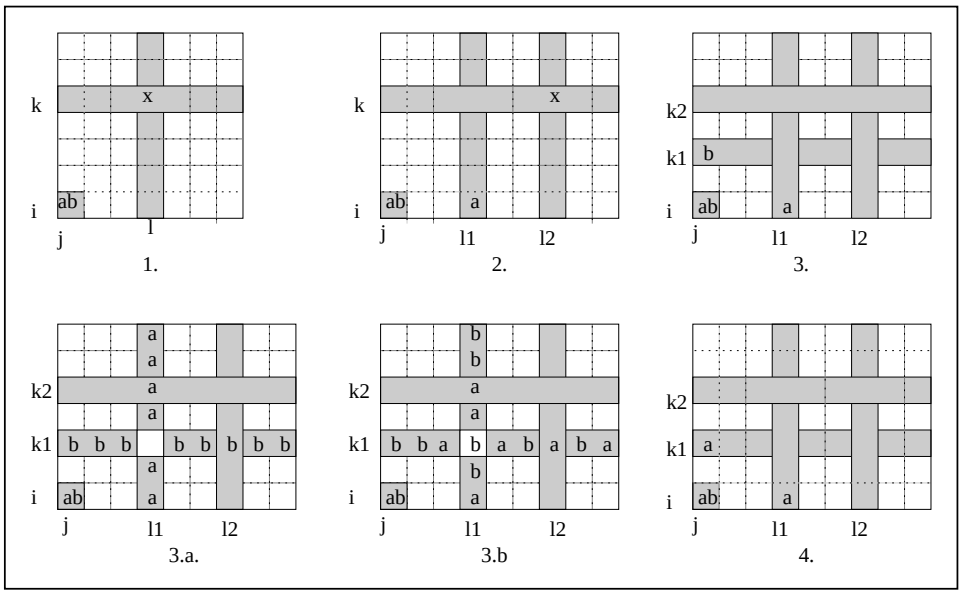


FIG. 4.8 – $T3C$ avec des produits des projections bornés

4.3 Reconstruction d'images numériques

Les problèmes de reconstruction d'images numérique sont présents dans plusieurs domaines tels que l'imagerie médicale (scanner) et les sciences de la terre (océanographie, géophysique, astronomie). Le principe est d'effectuer une section virtuelle de l'objet. L'objet peut être un patient comme en imagerie médicale ou bien un océan comme en océanographie. On enregistre la représentation de l'objet sous plusieurs angle de vue (projections). Ces données sont combinées pour obtenir une coupe de l'objet.

Dans cette partie, nous allons nous intéresser au problème de reconstruction d'images numériques avec niveaux de gris à partir de leurs projections orthogonales et de certaines de leurs propriétés. Une image numérique est un tableau de pixels. Le pixel correspond à un point ou à un petit carré. Chaque pixel a ses propres caractéristiques (couleurs, luminosité, brillance, etc). La reconstruction d'image peut être aussi considérée comme une technique de compression/décompression car au lieu de stocker les valeurs des pixels, on stocke juste les projections orthogonales des niveaux de gris.

Le problème de reconstruction d'images numérique avec k niveaux de gris est défini de la manière suivante : connaissant les projections orthogonales des niveaux de gris et certaines caractéristiques de l'image, on cherche une image qui satisfait ces projections et ces caractéristiques.

4.3.1 Images équivalentes

Une bascule est un ensemble de quatre cellules (i, j), $(i, j+k)$, $(i+h, j)$ et $(i+h, j+k)$ tel que les cellules (i, j) et $(i + h, j + k)$ sont colorées par x et les cellules $(i + h, j)$ et $(i, j+k)$ sont colorées par y, où x et y sont deux niveaux de gris différents. L'opération de bascule consiste à échanger les niveaux de gris x et y pour ces quatre cellules. Nous dirons que deux images sont équivalentes si elles admettent les mêmes projections orthogonales. Si une image I' est obtenue à partir de I par une suite finie d'opérations de bascule alors I et I' sont équivalentes. Nous donnons ci-après (voir les figures 4.9, 4.10, 4.11) différentes images équivalentes et constatons la nette différence visuelle entre elles.

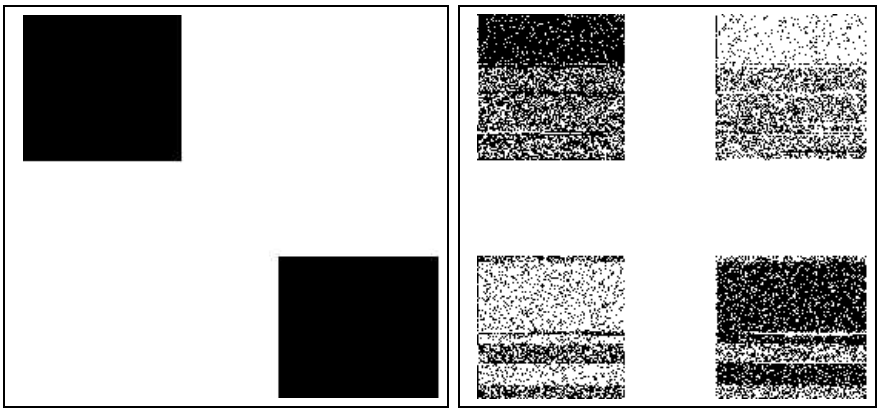


FIG. 4.9 – Images bicolorées équivalentes

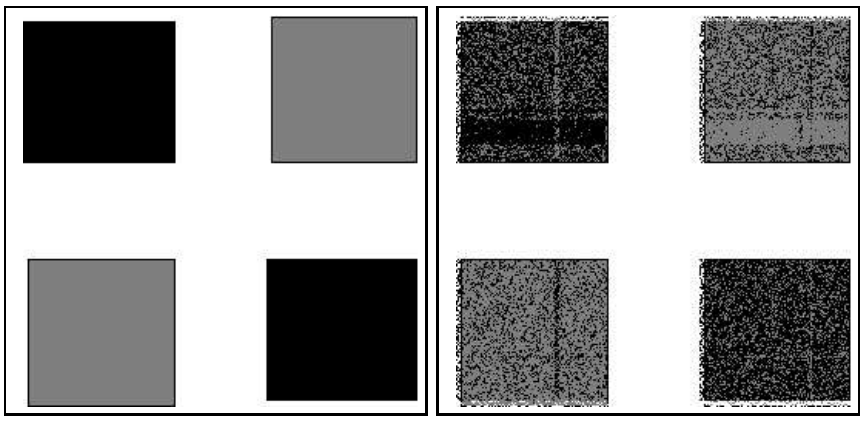


FIG. 4.10 – Images 3-colorées équivalentes

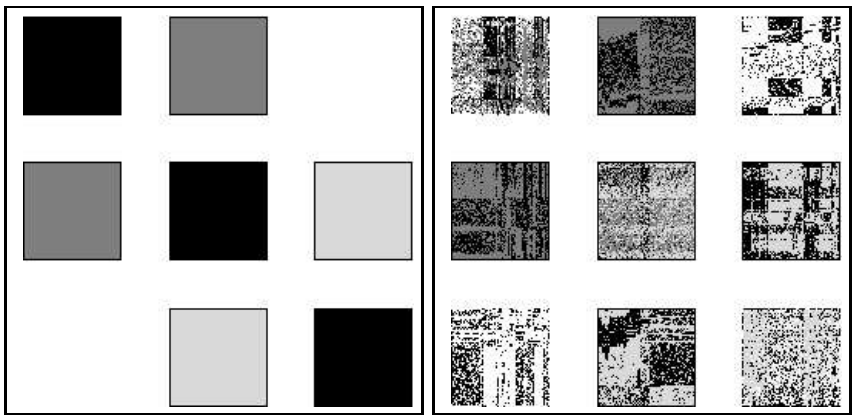


FIG. 4.11 – Images 4-colorées équivalentes

Nous allons nous intéresser au problème consistant à reconstruire à partir des projections orthogonales d'une image I une image équivalente I'. Malgré la dégradation possible, dans certains cas, cela pourra aider à retrouver l'image originale.

4.3.2 Formulation du problème de reconstruction d'une image équivalente

Nous proposons le modèle $\mathcal{MK}$ pour la reconstruction d'images avec niveaux de gris à partir de leurs projections orthogonales. On suppose que les images à reconstruire sont composées de K niveaux de gris : $n_k,\ k = 1, \ldots, K$. Chaque niveau de gris n_k doit respecter les projections orthogonales (H^k, V^k).

Nous introduisons les variables bivalentes : x_{ij}^k pour $i = 1, \ldots, m$, $j = 1, \ldots, n$ et $k = 1, \ldots, K$ définies comme suit :

$$x_{ij}^k = \begin{cases} 1 & \text{si la cellule } (i,j) \text{ est colorée par } n_k \\ 0 & \textit{sinon} \end{cases}$$

$$\mathcal{MK}\ \begin{cases} \sum_{j=1}^{n} x_{ij}^k = h_i^k & i = 1,\dots,m,\ k = 1,\dots,K \quad (1) \\ \sum_{i=1}^{m} x_{ij}^k = v_j^k & j = 1,\dots,n,\ k = 1,\dots,K \quad (2) \\ \sum_{k=1}^{K} x_{ij}^k = 1 & i = 1,\dots,m,\ j = 1,\dots,n \quad (3) \\ x_{ij}^k \in \{0,1\} & i = 1,\dots,m,\ j = 1,\dots,n,\ k = 1,\dots,K \end{cases}$$

La contrainte (1) assure la satisfaction des projections horizontales des niveaux de gris et la contrainte (2) garantit la satisfaction des projections verticales. La contrainte (3) permet d'imposer un niveau de gris unique à chaque cellule.

Le programme linéaire $\mathcal{MK}$ permet aussi de modéliser le problème de reconstruction de tableaux k-colorés à partir de leurs projections orthogonales. Pour le problème $T3C$, le programme $\mathcal{M}3$ est différent des programmes quadratiques $Q1$ et $Q2$ proposés dans la première partie.

Dans les applications telles que l'imagerie médicale, on cherche à obtenir l'image originale avec le maximum de précision. La résolution du programme $\mathcal{MK}$ permet de vérifier l'existence ou non d'images et de donner une image équivalente. Ce qui n'est généralement pas suffisant en pratique. Pour palier cette difficulté, on dispose de deux alternatives. La première consiste à augmenter le nombre de projections, par exemple, en considérant aussi les projections selon les diagonales. L'inconvénient de cette approche est que le problème de reconstruction d'une matrice binaire est NP-complet pour un nombre de projections supérieur ou égal à trois [3]. La deuxième alternative consiste à prendre en compte certaines propriétés connues a priori sur l'image à reconstruire (convexité, connexité, symétrie, périodicité). Pour retrouver l'image originale, il faut intégrer toutes ces caractéristiques.

Notre modèle $\mathcal{MK}$ constitue une partie seulement du problème de reconstruction d'une image. Toutefois, lorsqu'une image est entièrement et parfaitement déterminée par la donnée de ses projections orthogonales (image sans équivalence), la résolution de $\mathcal{MK}$ permet de la reconstruire.

4.3.3 Résolution approchée du programme linéaire $\mathcal{MK}$

Il n'existe pas de méthodes efficaces pour résoudre le programme linéaire en nombres entiers $\mathcal{MK}$ d'autant plus que dans la pratique une image numérique peut contenir plusieurs milliers de pixels voire plusieurs millions. A titre d'exemple, les images des figures 4.9, 4.10 et 4.11 sont de taille de l'ordre de 200×200 pixels. Le nombre de variables pour le programme $M4$ correspondant à l'image représentée par la figure 4.11 est environ 160000 et le nombre de contraintes est de l'ordre de 41600. Dans ce cas, une résolution approchée est indispensable.

Nous proposons de trouver une solution entière par un arrondi particulier de la solution continue du modèle $\mathcal{MK}$. A partir d'une solution (x_{ij}^k) continue de $\mathcal{MK}$, on détermine la valeur du niveau de gris de chaque pixel : $v(i,j) = \lceil \sum_{k=1}^{K} v(n_k) x_{ij}^k \rceil$ où v_k est la valeur du niveau de gris $n_k,\ k = 1,\dots,K$. Par exemple si $K = 4$ et si les quatre niveaux de gris sont le blanc, le gris à 50% , le gris à 70% et le noir alors on pose $v(1) = 0$, $v(2) = 5$, $v(3) = 7$ et $v(4) = 10$. Pour certains cas où la relaxation continue est fractionnaire, l'image reconstruite peut admettre plus de K niveaux de gris.

Résolution approchée de $\mathcal{MK}$

1. Résoudre la relaxation continue de $\mathcal{MK}$; soit (x_{ij}^k) la solution obtenue ;
2. $v(i,j) = \lceil \sum_{k=1}^{K} v(n_k) x_{ij}^k \rceil$ pour tout i et j .

Notons que cette approche de résolution est propre aux images numériques avec niveaux de gris car la somme pondérée de deux niveaux de gris est également un niveau de gris. Pour des images en couleur, cette approche n'est pas valide car les couleurs des cellules ne sont pas additives.

Pour valider notre approche, nous avons testé notre heuristique sur des images qui n'admettent pas d'autres images équivalentes. En effet, dans ce cas, reconstruire une image équivalente à une image donnée I revient à reconstruire I. Alors on peut aisément visualiser la dégradation éventuelle de l'image fournie après l'application de notre approche de résolution. Pour les trois images de la figure 4.12, nous avons constaté que l'image restituée après notre traitement était identique à l'image initiale. Nous avons utilisé le solveur Cplex pour effectuer la première étape de notre algorithme.

On pourrait déduire de l'expérimentation effectuée sur les trois images de la figure 4.12 que notre approche est justifiée pour les images sans images équivalentes, mais de nombreux tests supplémentaires sont nécessaires pour la validation de notre heuristique de résolution.

Nous avons remarqué que pour les images que nous avons testées (sans images équivalentes), la relaxation continue de $\mathcal{MK}$ fournit une solution entière. Il serait intéressant de montrer théoriquement ce résultat.

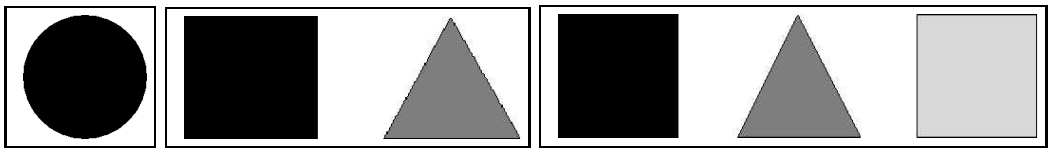


FIG. 4.12 – Images sans images équivalentes

4.4 Conclusion

Dans ce chapitre, nous avons proposé deux programmes mathématiques en nombre entiers pour modéliser le problème $T3C$. Nous avons montré qu'une solution continue du premier programme quadratique se transforme polynomialement en une solution de $T3C$. Cependant, cette démarche ne permet pas de résoudre $T3C$ en temps polynomial car le programme mathématique a une fonction objectif non convexe. C'est pourquoi nous avons étudié plusieurs cas particuliers que nous avons montré polynomiaux en utilisant des modèles de flot.

Néanmoins, la modélisation proposée et la diversité des cas polynomiaux traités ne nous permettent pas de répondre à la question principale liée à la NP-complétude de $T3C$. C'est pourquoi, nous avons conçu une approche de résolution approchée pour reconstruire

des images numériques avec niveaux de gris. Cette approche présente l'avantage d'être facile à implémenter, de plus elle peut s'intégrer facilement dans des algorithmes prenant en compte des contraintes supplémentaires.

Bibliographie

[1] M. Chrobak, P. Couperus, C. Dürr, and G. Woeginger. A note on tiling under tomographic constraints. *Theoretical Computer Science*, 290 :2125–2136, 2003.

[2] M. Chrobak and C. Dürr. Reconstructing polyatomic structures from x-rays : Np completness proof for three atoms. *Theoretical computer science*, 259(1) :81–98, 2001.

[3] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computionnal complexity of reconstructing lattice sets from their x-rays. *Discrete Mathematics*, 202 :45–71, 1999.

[4] H.J. Ryser. Combinatorial properties of matrices of zeros and ones. *Canad. J. Math*, 9 :371–377, 1957.

Chapitre 5

Pavage et packing de dominos

Les problèmes de pavage par des dominos sont de complexité inconnue dans le cas général (voir section 1.4). Toutefois, sous l'hypothèse d'une unique projection ou de projections monotones, ces problèmes sont souvent polynomiaux (voir [2] et [1]). Nous considérons ici des projections orthogonales quelconques et supposons que les dominos sont horizontaux. Nous montrons que les problèmes de pavage et de packing correspondant sont aussi souvent polynomiaux. Nous concluons ce chapitre par l'étude des dominos bicolorés orientés.

5.1 Description du problème

Un domino est un rectangle de taille 2×1 ou 1×2. Un domino horizontal (resp. vertical) **commence** à la colonne (resp. ligne) où se trouve sa cellule la plus à gauche (resp. en haut) (voir section 1.1).

Le problème de **pavage** par des dominos consiste à paver un tableau par des dominos respectant les projections (H, V). Le problème de **packing** de dominos consiste à placer sans recouvrement des dominos dans un tableau. Les projections donnent le nombre de cellules noires dans chacune des directions. La section 1.5 illustre différentes applications de ces divers problèmes. Nous remarquons que dans un problème de pavage par des dominos horizontaux, le nombre de colonnes du tableau est pair et tous les dominos commencent sur des colonnes impaires.

Nous rappelons que le problème $MB(H, V)$ [3] consiste à reconstruire une matrice binaire à partir de ses projections orthogonales (H, V).

5.2 Pavage par des dominos unicolorés horizontaux ($PavDUH(H, V)$)

Un domino unicoloré a ses deux cellules soit noires, soit blanches. Le problème *PavDUH(H,V)* consiste à paver un tableau par des dominos unicolorés placés horizontalement en respectant les projections (H, V). La figure 5.1 illustre ce problème. Le problème de décision correspondant, $\mathcal{E} - PavDUH(H, V)$, est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ et $V = (v_1, \ldots, v_n)$ deux vecteurs à coordonnées entières positives.
Question : Existe-t-il un pavage par des dominos unicolorés horizontaux respectant les projections (H, V) ?

Nous donnons les conditions nécessaires et suffisantes d'existence de solution :

Proposition 30:
$\mathcal{E} - PavDUH(H, V)$ a pour réponse 'oui' si et seulement si :
i) $h_i \leq \frac{n}{2},\ i = 1, \ldots, m$ *et* n *est pair,*
ii) $v_{2j-1} = v_{2j},\ j = 1, \ldots, \frac{n}{2}$,
iii) $MB(H', V')$ *admet une solution où* $h'_i = \frac{h_i}{2},\ i = 1, \ldots, m$, *et* $v'_j = v_{2j},\ j = 1, \ldots, \frac{n}{2}$.

Preuve:
Il suffit de montrer que le problème PavDUH(H,V) est équivalent au problème $MB(H', V')$ *avec* $h'_i = \frac{h_i}{2},\ i = 1, \ldots, m$, *et* $v'_j = v_{2j},\ j = 1, \ldots, \frac{n}{2}$ *(voir Figure 5.1).* ■

Notons que le problème de packing de dominos unicolorés est polynomial [2] car il est aussi un problème de packing de dominos horizontaux (voir section 1.4). En effet, les dominos noirs sont remplacés par des dominos non colorés et les dominos blancs sont remplacés tout simplement par des cellules non couvertes. Cette transformation n'est pas valide pour le problème $PavDUH(H, V)$ car les dominos noirs sont séparés par un nombre pair de cellules (dominos blancs).

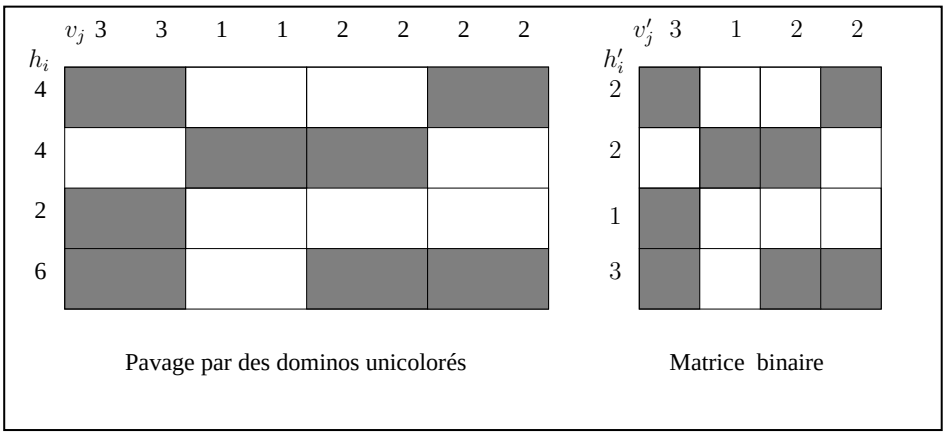


FIG. 5.1 – $PavDUH(H,V)$ et $MB(H',V')$

5.3 Packing de dominos bicolorés horizontaux (*PacDBH(H,V)*)

Un domino bicoloré a une cellule noire et une cellule blanche. Dans le problème *PacDBH(H,V)*, on cherche un packing de dominos bicolorés horizontaux satisfaisant les projections orthogonales (H,V) (voir figure 5.2). Le problème de décision correspondant, $\mathcal{E}-PacDBH(H,V)$, est défini de la façon suivante :

Données : $H=(h_1,\ldots,h_m)$ et $V=(v_1,\ldots,v_n)$ deux vecteurs à coordonnées entières positives.
Question : Existe-t-il un packing de dominos bicolorés horizontaux respectant les projections H et V ?

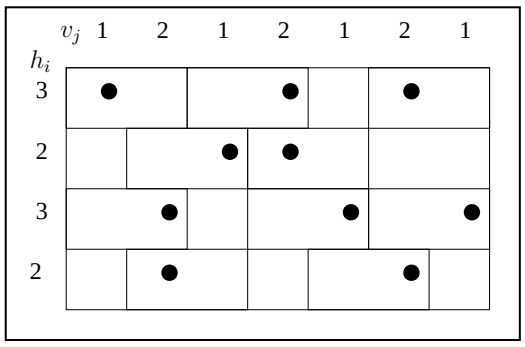


FIG. 5.2 – Packing de dominos bicolorés horizontaux

Il est évident que $h_i \leq \frac{n}{2},\ i=1,\ldots,m$, est une condition nécessaire d'existence de solution au problème *PacDBH(H,V)* car sur chaque ligne i, il y a h_i cellules noires et h_i dominos. Nous supposons dans la suite que cette condition est satisfaite et nous donnons une condition suffisante d'existence d'une solution :

Proposition 31:
Une condition suffisante pour que le problème PacDBH(H,V) admette une solution est que le problème $MB(H,V)$ avec contrainte de 2-adjacence horizontale admet une solution.

Preuve:
Le problème $MB(H,V)$ avec contrainte de 2-adjacence horizontale consiste à reconstruire une matrice binaire sans que deux cellules de valeur 1 soient adjacentes horizontalement (voir section 2.1.2). Si $h_i \leq \frac{n}{2}$ pour $i = 1, \ldots, m$ alors une solution à PacDBH(H,V) est construite par l'algorithme A-PacDBH(H,V). Soit $S1$ une solution au problème $MB(H,V)$ avec contrainte de 2-adjacence horizontale. Le principe de l'algorithme consiste à coupler chaque cellule de valeur 1 avec la cellule adjacente de droite ou de gauche pour former un domino bicoloré. Ce couplage existe car $2h_i \leq n$ pour $i = 1, \ldots, m$. ■

Algorithme A-PacDBH(H,V)

1. *Trouver une solution, $S1$, à $MB(H,V)$ avec contrainte de 2-adjacence horizontale ;*
2. pour $i = 1$ ***à*** m ***faire***
2.1 si S1(i,1)=0 *alors coupler chaque cellule de valeur 1 sur la ligne i avec la cellule adjacente de gauche ;*
2.2 si S1(i,1)=1 et S1(i,n)=0 *alors coupler chaque cellule de valeur 1 sur la ligne i avec la cellule adjacente de droite ;*
2.3 si S1(i,1)= S1(i,n)=1 *alors chercher j tel que $S1(i,j) = 1$ et $S1(i,j+1) = S1(i,j+2) = 0$ (j existe car $2h_i \leq n$) ;*
coupler toute cellule de valeur 1 avant j avec la cellule adjacente de droite ;
coupler toute cellule de valeur 1 après j avec la cellule adjacente de gauche ;
fin si ;
fin faire ;

Pour résumer, si le problème $MB(H,V)$ avec 2-adjacence horizontale admet une solution, on aura résolu le problème *PacDBH(H,V)* à l'aide de l'algorithme A-PacDBH(H,V). Autrement dit, il nous sera impossible de conclure. La difficulté provient du choix des lignes sur lesquelles les dominos commencent et de la position de la cellule blanche par rapport à la noire.

5.4 Pavage par des dominos bicolorés horizontaux (*PavDBH(H,V)*)

Dans cette section, on cherche à reconstruire un pavage par des dominos bicolorés horizontaux satisfaisant les projections orthogonales, comme nous pouvons le voir sur la figure 5.3. Les conditions nécessaires et suffisantes suivantes sont évidentes pour l'existence d'une solution.

Proposition 32:
Le problème PavDBH(H,V) admet une solution si et seulement si :
i) $h_i = \frac{n}{2}$ pour $i = 1, \ldots, m$ et n est pair,
ii) $v_{2j-1} + v_{2j} = m$ pour $j = 1, \ldots, \frac{n}{2}$.

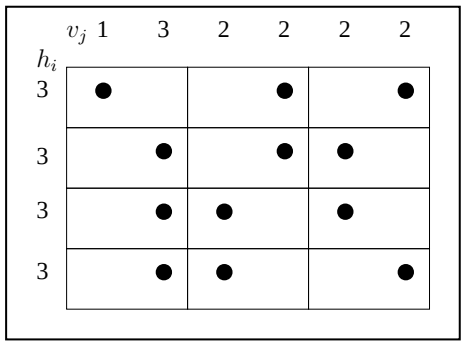


FIG. 5.3 – Pavage par des dominos bicolorés horizontaux

5.5 Pavage par des dominos bicolorés orientés (*PavDBO(H,V)*)

Dans cette section, nous revenons au problème de pavage de dominos bicolorés pouvant être placés horizontalement ou verticalement. Les dominos sont orientés de la façon suivante : les dominos verticaux ont leur cellule noire située au dessus de leur cellule blanche et les dominos horizontaux ont leur cellule noire située à gauche de leur cellule blanche. En premier lieu, nous donnons un résultat concernant le sous-problème *PavDBO(H)* où seule la projection horizontale est à respecter. En deuxième lieu, nous établissons l'équivalence polynomiale entre le problème dans toute sa généralité et le problème *PavD(H,V)* de pavage d'un tableau par des dominos. Les projections du problème *PavD(H,V)* donnent le nombre de dominos qui coupent chaque ligne et chaque colonne.

Proposition 33:
Le problème PavDBO(H) est équivalent au problème PavD(H') avec
$h'_i = 2\sum_{j=1}^{i-1} h_j + h_i - (i-1)n,\ i = 1, \ldots, m.$

Preuve:
Soit P une instance de PavDBO(H). Notons par h'_i le nombre de dominos qui coupent la ligne i. On déduit de h_i et h'_i que :
- *$h'_i - h_i$ est le nombre de dominos verticaux qui commencent à la ligne $i-1$.*
- *$2\sum_{j=1}^{i-1} h_j$ est le nombre de cellules couvertes par un domino commençant avant la ligne $i-1$.*
- *$(i-1)n$ est le nombre de cellules à couvrir entre les lignes 1 et $i-1$.*

D'où $(i-1)n = 2\sum_{j=1}^{i-1} h_j - (h'_i - h_i)$.
On peut alors exprimer les h'_i en fonction des h_i :
$h_1 = h'_1$ *et* $h'_i = 2\sum_{j=1}^{i-1} h_j + h_i - (i-1)n,\ i = 2, \ldots, m.$
Inversement, à partir des h'_i, il est facile de déduire les h_i :
$h_1 = h'_1$ *et* $h_{i+1} = h'_{i+1} - h'_i - h_i + n,\ i = 1, \ldots, m-1.$
Ainsi PavDBO(H) et PavD(H) sont équivalents. ■

De la même manière, *PavDBO(V)* et *PavD(V)* sont équivalents et nous énonçons le résultat suivant :

Proposition 34:
Les problèmes PavDBO(H,V) et PavD(H,V) sont équivalents.

5.6 Conclusion

Nous avons considéré dans ce chapitre certains problèmes de pavage et de packing de dominos colorés. Nous avons envisagé différentes coloration des dominos. Nous avons montré que le problème *PavDBO(H,V)* est équivalent au problème de pavage par des dominos. Ainsi la coloration et l'orientation simultanées des dominos ne changent pas la complexité du problème de pavage. Nous avons dégagé une condition suffisante d'existence de solution au problème $PacDBH(H,V)$ et avons proposé un algorithme polynomial qui construit une solution lorsque la condition est satisfaite. Les autres problèmes traités sont polynomiaux et les solutions sont reconstruites à l'aide des algorithmes gloutons. Ce travail peut être étendu en remplaçant les dominos colorés par des triminos colorés. Différentes colorations sont envisageables, par exemple, la cellule au milieu du trimino est noire et les deux autres cellules sont blanches ou bien les trois cellules ont des couleurs différentes.

Bibliographie

[1] C. Picouleau. Reconstruction of a coloured domino tiling from its projections. Technical report, Cedric-Cnam, 2001.

[2] C. Picouleau. Reconstruction of domino tiling from its two orthogonal projections. *Theoretical computer science*, 255(1) :437–447, 2001.

[3] H.J. Ryser. Combinatorial properties of matrices of zeros and ones. *Canad. J. Math*, 9 :371–377, 1957.

Chapitre 6

Packing de barres

Nous abordons dans ce chapitre le problème général de pavage et packing de barres. Le problème est de placer des barres dans un tableau en respectant les projections orthogonales. Nous distinguons les barres de longueur fixée (b-barres) et les barres de longueur arbitraire. Nous commençons par étudier le problème de packing de b-barres horizontales et nous introduisons la notion d'écart entre les barres. Puis, nous considérons le problème de packing de barres de longueurs supérieures ou égales à deux sous différentes contraintes d'écarts.

6.1 Description du problème

Les problèmes de packing de barres consistent à placer sans recouvrement des barres dans un tableau $m \times n$ en respectant les projections orthogonales sous différentes contraintes. Les projections sont le nombre de cellules couvertes par des barres dans chaque direction. Par exemple, les contraintes peuvent être un écart entre deux barres successives. Leur importance tient au fait qu'ils modélisent plusieurs situations de planification de personnel : une barre horizontale de longueur b représente une activité s'étendant sur b périodes de temps et un écart minimal entre les barres constitue un nombre minimal de périodes de repos entre deux activités.

Nous définissons, ci-dessous, le vocabulaire associé aux différents problèmes traités :
Une **barre** horizontale (resp. verticale) est un ensemble de cellules adjacentes d'une même ligne (resp. colonne).
Une **b-barre** est une barre de b cellules.
Une barre horizontale (resp. verticale) **commence** à la colonne (resp. ligne) où se trouve sa cellule la plus à gauche (resp. en haut).
Une barre horizontale (resp. verticale) **se termine** à la colonne (resp. ligne) où se trouve sa cellule la plus à droite (resp. en bas).
Un **domino** est une 2-barre.
Un **triomino** est une 3-barre.
L'écart est le nombre de cellules non couvertes du tableau entre deux barres consécutives d'une rangée.
L'écart minimal est le plus petit écart autorisé entre les barres consécutives (on ne considère pas les écarts avant la première barre et après la dernière barre de chaque ligne).
L'écart maximal est le plus grand écart autorisé entre deux barres consécutives, avant la première barre et après la dernière barre d'une rangée.
Remarque : Pour les problèmes de packing de b-barres horizontales, la projection horizontale de toute ligne est un multiple de b (longueur des b-barres).

6.2 Packing de b-barres horizontales

Dans cette section, nous étudions deux problèmes de packing de b-barres horizontales : packing avec un écart minimal et packing avec un écart maximal. Pour les problèmes de packing de b-barres, nous calculons d_j le nombre de b-barres horizontales qui commencent à la colonne j, $j = 1, \ldots, n$.

Propriété 3:
Dans un problème de packing de b-barres horizontales, d_j le nombre de b-barres qui commencent à la colonne j est :

$d_j = v_j - \sum_{k<j} d_k,\ j = 1, \ldots, b,$
$d_j = v_j - \sum_{k=j-1}^{j+1-b} d_k,\ j = b, \ldots, n-b+1,$
$d_j = 0,\ j = n-b+2, \ldots, n.$

d_j apparaît donc simplement comme la différence entre la projection de la colonne j et le nombre de b-barres commençant aux colonnes $j+1-b, \ldots, j-1$.

On constate que les valeurs $d_j,\ j = 1, \ldots, n$, sont déterminées de manière unique et on en déduit la proposition suivante :

Proposition 35:
Une condition nécessaire pour qu'un packing de b-barres horizontales respecte la projection verticale V est $d_j \geq 0,\ j = 1, \ldots, n$.

Nous supposons dans la suite de ce chapitre que cette condition est vérifiée pour tous les problèmes où la projection verticale est prise en compte.

6.2.1 Packing de b-barres horizontales avec un écart minimal (*PacBHMin(H,V)*)

Dans le problème *PacBHMin(H,V)*, les packings de b-barres horizontales respectent la contrainte suivante : pour toute ligne, deux b-barres consécutives sont espacées d'au moins a cellules ; $a \geq 0$ étant un paramètre du problème (voir Figure 6.1). Le problème de décision correspondant, $\mathcal{E}-$ *PacBHMin(H,V)*, est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ et $V = (v_1, \ldots, v_n)$ deux vecteurs à coordonnées entières positives et $a, b \leq n$ deux entiers positifs.
Question : Existe-t-il un packing de b-barres horizontales respectant les projections (H, V) tel que l'écart minimal entre deux b-barres est supérieur ou égal à a ?

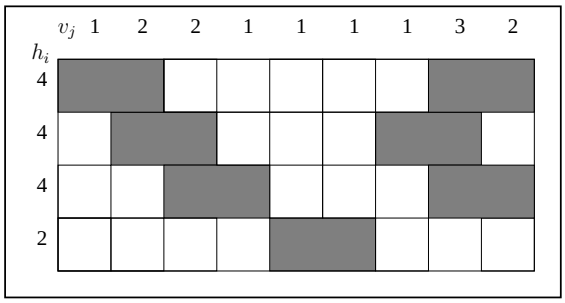


FIG. 6.1 – Packing de 2-barres horizontales avec un écart minimal égal à 3

Nous établissons une condition nécessaire d'existence d'une solution :

Proposition 36:
Une condition nécessaire pour que le problème $\mathcal{E}-$ PacBHMin(H,V) ait pour réponse 'oui' est : $h_i + (\frac{h_i}{b} - 1)a \leq n,\ i = 1, \ldots, m$.

Preuve:
Le résultat est immédiat car sur toute ligne i, il y a h_i cellules couvertes par $\frac{h_i}{b}$ b-barres. Deux b-barres consécutives sont séparées par au moins a cellules. Il y a donc au moins $(\frac{h_i}{b} - 1)a$ cellules non couvertes. ■

Nous supposons dans la suite que cette condition est satisfaite.

Nous proposons l'algorithme polynomial A-PacBHMin(H,V) pour résoudre les problèmes *PacBHMin(H,V)* et $\mathcal{E}-$ *PacBHMin(H,V)*. Le principe de l'algorithme est le suivant : à chaque étape j, $j = 1, \ldots, n-b+1$, d_j b-barres commencent sur les d_j lignes disponibles les plus prioritaires. Si le nombre de lignes disponibles est inférieur à d_j alors l'algorithme s'arrête et il le problème n'admet pas de solution. La ligne i est dite disponible à l'étape j lorsque la dernière cellule recouverte de la ligne i se trouve sur une colonne j' avec $j' < j-a$. α_i étant le nombre de b-barres horizontales non encore placées sur la ligne i à l'étape j de l'algorithme. Les lignes les plus prioritaires sont celles qui ont les plus grandes valeurs α_i.

Algorithme A-PacBHMin(H,V)

1. $\alpha_i = \frac{h_i}{b}$, $i = 1, \ldots, m$;
2. $disp(i,j) = vrai$, $i = 1, \ldots, m$, $j = 1, \ldots, n$;
3. pour $j = 1$ **à** $n-b+1$ **faire**
  3.1 s'il n'existe pas d_j lignes disponibles alors fin ;
  3.2 placer d_j b-barres sur les d_j lignes disponibles les plus prioritaires ;
  3.3 si une b-barre est placée sur la ligne i alors
    $\alpha_i \leftarrow \alpha_i - 1$ et $disp(i,j) = \cdots = disp(i, j+a+b-1) = faux$;
fin faire ;

Le résultat suivant établit la validité de l'algorithme A-PacBHMin(H,V) :

Proposition 37:
$\mathcal{E}-$ PacBHMin(H,V) est polynomial et l'algorithme A-PacBHMin(H,V) fournit une solution au problème PacBHMin(H,V) s'il en existe une.

Preuve:
L'algorithme A-PacBHMin(H,V) est clairement polynomial.

Supposons que le problème PacBHMin(H,V) admet une solution S. Notons $S_{(1..j)}$ la sous-solution de S correspondant aux b-barres qui commencent sur les colonnes $1, \ldots, j$. D'une manière analogue, nous notons $\hat{S}_{(1..j)}$ la solution partielle obtenue par l'algorithme A-PacBHMin(H,V) après l'étape j, c'est-à-dire, le placement des b-barres qui commencent aux colonnes $1, \ldots, j$. Nous montrons, par récurrence sur j, qu'il existe une solution S du problème telle que $S_{(1..j)} = \hat{S}_{(1..j)}$ pour $j = 1, \ldots, n$.

1. ***Initialisation de la récurrence***

   Suposons que $S_{(1)} \neq \hat{S}_{(1)}$. Il existe une ligne l sur laquelle commence une b-barre dans $\hat{S}_{(1)}$ et sur laquelle ne commence pas une b-barre dans $S_{(1)}$. Il existe aussi une ligne k sur laquelle commence une b-barre dans $S_{(1)}$ et sur laquelle ne commence pas une b-barre dans $\hat{S}_{(1)}$. L'algorithme A-PacBHMin(H,V) impose que $\alpha_k \leq \alpha_l$. Dans S numérotons de 1 à α_l (resp. de 1 à α_k) les b-barres de la ligne l (resp. k). Nous notons C_l^i (resp. C_k^i) la colonne à laquelle commence la $i^{ème}$ b-barre de la ligne l (resp. k) dans S. Soit r tel que $\forall i < r$, $C_k^i < C_l^i$ et $C_k^r \geq C_l^r$, c'est-à-dire, r est le

plus petit indice tel que la $r^{ème}$ b-barre de la ligne l commence avant celle de la ligne k dans S (voir Figure 6.2). r existe parce que $\alpha_k \leq \alpha_l$.

Nous montrons qu'en permutant les $r-1$ premières b-barres des lignes k et l, on obtient une solution équivalente où une b-barre commence sur la ligne l à la colonne 1. Cette permutation respecte la contrainte d'écart minimal entre la $(r-1)^{ème}$ b-barre et la $r^{ème}$. En effet, d'une part $C_l^r \geq C_l^{r-1}+b+a$ car S est une solution vérifiant les contraintes du problème, et d'autre part, d'après la définition de r, $C_l^r \leq C_k^r$ et $C_l^{r-1} > C_k^{r-1}$, et ainsi $C_l^r \geq C_k^{r-1}+b+a$ et $C_k^r \geq C_l^{r-1}+b+a$. Il est facile de voir que cette permutation ne modifie pas les projections orthogonales. En réitérant ce type de permutation, nous obtenons une solution S telle que $S_{(1)} = \hat{S}_{(1)}$.

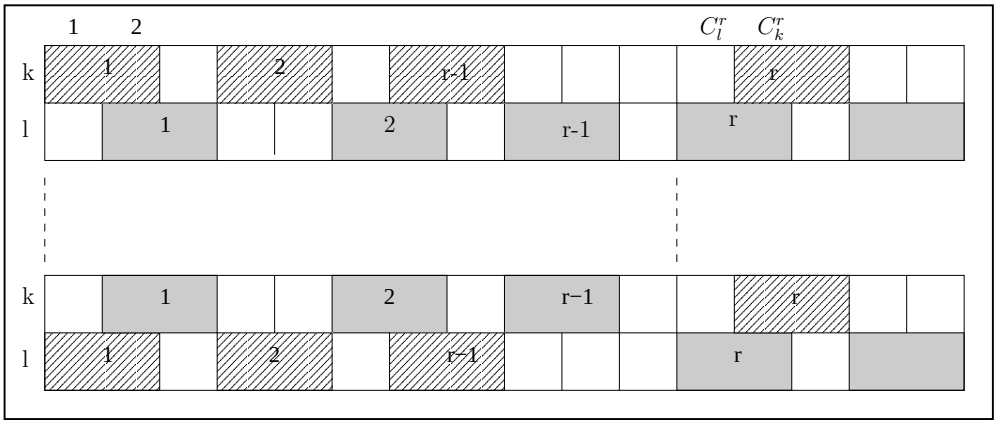


FIG. 6.2 – Permutation des b-barres $1, \ldots, r-1$ sur les lignes k et l à l'étape 1

2. ***Supposons qu'il existe une solution S telle que*** $S_{(1..j)} = \hat{S}_{(1..j)}$ *et* $S_{(1..j+1)} \neq \hat{S}_{(1..j+1)}$. *Montrons qu'il existe une solution équivalente S telle que $S_{(1..j+1)} = \hat{S}_{(1..j+1)}$. Le principe de la preuve est identique à celui utilisé pour $j = 1$. A l'étape $j+1$ de A-PacBHMin(H,V), il y a au moins d_{j+1} lignes disponibles puisque S est une solution. Il existe une ligne l (resp. k) sur laquelle commence une b-barre à la colonne $j+1$ dans $\hat{S}$ (resp. S), et sur laquelle ne commence pas une b-barre à la colonne $j+1$ dans S (resp. $\hat{S}$). Les b-barres qui commencent à partir de la colonne $j+1$ sont numérotées de 1 à α_l (resp. α_k) sur la ligne l (resp. k). De la même façon que pour $j = 1$, nous définissons C_l^i, C_k^i et r. Les b-barres de ces deux lignes, numérotées de 1 à $r-1$, placées entre les colonnes $j+1$ et C_l^{r-1} sont permutées de la même manière que pour $j = 1$ (voir Figure 6.3). Après cette permutation, la contrainte d'écart minimal est respectée et les projections orthogonales ne sont pas modifiées. On réitère ce type de permutation tant qu'il existe deux lignes l et k répondant au critère défini ci-dessus, jusqu'à obtenir la solution S cherchée.*

Ainsi l'algorithme A-PacBHMin(H,V) permet de répondre en temps polynomial au problème $\mathcal{E}-$ PacBHMin(H,V). ■

On déduit également que le problème de packing de b-barres horizontales respectant les projections orthogonales est polynomial. Ceci constitue une généralisation du résultat de Picouleau [3] concernant la polynomialité du problème de packing de dominos horizontaux.

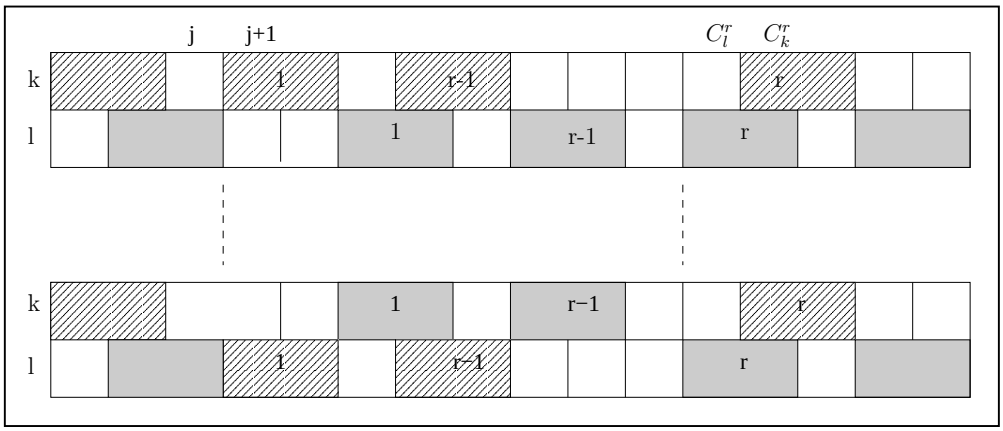


FIG. 6.3 – Permutation des b-barres $1, \ldots, r-1$ sur les lignes k et l à l'étape $j+1$

Dans la section suivante, nous imposons un écart maximal à la place de l'écart minimal.

6.2.2 Packing de b-barres horizontales avec un écart maximal satisfaisant la projection verticale (*PacBHMax(V)*)

Dans le problème *PacBHMax(V)*, on cherche un packing de b-barres horizontales qui respecte la projection verticale V et qui minimise l'écart maximal. Le problème de décision correspondant, $\mathcal{E}-$ *PacBHMax(V)*, est défini de la façon suivante :

Données : $V = (v_1, \ldots, v_n)$ un vecteur à coordonnées entières positives et a, b deux entiers positifs.
Question : Existe-t-il un packing de b-barres horizontales respectant la projection verticale V tel que l'écart maximal est inférieur à a ?

Pour reconstruire un packing avec un écart maximal minimum, nous proposons l'algorithme itératif polynomial A-PacBHMax(V). Le principe de cet algorithme consiste à placer à chaque étape j, d_j b-barres à la colonne j et sur les lignes disponibles les plus prioritaires. La ligne i est dite disponible à l'étape j lorsque la dernière b-barre placée sur la ligne i se termine avant la colonne j. La ligne k est plus prioritaire que la ligne l si la dernière b-barre placée sur la ligne k se termine avant la dernière placée sur la ligne l. On note la similitude avec l'algorithme A-PacBHMin(H,V).

Algorithme A-PacBHMax(V)

1. $disp(i,j) = vrai,\ i = 1, \ldots, m,\ j = 1, \ldots, n$;
2. pour $j = 1$ **à** $n - b + 1$ **faire**
  2.1 s'il n'existe pas d_j lignes disponibles alors fin ;
  2.2 placer d_j b-barres sur les d_j lignes disponibles les plus prioritaires ;
  2.3 si une b-barre commence sur la ligne i alors
    $disp(i,j) = \cdots = disp(i, j+b-1) = faux$;
fin faire ;

Nous énonçons le résultat suivant :

Proposition 38:
Si $h_i+(\frac{h_i}{b}-1)a \leq n,\ i=1,\ldots,m$ alors l'algorithme A-PacBHMax(V) donne une solution avec l' écart maximal minimum pour le problème PacBHMax(V).

Preuve:
L'algorithme A-PacBHMax(V) est clairement polynomial. Il fournit une solution $\hat{S}$ respectant la projection verticale car à chaque étape j, il existe au moins d_j lignes disponibles. En effet, le nombre de lignes disponibles à l'étape j est m moins le nombre de b-barres qui commencent sur les colonnes $j+1-b,\ldots,j-1$, à savoir, $m-\sum_{k=j-1}^{j+1-b} d_k$. En tenant compte de l'expression des d_j et du fait que $m \geq v_j,\ j=1,\ldots,n$, on obtient $m-\sum_{k=j-1}^{j+1-b} d_k \geq v_j - \sum_{k=j-1}^{j+1-b} d_k = d_j$.

Montrons que $\hat{S}$ minimise l'écart maximal. Supposons que dans $\hat{S}$, l'écart maximal correspond à l'écart situé entre deux b-barres placées sur la ligne i et qui commenent aux colonnes j et j'. On a $\sum_{k=j+1}^{j'-1} d_k < m$ car sur la ligne i, il ne commence pas de b-barres entre les colonnes $j+1$ et $j'-1$. Si dans $\hat{S}$ l'écart maximal n'est pas minimum alors il existe une solution S où l'écart maximal est strictement inférieur à celui de $\hat{S}$. Dans S, sur chaque ligne, il y a alors au moins une b-barre qui commence entre les colonnes $j+1$ et $j'-1$. Ainsi $\sum_{k=j+1}^{j'-1} d_k \geq m$, ce qui est en contradiction avec $\sum_{k=j+1}^{j'-1} d_k < m$. Donc $\hat{S}$ est bien une solution telle que l'écart maximal est minimum. ■

Concernant l'existence de solution, nous énonçons le résultat suivant :

Proposition 39:
L'algorithme A-PacBHMax(V) répond au problème $\mathcal{E}-$ PacBHMax(V)

Preuve:
En appliquant l'algorithme A-PacBHMax(V), on trouve une solution pour le problème PacBHMax(V) avec l'écart maximal minimum. En comparant la valeur a et l'écart minimum, on répond au problème $\mathcal{E}-$ PacPBHMax(V). ■

6.2.3 Packing de b-barres avec un écart maximal satisfaisant la projection horizontale (*PacBHMax(H)*)

Ici, on cherche un packing de b-barres horizontales qui respecte la projection horizontale et qui minimise le plus grand écart entre les b-barres et aussi les écarts situés avant la première b-barre et après la dernière b-barre d'une ligne. Le problème de décision correspondant, $\mathcal{E}-PacBHMax(H)$, est défini de la façon suivante :

Données : $H=(h_1,\ldots,h_m)$ un vecteur à coordonnées entières positives et a,b deux entiers positifs.
Question : Existe-t-il un packing de b-barres respectant la projection horizontale H tel que l'écart maximal est inférieur à a ?

Proposition 40:
L'écart maximal minimum vaut $max_i\lceil\frac{c_i}{n_i+1}\rceil$ *avec* $n_i = \frac{h_i}{b}$ *et* $c_i = n - h_i$ *pour* $i = 1, \ldots, n$.

Preuve:
Pour chaque ligne i *pour* $i = 1, \ldots, m$, *on cherche à placer* $n_i = \frac{h_i}{b}$ *b-barres avec l'écart maximal minimal. On note par* $c_i = n - h_i$ *le nombre de cellules non couvertes de la ligne* i. *L'écart maximal minimal* $\lceil\frac{c_i}{n_i+1}\rceil$ *est obtenu lorsque les cellules non couvertes sont équitablement réparties entre les* n_i+1 *écarts. Ainsi, pour PacBHMax(H), l'écart maximal minimum est* $max_i\lceil\frac{c_i}{n_i+1}\rceil$. ■

Lorsque les deux projections orthogonales sont prises en compte, le problème de packing de b-barres n'est pas résolu. Nous allons initier cette étude à travers quelques sous-problèmes polynomiaux. Nous considérons tout d'abord le sous-problème pour lequel la projection horizontale est constante, puis le sous-problème où l'écart maximal est unitaire.

6.2.4 Packing de b-barres horizontales avec un écart maximal satisfaisant la projection horizontale constante et la projection verticale (*PacBHMax(H=c,V)*)

Nous nous intéressons au cas particulier lorsque la projection horizontale est constante, c'est-à-dire, $h_1 = \cdots = h_m = c$. Le problème de décision correspondant, $\mathcal{E}-PacBHMax(H=c,V)$, est défini de la façon suivante :

Données : $V = (v_1, \ldots, v_n)$ un vecteur à coordonnées entières positives et $a, b, c \leq n$ trois entiers positifs.
Question : Existe-t-il un packing de b-barres horizontales respectant la projection verticale V et la projection horizontale $H = (c, \ldots, c)$ tel que l'écart maximal est inférieur à a ?

Les conditions nécessaires et suffisantes d'existence d'une solution sont données par le résultat suivant :

Proposition 41:
$\mathcal{E}-$ *PacBHMax(H=c,V) a pour réponse 'oui' si et seulement si :*
i) $mc = \sum_{j=1}^{n} v_j$,
ii) $\mathcal{E}-$ *PacBHMax(V) a pour réponse 'oui'.*

Preuve:
i) est nécessaire car elle traduit l'égalité entre les sommes des projections horizontales et verticales.
ii) est nécessaire parce que toute solution de PacBHMax(H=c,V) est une solution de PacBHMax(V) (voir section 6.2.2).

Inversement, une solution S donnée par l'algorithme A-PacBHMax(V) au problème PacBHMax(V) est aussi une solution du PacBHMax(H=c,V). En effet, si la condition i) est satisfaite alors l'algorithme A-PacBHMax(V) minimise l'écart maximal entre les b-barres et place $\frac{c}{b}$ *b-barres par ligne.* ■

L'algorithme A-PacBHMax(V) donne alors aussi une solution au problème *PacBH-Max(H=c,V)*.

6.2.5 Packing de b-barres horizontales avec un écart maximal unitaire (*PacBH-Maxu(H,V)*)

Nous montrons ici que lorsque l'écart maximal est unitaire, c'est-à-dire, les b-barres sont adjacentes ou bien séparées par au plus une cellule, le problème *PacBHMaxu(H,V)* est polynomial. Le problème de décision correspondant, $\mathcal{E}-$ *PacBHMaxu(H,V)*, est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ et $V = (v_1, \ldots, v_n)$ deux vecteurs à coordonnées entières positives et b un entier positif.
Question : Existe-t-il un packing de b-barres respectant les projections (H, V) avec un écart maximal inférieur ou égal à 1 ?

Pour la résolution de ce problème, nous proposons l'algorithme A-PacBHMaxu(H,V). L'idée de base de l'algorithme consiste à parcourir les colonnes de gauche à droite. Notons A_j l'ensemble de lignes où une b-barre doit commencer à la colonne j pour respecter l'écart maximal avec la b-barre précédente. À chaque itération j, d_j b-barres commencent à la colonne j : $|A_j|$ sur chacune des lignes de l'ensemble A_j, puis $d_j - |A_j|$ sur les lignes disponibles les plus prioritaires. Si $d_j < |A_j|$ ou il n'existe pas $d_j - |A_j|$ lignes disponibles alors l'algorithme est interrompu et le problème n'admet pas de solution. La ligne i est dite **disponible** à l'étape j lorsque la dernière b-barre se termine à la colonne $j-1$. A la première étape, toutes les lignes sont disponibles. α_i étant le nombre de b-barres horizontales non encore placées à l'étape j de l'algorithme. Les lignes les plus **prioritaires** sont celles qui ont les plus grandes valeurs de α_i.

Algorithme A-PacBHMaxu(H,V)

Pour $j = 1$ **à** $n - b + 1$ **faire**
1. si $d_j < |A_j|$ alors fin ;
2. placer une b-barre sur chacune des lignes de A_j ;
3. s'il n'existe pas $d_j - |A_j|$ lignes disponibles alors fin ;
4. placer $d_j - |A_j|$ b-barres sur les lignes disponibles les plus prioritaires ;
5. si une b-barre commence sur la ligne i alors
$\alpha_i \leftarrow \alpha_i - 1$ et $disp(i,j) = \cdots = disp(i, j+b-1) = faux$;
fin faire ;

La validité de l'algorithme A-PacBHMaxu est donnée par résultat suivant :

Proposition 42:
$\mathcal{E}-$ PacBHMaxu(H,V) est polynomial et l'algorithme A-PacBHMaxu fournit une solution au problème PacBHMaxu(H,V) s'il en existe une.

Preuve:
L'algorithme est clairement polynomial.

Supposons que PacBHMaxu(H,V) admet une solution S. Soit $S_{(1...j)}$ la sous-solution de S correspondant aux b-barres qui commencent sur les colonnes $1, \ldots, j$. Soit $\hat{S}_{(1...j)}$ la solution partielle obtenue par l'algorithme A-PacBHMaxu(H,V) après l'étape j, c'est-à-dire, le placement des b-barres qui commencent sur les colonnes $1, \ldots, j$.

Nous montrons, par récurrence sur j, qu'il existe une solution S du problème telle que $S_{(1..j)} = \hat{S}_{(1..j)}$ pour $j = 1, \ldots, n$.

1. ***Initialisation de la récurrence***

   Suposons que $S_{(1)} \neq \hat{S}_{(1)}$. Il existe une ligne l sur laquelle commence une b-barre dans $\hat{S}_{(1)}$ et sur laquelle ne commence pas de b-barre dans $S_{(1)}$. Il existe aussi une ligne k sur laquelle commence une b-barre dans $S_{(1)}$ et sur laquelle ne commence pas de b-barre dans $\hat{S}_{(1)}$. D'après l'algorithme A-PacBHMaxu(H,V), $\alpha_k \leq \alpha_l$. Dans S, numérotons de 1 à α_l (resp. de 1 à α_k) les b-barres de la ligne l (resp. k). Nous notons C_l^i (resp. C_k^i) la colonne à laquelle commence la $i^{ème}$ b-barre de la ligne l (resp. k). Soit r tel que $\forall i < r$, $C_k^i < C_l^i$, c'est-à-dire, r est le plus petit indice tel que la $r^{ème}$ b-barre de la ligne l commence avant celle de la ligne k dans S. r existe parce que $\alpha_k \leq \alpha_l$.

   Montrons qu'en permutant les $r-1$ premières b-barres des lignes k et l, on obtient une solution équivalente telle qu'une b-barre commence sur la ligne l à la première colonne. Cette permutation respecte l'écart maximal entre la $(r-1)^{ème}$ et la $r^{ème}$ b-barres car les $r^{ème}$ b-barres des lignes k et l commencent toutes les deux à la même colonne $C_l^r = C_k^r = C_l^{r-1} + b$ (voir Figure 6.4). Il est immédiat de vérifier que les projections orthogonales ne sont pas modifiées. En effectuant cette opération autant de fois que nécessaire, on obtient $S_{(1)} = \hat{S}_{(1)}$.

2. ***Supposons qu'il existe une solution*** *S* ***telle que*** *$S_{(1...j)} = \hat{S}_{(1...j)}$* ***et*** *$S_{(1...j+1)} \neq \hat{S}_{(1...j+1)}$. Montrons qu'il existe une solution S telle que $S_{(1...j+1)} = \hat{S}_{(1...j+1)}$. Notons, tout d'abord, que $\hat{S}_{(1...j+1)}$ existe car S est une solution. Le principe de la preuve est identique à celui utilisé pour $j = 1$. Il existe une ligne l (resp. k) disponible et n'appartenant pas à A_{j+1} sur laquelle commence une b-barre à la colonne $j+1$ dans $\hat{S}$ (resp. S), et sur laquelle ne commence pas de b-barre à la colonne $j+1$ dans S (resp. $\hat{S}$). Les b-barres qui commencent à partir de la colonne $j+1$ sont numérotées de 1 à α_l (resp. α_k) sur la ligne l (resp. k). De la même façon que pour $j = 1$, nous définissons les C_l^i, C_k^i et r. Les b-barres de ces deux lignes, numérotées de 1 à $r-1$, placées entre les colonnes $j+1$ et C_l^{r-1} sont permutées de la même manière que pour $j = 1$. Comme pour $j = 1$, cette permutation respecte l'écart maximal entre la $(r-1)^{ème}$ b-barre et la $r^{ème}$ sur les deux lignes. Il est facile de voir que les projections orthogonales ne sont pas modifiées. Cette permutation respecte en plus l'écart maximal au niveau de la colonne $j+1$ car les dernières b-barres des deux lignes se terminent à la colonne j. Ceci est dû au fait que les lignes k et l sont disponibles à l'étape $j+1$ et n'appartiennent pas à A_{j+1}. En effectuant cette opération autant de fois que nécessaire, on obtient la solution S cherchée.* ■

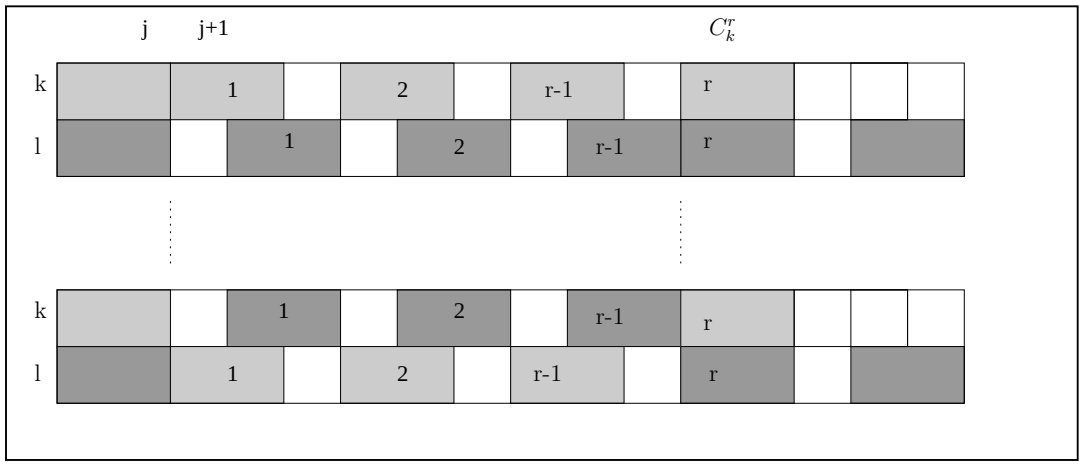


FIG. 6.4 – Permutation des b-barres $1, \ldots, r-1$ pour *PacBHMaxu(H,V)*

Cet algorithme n'est bien sûr valide que si l'écart maximal est unitaire, sinon la permutation des lignes k et l ne respecterait pas l'écart maximal au niveau de la colonne $j+1$. Par conséquent, cet algorithme ne permet pas de résoudre le problème de packing avec un écart maximal quelconque.

Le tableau 6.1 synthétise la complexité des problèmes de packing de b-barres.

Projections	**Sans écart**	**Ecart min**	**Ecart max**
H	P	P	P
V	P	P	P
H et V	P	P	?, P si H constant ou écart = 1

TAB. 6.1 – Complexité des problèmes de packing de b-barres horizontales. P : polynomial

6.3 Packing de barres de longueurs non fixées

Contrairement aux sections précédentes, nous considérons des barres horizontales de longueurs supérieures ou égales à deux. Le problème de packing de barres de longueurs supérieures ou égales à 1 est équivalent au problème de reconstruction de matrices binaires. Ceci va donner plus de souplesse au niveau de placement de barres, mais elle va augmenter la taille du domaine de recherche de solutions admissibles.

Pour pouvoir appliquer le formalisme relatif aux problèmes de packing de b-barres et en particulier l'algorithme A-PacBHMin(H,V), il faut connaître le nombre de barres qui commencent sur chaque colonne, ce qui n'est pas a priori le cas. Nous allons d'abord étudier le sous-problème avec deux longueurs possibles des barres. Ensuite, le sous-problème dans lequel l'écart entre les barres est constant, puis le sous-problème avec la contrainte d'un écart maximal.

6.3.1 Packing de 2-3-barres (*Pac23BH(H,V)*)

Dans cette section, nous cherchons un packing avec des 2-barres et 3-barres respectant les projections orthogonales. A l'exception du cas général, les projections orthogonales sont le nombre de 2-barres et de 3-barres par ligne et par colonne, c'est-à-dire, on connaît les projections de chaque type de barre. Le problème de décision associé, $\mathcal{E}-$ *Pac23BH(H,V)*, est le suivant :

Données : $H^2 = (h_1^2, \ldots, h_m^2)$, $H^3 = (h_1^3, \ldots, h_m^3)$, $V^2 = (v_1^2, \ldots, v_n^2)$ et $V^3 = (v_1^3, \ldots, v_n^3)$ quatre vecteurs à coordonnées entières positives.
Question : Existe-t-il un packing de 2-barres et 3−barres respectant (H^2, V^2) pour les 2-barres et (H^3, V^3) pour les 3-barres ?

Proposition 43:
Le problème Pac23BH(H,V) est au moins aussi difficile que le problème de reconstruction d'un tableau 3-coloré.

Preuve:
Nous rappelons que dans un tableau 3-coloré, chaque cellule est colorée par les couleurs a ou b ou non colorée (voir chapitre 4). Les projections donnent le nombre de cellules de couleurs a et b dans chacune des lignes et des colonnes. Nous montrons que toute instance de ce problème peut être réduite en une instance du problème Pac23BH(H,V).

Soit Q une instance du problème de reconstruction d'un tableau 3-coloré respectant (H'^a, V'^a) pour la couleur a et (H'^b, V'^b) pour la couleur b avec :

- *h'^a_i (resp. h'^b_i) : le nombre de cellules de couleur a (resp. b) sur la ligne i.*
- *v'^a_j (resp. v'^b_j) : le nombre de cellules de couleur a (resp. b) sur la colonne j.*

Nous définissons une instance P de taille $m \times 3n$ du problème Pac23BH(H,V) respectant les projections (H^2, V^2) et (H^3, V^3) en associant les couleurs a et b respectivement aux 2-barres et 3-barres :

- $v^2_{3j-2} = v^2_{3j-1} = v'^a_j$, $v^2_{3j} = 0$, $j = 1, \ldots, n$,
- $v^3_{3j-2} = v^3_{3j-1} = v^3_{3j} = v'^b_j$, $j = 1, \ldots, n$,
- $h^2_i = h'^a_i$, $i = 1, \ldots, m$,
- $h^3_i = h'^b_i$, $i = 1, \ldots, m$.

Les problèmes P et Q sont équivalents car d'après les projections (H^2, V^2) et (H^3, V^3), les barres commencent seulement sur les colonnes $3j-2$. Les nombres de 2-barres et 3-barres qui commencent à la colonne $3j-2$ dans P sont égaux au nombre de cellules colorées respectivement par a et b sur la colonne j dans Q (voir Figure 6.5). Ainsi le problème Pac23BH(H,V) est au moins aussi difficile que le problème de reconstruction d'un tableau 3-coloré dont la complexité est inconnue. ■

Une forme plus générale de ce résultat est donnée par la proposition suivante :

Proposition 44:
Le problème de packing de b_1-barres, b_2-barres, ... , b_k-barres est NP-complet pour tout entier $k \geq 3$ fixé.

Preuve:
Pour établir cette proposition, nous utilisons une réduction polynomiale à partir du problème de reconstruction d'un tableau (k+1)-coloré (voir chapitre 4). Nous rappelons que ce problème est NP-complet pour tout ($k+1 \geq 4$) et de complexité inconnue pour $k+1 = 3$ [1], [2]. ■

Notons que lorsque les projections donnent le nombre de cellules couvertes, le problème de packing 2-barres et 3-barres est de complexité inconnue. Il est équivalent au problème de packing de barres de longueurs supérieures ou égales à deux car tout entier $b > 1$ peut se mettre sous la forme $b = 2\alpha + 3\beta$ avec α et β deux entiers positifs (voir section 6.3.5).

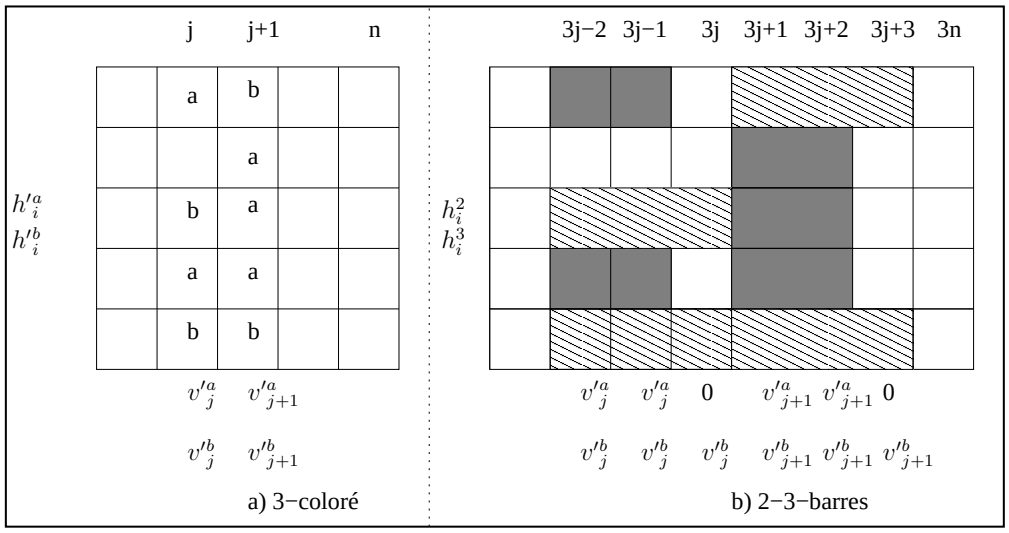


FIG. 6.5 – Packing de $2-3-$barres

6.3.2 Packing de barres horizontales avec un écart constant (*PacHC(H,V)*)

Nous cherchons un packing de barres de longueurs supérieurs ou égales à b avec un écart égal à a cellules respectant les projections orthogonales. Le nombre de cellules non couvertes avant la première barre ou après la dernière barre sur chaque ligne est égal à 0 ou à a. Dans les problèmes de planification de personnel, ce problème modélise les emplois du temps où les vacations sont de longueurs supérieures ou égales à b périodes de temps séparées par exactement a périodes de repos. Le problème de décision correspondant, $\mathcal{E}-$ *PacHC(H,V)*, est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ et $V = (v_1, \ldots, v_n)$ deux vecteurs à coordonnées entières positives et $a, b \leq n$ deux entiers positifs.
Question : Existe-t-il un packing de barres de longueurs supérieures ou égales à b respectant les projections (H, V) tel que les écarts entre les barres valent a ?

Proposition 45:
Les problèmes PacHC(H,V) et $\mathcal{E}-$ PacHC(H,V) sont polynomiaux.

Preuve:
Rappelons que le problème PacBHMin(H,V) est polynomial et consiste à placer des b-barres espacées d'au moins a cellules (voir section 6.2.1). Tout ensemble de cellules de valeur 0 situées entre deux b-barres consécutives représente une barre de longueur supérieure à a. Il en découle que PacBHMin(H,V) avec des b-barres et un écart minimal égal à a est équivalent à PacHC(H',V') avec des barres de longueurs supérieures ou égales à a, un écart constant égal à b et $h'_i = n - h_i,\ i = 1, \ldots, m$, *et* $v'_j = m - v_j,\ j = 1, \ldots, n$. ■

6.3.3 Packing de barres horizontales avec un écart maximal satisfaisant la projection horizontale (*PacHMax(H)*)

Nous considérons ici le problème de packing de barres de longueurs supérieures à deux avec l'écart maximal minimum. On ne peut pas utiliser la formulation proposée pour le problème *PacHMax(H)* car on ne connaît pas le nombre des barres par ligne. Le problème de décision correspondant, $\mathcal{E}-$ *PacHMax(H)*, est défini de la façon suivante :

Données : $H = (h_1, \ldots, h_m)$ un vecteur à coordonnées entières positives et $a \leq n$ un entier positif.
Question : Existe-t-il un packing de barres horizontales respectant la projection horizontale H tel que l'écart maximal est inférieur à a ?

Le résultat suivant donne l'écart maximal minimum pour le problème *PacHMax(H)* :

Proposition 46:
L'écart maximal minimum est $max_i \frac{n-h_i}{1+\lfloor \frac{h_i}{2} \rfloor}$ *pour* $i = 1, \ldots, m$.

Preuve:
Calculons l'écart maximal minimal sur la ligne i. Le nombre de cellules non couvertes est $n - h_i$. *L'écart maximal minimal est obtenu lorsque les cellules non couvertes sont équitablement réparties entres les écarts. Cherchons alors à maximiser le nombre d'écarts. Ceci est équivalent à maximiser le nombre des barres. Si* h_i *est pair, il y a au plus* $\frac{h_i}{2}$ *dominos. Si* h_i *est impair, il y a au plus* $\lfloor \frac{h_i}{2} \rfloor - 1$ *dominos et un trimino. Dans les deux cas, il y a au plus* $\lfloor \frac{h_i}{2} \rfloor$ *barres et* $\lfloor \frac{h_i}{2} \rfloor + 1$ *écarts. Ainsi l'écart maximal minimal est* $\frac{n-h_i}{1+\lfloor \frac{h_i}{2} \rfloor}$. *Donc sur l'ensemble des lignes, l'écart maximal minimum est* $max_i \frac{n-h_i}{1+\lfloor \frac{h_i}{2} \rfloor}$. ■

La preuve de cette proposition se transforme en l'algorithme A-PacHMax(H).

Algorithme A-PacHMax(H)

Pour $i = 1$ **à** m **faire**
1. si h_i est pair alors placer $\frac{h_i}{2}$ dominos équitablement répartis sur la ligne i ;
2. si h_i est impair alors placer $\frac{h_i-1}{2} - 1$ dominos et un trimino équitablement répartis sur la ligne i ;
fin faire ;

6.3.4 Packing de barres horizontales avec un écart maximal satisfaisant la projection verticale (*PacHMax(V)*)

Dans cette section, nous cherchons un packing satisfaisant la projection verticale V et qui minimise l'écart maximal. Le problème de décision correspondant, $\mathcal{E}-$ *PacHMax(V)*, est défini de la façon suivante :

Données : $V = (v_1, \ldots, v_n)$ un vecteur à coordonnées entières positives et $a \leq n$ un entier positif.
Question : Existe-t-il un packing de barres horizontales respectant la projection verticale V tel que l'écart maximal entre les barres est inférieur à a ?

Nous établissons des conditions nécessaires d'existence de solution :

Proposition 47:
Pour que $\mathcal{E}-$ *PacHMax(V) ait pour réponse 'oui' il faut que :*
i) $v_1 \leq v_2,\ v_n \leq v_{n-1}$ *et* $v_j \leq v_{j+1} + v_{j-1},\ j = 2, \ldots, n-1,$
ii) $m \leq v_j + v_{j+a} + \frac{1}{2}\sum_{k=j+1}^{j+a-1} v_k,\ j = 1, \ldots, n-a.$

Preuve:
i) est évidente car les barres sont de longueurs supérieures ou égales à deux.
ii) Pour respecter la contrainte de l'écart maximal, il faut pour chaque ligne, couvrir au moins une cellule sur chaque intervalle de $a+1$ *cellules consécutives. Le nombre maximal de barres pouvant couvrir les cellules situées de la colonne* j *à la colonne* $j+a$ *est :* $v_j + v_{j+a} + \frac{1}{2}\sum_{k=j+1}^{j+a-1} v_k,\ j = 1, \ldots, n-a$. *Les projections des colonnes* j *et* $j+a$ *sont affectées du coefficient* 1 *parce que les barres qui se terminent à la colonne* j *et celles qui commencent à la colonne* $j+a$ *ont leurs autres cellules hors de l'intervalle. Pour les autres colonnes, le facteur* $\frac{1}{2}$ *correspond au fait qu'une barre est constituée d'au moins deux cellules. Pour couvrir au moins une cellule par ligne, il faut que le nombre maximal de barres soit supérieur au nombre des lignes.* ■

Nous proposons l'algorithme polynomial A-PacHMax(V) qui donne une solution $\hat{S}$ avec l'écart maximal minimum lorsqu'il en existe une où il s'arrête en indiquant qu'il y en a pas. Notons $\hat{S}(i,j) = 1$ si la cellule (i,j) est recouverte par une barre et $\hat{S}(i,j) = 0$ dans le cas contraire. Une barre est alors un ensemble d'au moins deux cellules de valeur 1 adjacentes horizontalement. A l'étape j, les lignes sont regroupées en trois classes (voir Figure 6.6) :

C_1^j : les lignes, i, avec $\hat{S}(i, j-1) = 1$ et $\hat{S}(i, j-2) = 0$.
C_2^j : les lignes, i, avec $\hat{S}(i, j-1) = 1$ et $\hat{S}(i, j-2) = 1$.
C_3^j : les lignes, i, avec $\hat{S}(i, j-1) = 0$.

L'algorithme A-PacHMax(V) affecte à chaque itération j, la valeur 1 à d_j cellules de la colonne j en commençant par les lignes de la classe C_1^j, ensuite les lignes les plus prioritaires de la classe C_3^j et enfin les lignes de la classe C_2^j. La ligne la plus prioritaire de C_3^j est celle qui admet la dernière cellule de valeur 1 la plus à gauche.

Notons que pour toute ligne i de la classe C_1^j, on doit avoir $\hat{S}(i,j) = 1$ pour respecter la contrainte de la longueur minimale des barres.

Algorithme A-PacHMax(V)

1. $d_j = v_j,\ j = 1, \ldots, n$;
2. pour $j = 1$ **à** n **faire**
2.1 $\hat{S}(i,j) = 1$ pour toute ligne i de C_1^j et $d_j = v_j - |C_1^j|$;
2.2 $\hat{S}(i,j) = 1$ pour $\min(d_j, v_{j+1})$ lignes les plus prioritaires de C_3^j ;
2.3 si $d_j - \min(d_j, v_{j+1}) > |C_2^j|$ alors fin ;
2.4 $\hat{S}(i,j) = 1$ pour $d_j - \min(d_j, v_{j+1})$ lignes de C_2^j ;
fin faire ;

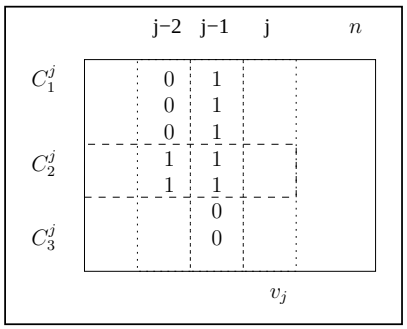


FIG. 6.6 – Classes des lignes à l'étape j

La validité de l'algorithme A-PacHMax(V) est donnée par le résultat suivant :

Proposition 48:
Le problème PacHMax(V) est polynomial et l'algorithme A-PacHMax(V) en fournit une solution s'il en existe une.

Preuve:
L'algorithme est clairement polynomial. Le principe de la preuve est très voisin de celui de la démonstration de la validité de l'algorithme A-PacBHMin(H,V) (voir section 6.2.1).

Supposons que le problème PacHMax(V) admet une solution S. Soit $S_{(1\ldots j)}$ la sous-solution de S correspondant aux $'1'$ placés dans les colonnes $1, \ldots, j$. Soit $\hat{S}_{(1\ldots j)}$ la solution partielle obtenue par l'algorithme A-PacHMax(V) à l'étape j, c'est-à-dire, les positions des $'1'$ dans les colonnes $1, \ldots, j$. Nous montrons, par récurrence sur j, qu'il existe une solution S telle que $S_{(1\ldots j)} = \hat{S}_{(1\ldots j)}$.

1. ***Initialisation de la récurrence***
   Suposons que $S_{(1)} \neq \hat{S}_{(1)}$. Il existe deux lignes l et k telles que $\hat{S}(l,1) = 1, S(l,1) = 0$ et $\hat{S}(k,1) = 0, S(k,1) = 1$. En permutant les lignes k et l dans S, on obtient $S(k,1) = 0$ et $S(l,1) = 1$. On réitère ce type de permutation jusqu'à avoir $S_{(1)} = \hat{S}_{(1)}$.
2. ***Supposons qu'il existe une solution** S **telle que** $S_{(1\ldots j)} = \hat{S}_{(1\ldots j)}$ et $S_{(1\ldots j+1)} \neq \hat{S}_{(1\ldots j+1)}$. Montrons qu'il existe S telle que $S_{(1\ldots j+1)} = \hat{S}_{(1\ldots j+1)}$. Remarquons tout*

d'abord que $\hat{S}_{(1\ldots j+1)}$ existe car S est une solution au problème PacHMax(V). Le principe de la preuve est identique à celui utilisé pour $j = 1$. Il existe deux lignes l et k telles que $\hat{S}(l, j+1) = 1, S(l, j+1) = 0$ et $\hat{S}(k, j+1) = 0, S(k, j+1) = 1$. Nous distinguons tous les cas possibles selon la classe d'appartenance des lignes l et k et pour chacun nous proposons une transformation (permutation) appropriée.

Premier cas *si l et k appartiennent toutes les deux à C_3^{j+1} ou à C_2^{j+1} alors en permutant les lignes k et l dans S à partir de la colonne $j+1$, on obtient $S(k, j+1) = 0$ et $S(l, j+1) = 1$.*

Deuxième cas *si $k \in C_3^{j+1}$ et $l \in C_2^{j+1}$ alors permuter les lignes k et l à partir de la colonne $j+1$.*

Troisième cas *si $k \in C_2^{j+1}$, $l \in C_3^{j+1}$ et $S(k, j+2) = 1$ alors permuter k et l à partir de la colonne $j+1$.*

Quatrième cas *si $k \in C_2^{j+1}$, $l \in C_3^{j+1}$, $S(k, j+2) = 0$ et $S(l, j+2) = 1$ alors $S(k, j+1) = 0$ et $S(l, j+1) = 1$.*

Cinquième cas *si $k \in C_2^{j+1}, l \in C_3^{j+1}, S(k, j+2) = 0$ et $S(l, j+2) = 0$ alors chercher un autre couple (k, l) vérifiant l'un des quatre cas précédents et effectuer dans S la transformation correspondant. Supposons qu'à ce stade si $S_{(1\ldots j+1)} \neq \hat{S}_{(1\ldots j+1)}$ alors elles se différent par des lignes k et l appartenant au cinquième cas, c'est-à-dire $\hat{S}(l, j+1) = 1$, $S(l, j+1) = 0$, $\hat{S}(k, j+1) = 0$ et $S(k, j+1) = 1$. La transformation s'effectue par le biais d'une ligne auxiliaire i_0. Pour déterminer i_0, nous introduisons les sous-classes suivantes :*

$C'2j = \{i \in C_2^{j+1} / \hat{S}(i, j+1) = 1,\ S(i, j+1) = 1\}$.
$C2j = \{i \in C_2^{j+1} / \hat{S}(i, j+1) = 0,\ S(i, j+1) = 1\}$.
$C'3j = \{i \in C_3^{j+1} / \hat{S}(i, j+1) = 1,\ S(i, j+1) = 1\}$.
$C3j = \{i \in C_3^{j+1} / \hat{S}(i, j+1) = 1,\ S(i, j+1) = 0\}$.

On observe les propriétés suivantes :

(a) $|C3j| = |C2j|$.

(b) $v_{j+2} \geq |C'3j| + |C3j|$ *car d'après l'algorithme A-PacHMax(V), le nombre de barres qui commencent dans la colonne $j+1$ est inférieur à v_{j+2}.*

(c) Dans la colonne $j+1$, $|C3j| + |C'3j|$ barres commencent dans $\hat{S}$ et $|C'3j|$ barres commencent dans S. De plus, dans S, il y a $|C3j|$ lignes n'appartenant pas à $C'3j$ et sur lesquelles il y a une cellule de valeur 1 dans la colonne $j+2$.

(d) D'après (a) et (c), pour chaque ligne $k \in C2j$, il existe une ligne $l \in C3j$ et une ligne $i_0 \notin C'3j$ avec $S^j(i_0, j+2) = 1$ (voir Figure 6.7).

La transformation est la suivante : on permute tout d'abord les lignes k et i_0 à partir de la colonne $j+2$. Ensuite, on permute k et l à partir de la colonne $j+1$ (troisième cas). On réitère ce type de permutation tant qu'il existe deux lignes l et k appartenant au $5^{ème}$ cas, jusqu'à obtenir la solution S cherchée. ■

En appliquant l'algorithme A-PacHMax(V), on trouve une solution à PacHMax(V) avec l'écart maximal minimum. En comparant la valeur a et l'écart minimum, on répond au problème $\mathcal{E}-$ PacHMax(V). D'où le résultat suivant :

Proposition 49:
L'algorithme A-PacHMax(V) répond au problème $\mathcal{E}-$ PacHMax(V).

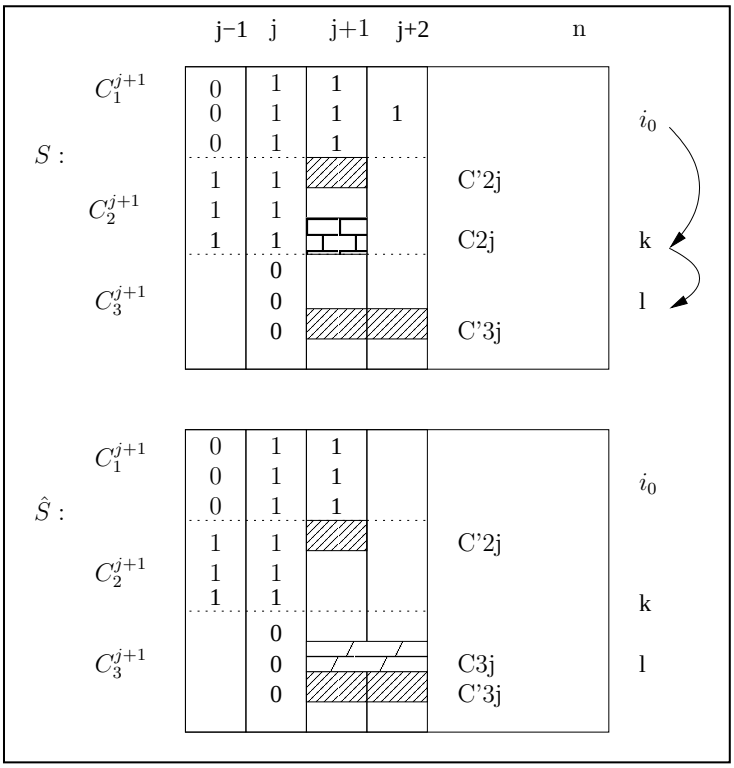


FIG. 6.7 – Permutation des lignes k, i_0 et l à l'étape j+1

Nous généralisons maintenant les problèmes *PacHMax(H)* et *PacHMax(V)* pour prendre en compte les deux projections orthogonales.

6.3.5 Packing de barres horizontales avec un écart maximal (*PacHMax(H, V)*)

Nous cherchons un packing de barres horizontales respectant les projections orthogonales (H, V) tel que l'écart maximal est inférieur à a. Le résultat suivant donne des conditions nécessaires d'existence d'une solution :

Proposition 50:
Pour que PacHMax(H,V) admette une solution, il faut que :
i) $a \geq \frac{n-h_i}{1+\lfloor \frac{h_i}{2} \rfloor}$ *pour* $i = 1, \ldots, m$.
ii) $v_1 \leq v_2,\ v_n \leq v_{n-1}$ *et* $v_j \leq v_{j+1} + v_{j-1},\ j = 2, \ldots, n-1$,
iii) $m \leq v_j + v_{j+a} + \frac{1}{2}\sum_{k=j+1}^{j+a-1} v_k,\ j = 1, \ldots, n-a$.

Ces condition sont nécessaires car toute solution au problème *PacHMax(H,V)* est aussi une solution aux problèmes *PacHMax(V)* et *PacHMax(H)*.

Nous remarquons que le problème de packing de barres correspond au problème de packing de barres avec un écart maximal égal à n. Ainsi le problème de packing de barres avec un écart maximal est au moins aussi difficile que le problème de packing de barres. On

remarque également que le problème de packing de barres respectant une seule projection horizontale ou verticale est polynomial.

Le tableau 6.2 donne une synthèse sur la complexité des problèmes de packing par des barres de longueurs supérieures ou égales à deux.

Projections	**Sans écart**	**Ecart constant**	**Ecart max**
H	P	P	P
V	P	P	P
H et V	?	P	?

TAB. 6.2 – Complexité des problèmes de packing de barres horizontales. P : polynomial

6.4 Conclusion

Nous avons considéré dans ce chapitre le problème de packing de barres horizontales. Il s'agit d'un problème de reconstruction pour lequel il existe très peu de travaux. Nous avons proposé différents algorithmes, essentiellement de type glouton pour résoudre les problèmes de packing sous différentes contraintes d'écart. Nous avons montré que lorsqu'une projection est relâchée, ces problèmes sont polynomiaux. Nous avons montré également que le problème de packing de b-barres avec un écart minimal est polynomial. Pour les problèmes de complexité inconnue, nous avons dégagé des conditions nécessaires d'existence d'une solution.

Ces résultats fournissent un point de départ à l'étude de nouveaux problèmes de packing de barres puisque à notre connaissance aucun résultat n'a été publié concernant ce type de problèmes de packing.

Bibliographie

[1] M. Chrobak and C. Dürr. Reconstructing polyatomic structures from x-rays : Np completness proof for three atoms. *Theoretical computer science*, 259(1) :81–98, 2001.

[2] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computational complexity of determining polyatomic structures by x-rays. *Theoretical computer science*, 233 :91–106, 2000.

[3] C. Picouleau. Reconstruction of domino tiling from its two orthogonal projections. *Theoretical computer science*, 255(1) :437–447, 2001.

Chapitre 7

Application de la tomographie discrète à la planification de personnel

Ce chapitre constitue une application de la tomographie discrète à la planification de personnel. Il s'agit d'affecter des périodes de repos et des vacations à des employés tout en respectant différentes contraintes.

La première partie est consacrée à un rapide état de l'art de ce très vaste domaine qui a suscité une abondante littérature. Nous excluons de cette étude bibliographique les

problèmes où la charge est décrite par une liste de tâches insécables (transport aérien et ferroviaire). Nous ne traiterons pas non plus la prévision de la charge et nous considérons que les besoins sont connus à l'avance.

Dans la deuxième partie, nous considérons des problèmes de planification avec deux ou trois jours de repos par semaine, nous dégageons certaines propriétés et nous mettons en évidence le lien avec les problèmes de la tomographie discrète. Ensuite, nous résolvons ce que nous appelons le problème de base à l'aide d'un algorithme polynomial en trois étapes. Puis, nous ajoutons des contraintes sur les repos mensuels et nous modifions le précédent algorithme pour les prendre en compte. Enfin, nous imposons des contraintes sur le quota de week-ends libres et nous étudions sous certaines conditions ce nouveau problème.

Dans la troisième partie, nous reprenons la démarche adoptée dans la deuxième partie pour traiter les problèmes de planification avec trois ou quatre jours de repos par semaine.

7.1 Description des problèmes de planification de personnel

Le problème de planification de personnel consiste à planifier les horaires de travail de personnel pour couvrir une demande (charge) tout en respectant un ensemble de règles plus au moins rigides (contraintes). C'est un problème pratique très important pour les organismes opérant sept jours par semaines ou 24 heures par jour car il a un impact direct sur le coût de la main d'oeuvre, la qualité du service offert, la productivité et la satisfaction des employés, etc. Dans le cas général, ce problème est NP-difficile car sa complexité dépend de plusieurs paramètres tels que la disponibilité et les préférences des employés, la variation et la périodicité de la demande, les règles de production, les conventions de travail, etc.

7.1.1 Définitions et notations

Tout au long de ce chapitre, nous adoptons les définitions et les notations suivantes :
Une semaine est une séquence de 7 jours allant du lundi au dimanche. Les jours de la semaine sont représentés par $d_j,\ j = 1, \ldots, 7$.
Un week-end est une séquence "samedi-dimanche". Certaines contraintes fixent un quota de week-ends libres.
L'horizon de planification est composé de W semaines $w_k,\ k = 1, \ldots, W$.
Un mois est une séquence de quatre semaines successives.
Une séquence de travail est une suite de jours de travail sans repos. Les séquences de travail ont souvent une longueur inférieure ou égale à six jours.
Une vacation est un programme journalier non nominatif. Elle est caractérisée par l'heure de début, les pauses et l'heure de fin. Les vacations sont regroupées par étiquettes. Certaines contraintes de planification portent sur l'enchaînement des vacations.
Une étiquette (type de vacation) est un ensemble de vacations ayant les mêmes heures de début et de fin. Deux vacations appartenant à la même étiquette peuvent différer par les pauses.

La charge ou la demande représente le nombre d'employés effectuant chacune des vacations par jour.
La période est le plus petit intervalle horaire où la charge est exprimée (un quart d'heure, une demi-heure, une journée, etc). Pour les problèmes que nous étudierons dans la deuxième et la troisième parties, la période est la journée.
Une charge cyclique est une charge de périodicité la semaine. Toutefois, la charge peut varier au cours de la semaine.
Un programme cyclique ou cycle est un programme où au bout d'un certain nombre de semaines, chaque employé reprend son programme de départ.
Une classe est un ensemble d'employés ayant les mêmes préférences.

7.1.2 Contraintes de planification

Certaines contraintes sont à prendre en compte dans les problèmes réels de planification de personnel. Elles sont essentiellement dues à la charge et à la planification. **Les contraintes de charge** sont imposées par la politique de gestion de l'organisme. Elles indiquent l'effectif devant être présent au cours de chaque période. **Les contraintes de planification** sont imposées par les préférences des employés et la réglementation du travail. On retrouve par exemple :

1. Les contraintes de placement de jours de repos qui englobent les contraintes suivantes :
   - un nombre minimal de jours de repos hebdomadaires : chaque employé bénéficie souvent de deux jours de repos par semaine ;
   - des couples de jours de repos : les jours de repos hebdomadaires ne sont pas isolés ;
   - quota de week-ends en repos : chaque employé prend au moins A week-ends libres sur B ;
   - une longueur maximale de séquences de travail : la longueur maximale est souvent fixée à six jours.
2. Les contraintes d'enchaînement de vacations indiquent les vacations pouvant être enchaînées puisqu'une vacation d'un jour donné n'est pas suivie par n'importe quelle vacation le lendemain, par exemple, les vacations *soir* et *matin*.

7.2 Etat de l'art sur les problèmes de planification de personnel

7.2.1 Un classement des problèmes

Selon l'horizon et la période de la planification, on distingue trois types de problèmes de planification (voir [2]) : le problème de construction de vacations *(PCV)* (Shift scheduling problem), le problème de construction de grilles de travail *(PCG)* (Days-off scheduling problem) et le problème de construction de cycles *(PCC)* (Tour scheduling problem).

Problème de construction de vacations (*PCV*)

Le problème de construction de vacations, *PCV*, consiste à déterminer les emplois du temps (vacations) des employés afin de couvrir la charge journalière à moindre côut. Le coût d'une vacation est souvent proportionnel à sa longueur. La charge représente le nombre d'employés par période. La plupart des études supposent implicitement que la charge est la même tous les jours et résolvent le *PCV* pour une *journée* type. Si la charge fluctue d'un jour à un autre, il faudra résoudre un *PCV* par jour. Nous distinguons deux types d'approches de résolution : les approches explicites ou de couverture d'ensemble (voir [8]) et les approches implicites basées sur la programmation mathématique (voir [10], [5], [3] et [13]).

Problème de construction de grilles (*PCG*)

Après la résolution du *PCV*, les vacations sont regroupées en *étiquettes*. Le problème *PCG* consiste à affecter les jours de repos et les vacations aux employés à moindre effectif tout en respectant toutes les contraintes de planification et de charge. La charge représente le nombre d'employés par vacation et par jour (voir Tableau 7.1). Elle est souvent cyclique de périodicité *une semaine*. Une solution à *PCG* est représenté par des grilles cyclique. Une grille est un tableau de m lignes correspondant aux employés et de 7 colonnes correspondant aux jours. La valeur de la cellule (i, j) indique la vacation exercée par l'employé i le jour j. Les employés effectuent un roulement sur le programme hebdomadaire (voir Tableau 7.1). 'R' représente un jour de repos, 'M' représente une vacation le matin et 'S' représente une vacation le soir. Durant la première semaine, l'employé i exécute le programme i. Au cours de la deuxième semaine, l'employé i exerce le programme $i + 1 \, mod \, 4$ et ainsi de suite. A la cinquième semaine, chaque employé reprend son programme de départ.

Selon le type de vacation, on distingue deux sous-problèmes : le problème de placement de jours de repos (*PPR*) (days-off scheduling problems, single shift sheduling problems) où un seul type de vacations est considéré. Cependant, dans le problème de construction de grilles étiquetées (*PGE*) (multiple shift scheduling problems), plusieurs types de vacations sont considérés. Le sous-problème *PPR* sera bien détaillé car nous étudierons des problèmes de ce type.

Problème de construction de cycles (*PCC*)

Une question importante dans la planification de personnel concerne la construction simultanée des vacations (*PCV*) et des grilles (*PCG*). Le problème *PCC* consiste à élaborer un programme détaillé sur l'horizon de planification à moindre effectif en satisfaisant la charge journalière et les règles (contraintes) de planification. Il est est défini par :

- le nombre d'employés nécessaires par période. Pour traduire la charge en nombre d'étiquettes, il faut résoudre les problèmes *PCV*,
- les classes des employés. Chaque classe est définie par ses préférences et ses contraintes.

Jour	**L**	**M**	**M**	**J**	**V**	**S**	**D**
Charge	2M,2S	2M,2S	M,2R,S	2R,2S	M,2R,S	M,R,2S	2M,S,R

Employé \ jour	**L**	**M**	**M**	**J**	**V**	**S**	**D**
1	M	M	R	S	S	S	R
2	S	S	S	S	R	R	M
3	M	M	M	R	R	S	S
4	S	S	R	R	M	M	M

TAB. 7.1 – Grille de travail : en haut la charge, en bas la grille de solution

Du fait de sa généralité, une large gamme de méthodes de résolution spécifiques ont été envisagées (voir, par exemple [11], [7] et [12] ou Alfares [2] pour un état de l'art détaillé sur les différentes méthodes).

7.2.2 Problème de placement de repos (*PPR*)

Le problème de placement de repos *(PPR)* consiste à affecter seulement les jours de travail et les jours de repos aux employées tout en respectant la charge et les contraintes de placement de repos. La charge représente le nombre d'employés présents par jour.

La démarche suivie pour résoudre ce problème consiste, tout d'abord, à calculer l'effectif minimal nécessaire et ensuite, à développer des algorithmes qui construisent des solutions admissibles. On retrouve dans la littérature trois principales variantes du problème *PPR* : *PPR* avec repos hebdomadaires découplés, *PPR* avec repos hebdomadaires couplés et *PPR* avec quatre jours de travail par semaine.

[A] PPR avec repos hebdomadaires découplés

Les premières études ont découplé les semaines, c'est-à-dire qu'elles ont négligé les contraintes portant sur plus qu'une semaine. Dans ce cas, une résolution hebdomadaire est suffisante pour construire une solution sur l'horizon de planification. Le problème de référence est le suivant :

$$P_d \begin{cases} \text{Minimiser l'effectif} \\ \text{s.c.} \\ \text{Une charge cyclique } v_j,\ j = 1, \ldots, 7, \text{ donnée.} \\ \text{Deux jours de repos consécutifs par employé et par semaine.} \end{cases}$$

Tiberwala et al. [14] proposent une modélisation en couverture d'ensemble et montrent qu'il y a 7 types de programmes hebdomadaires (pattern) parce que les deux jours de repos sont enchaînés. Ils développent un algorithme optimal pour résoudre le problème de couverture associé. L'idée générale de l'algorithme est la suivante : à chaque itération de l'algorithme, on choisit le programme dont les jours de repos sont $(j_0, j_0 + 1)$ si

$max(v_{j_0}, v_{j_0+1}) = min_j(max(v_j, v_{j+1}))$. Il est évident que le nombre d'itérations dépend de l'amplitude de la charge hebdomadaire puisque l'effectif augmente de 1 à chaque itération.

Pour améliorer la résolution du problème P_d, Baker [4] a raisonné sur les jours de repos au lieu des jours de travail et a montré le théorème suivant :

Théorème 6 (Baker):
Pour que le problème P_d admette une couverture exacte il faut que :
i) $\sum_{j=1}^{7} v_j$ *est un multiple de* 5,
ii) $max_j v_j \leq \frac{1}{5} \sum_{j=1}^{7} v_j$.

Preuve:
$\sum_{j=1}^{7} v_j$ *est le nombre de jours de travail par l'ensemble des employés, d'où la condition i) est nécessaire car chaque employé travaille 5 jours par semaine.* $max_j v_j$ *représente l'effectif nécessaire pour couvrir la charge maximale, d'où si* $5max_j v_j > \sum_{j=1}^{7} v_j$ *alors il y a une sur-couverture.* ■

Pour construire une solution, Baker introduit les variables x_j pour donner le nombre d'employés en repos les jours j et $j+1$. Ensuite, il exprime les x_j en fonction de la charge journalière et de l'effectif total afin d'assurer une couverture exacte de la charge. Pour un effectif w donné, la solution (x_j) est unique. Si cette solution n'est pas admissible alors w n'assure pas une couverture exacte. L'objectif revient alors à construire une solution avec le minimum de sur-effectif. Il remarque que la solution du problème initial correspond à une solution sans sur-effectif pour un problème modifié où les charges sont artificiellement augmentées. Lorsque l'une des conditions *i)* ou *ii)* n'est pas satisfaite, Baker résout le problème augmenté pour calculer une borne inférieure de l'effectif.

[B] PPR avec repos hebdomadaires couplés

Dans ce problème, on introduit les contraintes de quota de week-ends libres et les séquences maximales de travail pour coupler les semaines. Burns et Carter [6] ont résolu le problème suivant :

$$P_c \begin{cases} \text{Minimiser l'effectif} \\ \text{s.c.} \\ \text{Une charge cyclique } v_j,\ j = 1, \ldots, 7, \text{ donnée.} \\ \text{Chaque employé bénéficie de deux jours de repos par semaine.} \\ \text{Chaque employé dispose de } A \text{ week-ends de repos sur } B. \\ \text{Les séquences de travail sont de longueurs inférieures à 6.} \end{cases}$$

Ils calculent tout d'abord l'effectif minimal w. Puis, ils proposent un algorithme *glouton* pour placer les jours de repos avec cet effectif minimal. L'effectif w est égal au maximum des trois bornes suivantes :

– $w \geq \lceil \frac{1}{5} \sum_{j=1}^{7} v_j \rceil$ car la charge couverte par les w employés doit être supérieure à la charge exigée.

- $w \geq \lceil \frac{Bv}{B-A} \rceil,\ v = max(v_6, v_7)$ car sur B semaines, la charge maximale de week-ends couverte par les employés disponibles, $(B-A)w$, doit être supérieure à la charge maximale exigée en week-ends, Bv.
- $w \geq max_j v_j$ car l'effectif doit être suffisant pour couvrir la charge maximale.

Pour assurer un programme avec des séquences de travail de longueurs minimales, il suffit de respecter le même ordre de placement de repos. Si le précédent repos de l'employé i_1 se termine avant celui de l'employé i_2 alors le repos suivant de i_1 commence avant celui de i_2. Dans les deux dernières parties de ce chapitre, nous utiliserons cette règle pour calculer les dates de début des repos.

[C] PPR avec 4 jours de travail par semaine

Les travaux les plus récents sur le PPR, se sont intéressés aux problèmes de placement de jours de repos avec 4 jours de travail par semaine, 10 heures par jour. Alfares [1] a étudié le problème suivant :

$$P_r \begin{cases} \text{Minimiser l'effectif} \\ \text{s.c.} \\ \text{Une charge cyclique } v_j,\ j = 1, \ldots, 7, \text{ donnée.} \\ \text{Chaque employé bénéficie de trois jours de repos par semaine.} \\ \text{Chaque employé bénéficie d'au moins deux jours de repos consécutifs par semaine.} \end{cases}$$

Il a proposé un modèle de couverture d'ensemble pour résoudre le problème. Tout d'abord, il génère les 28 programmes hebdomadaires admissibles. Puis, il calcule l'effectif minimal et introduit cette contrainte pour améliorer le modèle de couverture. Enfin, il résout le modèle augmenté par un algorithme Branch and Bound. Il montre que dans certains cas, selon l'effectif, une solution est retrouvée analytiquement.

7.2.3 Problème de construction de grilles étiquetées (*PGE*)

Le problème *PGE* (Multiple shift workforce scheduling) est une généralisation du problème de placement de repos. Il consiste à affecter les vacations et les jours de repos aux employées tout en respectant la charge, les contraintes de placement de repos et les contraintes d'enchaînement de vacations. La charge représente le nombre d'employés effectuant chacune des vacations par jour. Les vacations peuvent différer par leur nature (perçage, tournage, etc) ou par les dates de début (8h, 9h, etc). Chaque jour, un employé est soit au repos, soit il effectue la même vacation tout au long de la journée.

Hung [9] s'est intéressé au problème suivant :

$$P_e \begin{cases} \text{Minimiser l'effectif} \\ \text{s.c.} \\ p \text{ étiquettes chaque jour, } k = 1, \ldots, p. \\ \text{La charge est } D_k \text{ au cours de la semaine et } E_k \text{ au cours du week-end pour } k. \\ \text{Au maximum cinq jours de travail consécutifs par employé.} \\ \text{Pour tout couple de semaines consécutives, chaque employé reçoit trois jours} \\ \text{de repos dans l'une et quatre jours de repos dans l'autre.} \\ \text{Un employé bénéficie de A week-ends libres sur B semaines.} \\ \text{Une vacation n'est suivie que par elle même ou par un jour de repos.} \end{cases}$$

Hung commence par calculer l'effectif minimal $w = max(\lceil \frac{10D+4E}{7} \rceil, \lceil \frac{BE}{B-A} \rceil)$ avec $D = \sum_{k=1}^{p} D_k$ et $E = \sum_{k=1}^{p} E_k$. Puis, il affecte les jours de repos et les jours de travail aux employés en respectant seulement les contraintes de placement de jours de repos. Ensuite, il affecte les étiquettes aux jours de travail en tenant compte des contraintes d'enchaînement de vacations.

Après ce tour d'horizon des différentes approches développées pour le problème de planification de personnel, nous nous proposons de résoudre un problème *PPR* où nous considérons la présence des employés. La présence d'un employé indique le nombre de jours où il travaille. Les approches précédentes ne peuvent pas être utilisées car la contrainte de la présence n'était pas considérée. Nous allons voir qu'il est intéressant de combiner des modèles de flot et de tomographie discrète pour une résolution optimale polynomiale.

7.3 Planification avec deux ou trois jours de repos par semaine *(P23R)*

7.3.1 Présentation des problèmes

Dans la plupart des cas étudiés, les auteurs s'intéressent à élaborer des emplois du temps qui permettent de minimiser l'effectif sur l'horizon de planification. Ainsi selon les besoins, on cherchait le personnel pouvant les satisfaire.

Notre contribution consiste à établir un emploi du temps nominatif respectant aussi bien la charge que le nombre de jours de travail par employé pour un effectif fixé. Nous considérons différentes contraintes de placement des périodes de repos. La méthode de résolution proposée combine une approche algébrique et des techniques de flot.

Le problème *P23R* consiste à affecter les jours de travail et les jours de repos aux employés sous différentes stratégies de placement de repos. L'effectif est composé de m employés $e_i,\ i = 1, \ldots, m$, ayant les mêmes qualifications. L'horizon de planification est composé de W semaines. v'^k_j employés doivent travailler le jour d_j de la semaine w_k pour $j = 1, \ldots, 7$ et $k = 1, \ldots, W$. Chaque employé e_i travaille au total h'_i jours et a droit à deux jours (2-RC) ou trois jours (3-RC) de repos consécutifs par semaine.

Chaque jour, un employé est soit au travail, soit au repos, il est donc équivalent de considérer h_i le nombre total de jours de repos de l'employé e_i et v^k_j le nombre d'employés

au repos le jour d_j de la semaine w_k. Nous avons $v_j^k = m - v'^k_j$ pour $k = 1, \ldots, W$ et $j = 1, \ldots, 7$ et $h_i = 7W - h'_i$ pour $i = 1, \ldots, m$. Pour qu'il existe une solution, il faut que $\sum_{i=1}^{m} h_i = \sum_{k=1}^{W} \sum_{j=1}^{7} v_j^k$. Nous supposons dans la suite que cette condition est vérifiée.

Nous considérons trois problèmes *P23R* :

1. **Le problème de base** respecte les contraintes suivantes : i) v_j^k employés au repos le jour d_j de la semaine w_k, ii) l'employé e_i prend h_i jours de repos sur tout l'horizon de planification et iii) chaque employé bénéficie d'un 2-RC ou d'un 3-RC par semaine.
2. **Le problème de repos mensuels** respecte les contraintes du problème de base et un employé bénéficie d'au moins une semaine avec un 3-RC par mois.
3. **Le problème de week-ends mensuels** respecte les contraintes du problème de base et tout employé bénéficie d'au moins un week-end libre par mois.

7.3.2 Propriétés des données pour *P23R*

Nous dégageons les propriétés toujours vérifiées par tous les problèmes *P23R* traités. Elles établissent les relations liant les v_j^k, $j = 1, \ldots, 7$, et $k = 1, \ldots, W$, c'est-à-dire, le nombre d'employés au repos par jour. Certaines inéquations sont étiquetées $((a), (b), ...)$ pour s'en servir ultérieurement. Nous supposons que d_1 = lundi est le premier jour de la semaine. Chaque employé bénéficie de deux ou trois jours de repos consécutifs par semaine. On en déduit les propriétés suivantes :

Un employé est au repos une fois par semaine et un repos débutant le jour d_j se termine au plus tard d_{j+2}. D'où aucun employé ne prend simultanément :

- d_1 et (d_4 ou d_5 ou d_6 ou d_7) comme jours de repos donc
  $v_1^k + v_4^k \leq m$, $v_1^k + v_5^k \leq m$ (a), $v_1^k + v_6^k \leq m$ et $v_1^k + v_7^k \leq m$,
- d_2 et (d_5 ou d_6 ou d_7) comme jours de repos donc
  $v_2^k + v_5^k \leq m$ (b), $v_2^k + v_6^k \leq m$ (c) et $v_2^k + v_7^k \leq m$,
- d_3 et (d_6 ou d_7) comme jours de repos donc
  $v_3^k + v_6^k \leq m$ (d) et $v_3^k + v_7^k \leq m$ (e),
- d_4 et d_7 comme jours de repos donc
  $v_4^k + v_7^k \leq m$,
- d_1, d_4 et d_7 comme jours de repos donc
  $v_1^k + v_4^k + v_7^k \leq m$ (f).

Les jours d'un repos (consécutifs) sont inclus dans la même semaine :

- tout employé au repos en d_1 l'est aussi en d_2 donc
  $v_1^k \leq v_2^k$ (g),
- tout employé au repos en d_7 l'est aussi en d_6 donc
  $v_7^k \leq v_6^k$ (h).

Chaque employé bénéficie d'au moins deux jours de repos consécutifs par semaine :

- un jour de repos est suivi ou précédé par un autre jour de repos donc
  $v_2^k \leq v_1^k + v_3^k$ (i), $v_3^k \leq v_2^k + v_4^k$, $v_4^k \leq v_3^k + v_5^k$, $v_5^k \leq v_4^k + v_6^k$ et $v_6^k \leq v_5^k + v_7^k$ (j),
- un employé prend au moins un jour pair et un jour impair de repos donc
  $m \leq v_2^k + v_4^k + v_6^k$ (k) et $m \leq v_1^k + v_3^k + v_5^k + v_7^k$ (l).

Nous dirons que les données sont cohérentes si elles vérifient l'ensemble des propriétés. Nous verrons que les problèmes de planification posés peuvent être vus comme des problèmes de tomographie discrète.

7.3.3 Modèle basé sur la tomographie discrète

La planification de personnel constitue un domaine assez large d'applications de la tomographie discrète. En effet, les séquences de jours de repos s'assimilent aux barres et les séquences de jours de travail s'assimilent aux écarts entre les barres (voir chapitre 6).

Les problèmes *P23R* sont équivalents à des problèmes de packing de dominos et de triminos sur un tableau $m \times 7W$. La ligne i correspond à l'employé e_i et la colonne $col_{j,k}$ correspond au jour d_j de la semaine w_k pour $i = 1, \ldots, m$, $j = 1, \ldots, 7$ et $k = 1, \ldots, W$. Les colonnes sont regroupées par paquet de sept colonnes (semaines) car un repos est inclus dans une semaine. Les dominos et les triminos sont associés respectivement aux 2-RC et 3-RC. Les projections horizontales et verticales sont respectivement h_i et v_j^k. On voudrait reconstruire un packing de dominos et de triminos en imposant que chaque barre (domino ou trimino) couvre seulement des cellules d'un même paquet de colonnes. Un domino (resp. trimino) qui commence sur la ligne i et à la colonne $col_{j,k}$ signifie que l'employé e_i bénéficie d'un 2-RC (resp. 3-RC) qui débute le jour d_j de la semaine w_k.

Après cette description générale, nous étudions les trois problèmes *P23R* définis plus haut.

7.3.4 Résolution du problème de base

Nous proposons une procédure pour résoudre le problème de base tel qu'il a été défini dans la section 7.3. La procédure est composée de trois étapes et donne une solution si et seulement s'il en existe une. A l'étape 1, nous calculons le nombre de 2-RC et 3-RC qui commencent le jour d_j de la semaine w_k. Notons qu'à ce stade, nous ne connaissons pas les employés auxquels ils seront attribués. A l'étape 2, nous décidons pour chaque employé s'il bénéficie d'un 2-RC ou d'un 3-RC au cours de chaque semaine. Constatons qu'à ce stade, les dates de début des repos sont inconnues. Les deux premières étapes sont indépendantes. A l'étape 3, nous combinons les résultats précédents pour affecter les 2-RC et les 3-RC aux employés et calculer leurs dates de début. Un exemple est détaillé à la fin de cette section.

[A] Etape 1 : Calcul du nombre de repos débutant chaque jour

Nous allons déterminer le nombre de 2-RC et 3-RC qui commencent chaque jour. Ce problème est décomposable en W sous-problèmes P_k, $k = 1, \ldots, W$, parce qu'un repos est inclus dans une semaine. Le sous-problème P_k est la restriction du problème de base sur la semaine w_k. Il consiste à calculer le nombre de repos qui commencent chaque jour dans w_k en respectant la demande journalière et en assurant que tout employé bénéficie d'un 2-RC ou d'un 3-RC.

Nous allons nous intéresser à résoudre un sous-problème type, noté P_k. Notons α_j^k et $\beta_j^k,\ j = 1, \ldots, 7$, et $k = 1, \ldots, W$ respectivement le nombre de 2-RC et de 3-RC qui commencent le jour d_j de la semaine w_k.

La résolution du sous-problème P_k revient à calculer les valeurs des variables α_j^k et $\beta_j^k,\ j = 1, \ldots, 7$. $\alpha_7^k,\ \beta_6^k,\ \beta_7^k$ sont nuls car un repos (2-RC ou 3-RC) est inclus dans la même semaine. Pour simplifier la présentation, nous supprimerons l'indice k dans les notations : $\alpha_j,\ \beta_j,\ v_j$ seront utilisés à la place des $\alpha_j^k,\ \beta_j^k,\ v_j^k$.

Le nombre d'employés au repos le jour d_j est égal au nombre de 2-RC débutant d_{j-1} et d_j plus le nombre de 3-RC débutant d_{j-2}, d_{j-1} et d_j. En effet, les 2-RC (resp. 3-RC) qui commencent le jour d_j couvrent aussi d_{j+1} (resp. d_{j+1} et d_{j+2}). Nous obtenons un système linéaire de 8 équations à 11 variables $\alpha_1, \ldots, \alpha_6, \beta_1, \ldots, \beta_5$ qui s'écrit sous la forme :

$$\begin{cases} (1) & v_1 = \alpha_1 + \beta_1 \\ (2) & v_2 = \alpha_1 + \beta_1 + \alpha_2 + \beta_2 \\ (3)\ldots(7) & v_j = \alpha_j + \beta_j + \alpha_{j-1} + \beta_{j-1} + \beta_{j-2},\ j = 3, \ldots, 7, \\ (8) & \sum_{j=1}^{7}(\alpha_j + \beta_j) = \sum_{j=1}^{6} \alpha_j + \sum_{j=1}^{5} \beta_j = m \\ & 0 \leq \alpha_j, \beta_j \leq m \quad j = 1, \ldots, 7 \end{cases}$$

Les équations $(1)\ldots(7)$ assurent qu'il y a v_j employés au repos le jour d_j. L'équation (8) impose m repos hebdomadaires car il y a un 2-RC ou un 3-RC par employé dans chaque semaine.

Les contraintes $\alpha_j, \beta_j \leq m,\ j = 1, \ldots, 7$, sont redondantes car d'après l'équation (8) si toutes les variables sont positives alors elles sont toutes inférieures ou égales à m.

La matrice du système est :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

On vérifie à l'aide du logiciel Scilab que le rang de la matrice est 8. On a alors 8 variables principales qui peuvent être exprimées en fonction de 3 autres, par exemple, $\beta_3, \beta_4, \beta_5$. La résolution du système se ramène au système de Cramer 8×8 suivant :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 - \beta_3 \\ v_4 - \beta_3 - \beta_4 \\ v_5 - \beta_3 - \beta_4 - \beta_5 \\ v_6 - \beta_4 - \beta_5 \\ v_7 - \beta_5 \\ m - \beta_3 - \beta_4 - \beta_5 \end{pmatrix}$$

On vérifie bien qu'il s'agit d'un système de Cramer car le déterminant de sa matrice vaut 1. Pour $\beta_3, \beta_4, \beta_5$ fixées, le système de Cramer ci-dessus est résolu par substitution :

$$\mathcal{S} \begin{cases} \alpha_1 = m - (v_3 + v_5 + v_7) + (\beta_3 + \beta_5) \\ \alpha_2 = m - (v_1 + v_4 + v_6) + \beta_4 \\ \alpha_3 = m - (v_2 + v_5 + v_7) + \beta_5 \\ \alpha_4 = (v_5 - v_6 + v_7) - (\beta_3 + \beta_5) \\ \alpha_5 = v_6 - v_7 - \beta_4 \\ \alpha_6 = v_7 - \beta_5 \\ \beta_1 = -m + (v_1 + v_3 + v_5 + v_7) - (\beta_3 + \beta_5) \\ \beta_2 = -m + (v_2 + v_4 + v_6) - \beta_4 \\ 0 \leq \alpha_j, \beta_j \quad j = 1, \ldots, 7 \end{cases}$$

Déterminons les valeurs admissibles de $\beta_3,\ \beta_4$ et β_5 pour que le système $\mathcal{S}$ soit non vide.

- $\alpha_3 = m - (v_2 + v_5 + v_7) + \beta_5 \geq 0 \ \Leftrightarrow \beta_5 \geq -m + v_2 + v_5 + v_7$.
- $\alpha_6 = v_7 - \beta_5 \geq 0 \Leftrightarrow \beta_5 \leq v_7$. D'où $\alpha_3, \alpha_6 \geq 0 \Leftrightarrow \beta_5 \in [-m + v_2 + v_5 + v_7, v_7] = I1$,
- $\alpha_2 = m - (v_1 + v_4 + v_6) + \beta_4 \geq 0 \Leftrightarrow \beta_4 \geq -m + v_1 + v_4 + v_6$,
- $\alpha_5 = v_6 - v_7 - \beta_4 \geq 0 \Leftrightarrow \beta_4 \leq v_6 - v_7$,
- $\beta_2 = -m + (v_2 + v_4 + v_6) - \beta_4 \geq 0 \Leftrightarrow \beta_4 \leq -m + (v_2 + v_4 + v_6)$. D'où $\alpha_2, \alpha_5, \beta_2 \geq 0 \Leftrightarrow \beta_4 \in [-m + v_1 + v_4 + v_6, min(v_6 - v_7, -m + v_2 + v_4 + v_6)] = I2$,
- $\alpha_1 = m - (v_3 + v_5 + v_7) + (\beta_3 + \beta_5) \geq 0 \Leftrightarrow \beta_3 + \beta_5 \geq -m + (v_3 + v_5 + v_7)$,
- $\alpha_4 = (v_5 - v_6 + v_7) - (\beta_3 + \beta_5) \geq 0 \Leftrightarrow \beta_3 + \beta_5 \leq (v_5 - v_6 + v_7)$,
- $\beta_1 = -m + (v_1 + v_3 + v_5 + v_7) - (\beta_3 + \beta_5) \geq 0 \Leftrightarrow \beta_3 + \beta_5 \leq -m + (v_1 + v_3 + v_5 + v_7)$. D'où $\alpha_1, \alpha_4, \beta_1 \geq 0 \Leftrightarrow \beta_3 + \beta_5 \in [-m + v_3 + v_5 + v_7, min(v_5 - v_6 + v_7, -m + v_1 + v_3 + v_5 + v_7)] = I3$.

Les valeurs de $\beta_3,\ \beta_4$ et β_5 existent si et seulement si

i) les intervalles $I1 \cap [0, m]$, $I2 \cap [0, m]$ et $I3 \cap [0, m]$ ne sont pas vides et permettent d'obtenir des valeurs entières, et

ii) $\beta_5 \in I1 \cap [0, m]$ et $\beta_3 \in [0, m]$ sont compatibles avec $\beta_3 + \beta_5 \in I3 \cap [0, 2m]$.

Notons que les bornes des intervalles sont entières et par conséquent, un intervalle non vide contient au moins une valeur entière.

Proposition 51:
Si les données sont cohérentes alors le système $\mathcal{S}$ admet une solution.

Preuve:
Montrons que les propriétés $(a), \ldots, (l)$ établies dans la section 7.3.2 garantissent que les intervalles $I1, I2$ et $I3$ sont bien définis et que $\mathcal{S}$ admet une solution.
Soit $I = [inf(I), sup(I)]$ un intervalle où sup(I) et inf(I) sont ses bornes. $I \neq \emptyset$ si et seulement si $inf(I) \leq sup(I)$. $I \cap [0, m] \neq \emptyset$ si et seulement si $sup(I) \geq 0$ et $inf(I) \leq m$.

- *$I1$ est non vide car $(b) = (v_2 + v_5 \leq m) \Leftrightarrow -m + v_2 + v_5 + v_7 \leq v_7$. $I1 \cap [0, m] \neq \emptyset$ parce que d'une part, $0 \leq v_7$, et d'autre part, $-m + v_2 + v_5 + v_7 \leq v_7 \leq m$.*
- *$I2$ est non vide car d'une part, $(f) = (v_1 + v_4 + v_7 \leq m) \Leftrightarrow -m + v_1 + v_4 + v_6 \leq v_6 - v_7$, et d'autre part, $(g) = (v_1 \leq v_2) \Leftrightarrow -m + v_1 + v_4 + v_6 \leq -m + v_2 + v_4 + v_6$. $I2 \cap [0, m] \neq \emptyset$ car d'une part, $(h) = (v_6 - v_7 \geq 0), (k) = (-m + v_2 + v_4 + v_6 \geq 0)$, et d'autre part, $-m + v_1 + v_4 + v_6 \leq v_6 - v_7 \leq m$,*
- *$I3$ est non vide car d'une part, $(c) = (v_2 + v_6 \leq m) \Leftrightarrow -m + v_3 + v_5 + v_7 \leq v_5 - v_6 + v_7$, et d'autre part, $-m + v_3 + v_5 + v_7 \leq -m + v_1 + v_3 + v_5 + v_7$. $I3 \cap [0, 2m] \neq \emptyset$ car d'une part, $(j) = (v_5 - v_6 + v_7 \geq 0)$ et $(l) = (-m + v_1 + v_3 + v_5 + v_7 \geq 0)$, et d'autre part, $-m + v_3 + v_5 + v_7 \leq 2m$.*

Soient Ix, Iy et Iz trois intervalles non vides, il existe $x \in Ix$ et $y \in Iy$ avec $x + y \in Iz$ si et seulement si $sup(Ix) + sup(Iy) \geq inf(Iz)$ et $inf(Ix) + inf(Iy) \leq sup(Iz)$. Montrons qu'il existe $\beta_5 \in I1 \cap [0, m]$ et $\beta_3 \in [0, m]$ avec $\beta_3 + \beta_5 \in I3 \cap [0, 2m]$:

- *$sup(I1 \cap [0, m]) + sup([0, m]) \geq inf(I3 \cap [0, 2m])$ parce que d'une part, $v_3 \leq m$ et $v_5 \leq m \Rightarrow (-m + v_3 + v_5 + v_7) \leq v_7 + m$, et d'autre part, $0 \leq v_7 + m$,*
- *$inf(I1 \cap [0, m]) + inf([0, m]) \leq sup(I3 \cap [0, 2m])$ parce que d'une part, $(b) = (v_2 + v_6 \leq m) \Rightarrow -m + v_2 + v_5 + v_7 \leq v_5 + v_7 - v_6 \leq 2m$, et d'autre part, $(a) = (v_1 + v_5 \leq m), (d) = (v_3 + v_7 \leq m)$ et $(i) = (v_2 \leq v_1 + v_3) \Rightarrow -m + v_2 + v_5 + v_7 \leq -m + v_1 + v_3 + v_5 + v_7 \leq 2m$.* ■

Il y a autant de solutions que de valeurs admissibles de $(\beta_3, \beta_4, \beta_5)$ (voir Figure 7.1), c'est-à-dire, une fois que les valeurs de $(\beta_3, \beta_4, \beta_5)$ ont été calculées, le système $\mathcal{S}$ fournit les valeurs correspondant de $\alpha_1, \ldots, \alpha_6, \beta_1$ et β_2.

Déterminons le domaine admissible de β_5 car nous en aurons besoin dans le problème de week-ends mensuels. Notons que $inf(I3) - m \leq 0$ car $(e) = (v_3 + v_7 \leq m)$ et $v_5 \leq m$.

$\beta_3 + \beta_5 \in I3$, $\beta_3 \geq 0$ et $\beta_5 \geq 0$ impliquent que $0 \leq \beta_5 \leq sup(I3)$ car $inf(I3) \leq m$. Donc β_5 appartient à l'intersection des intervalles $[0, sup(I3)]$ et $I1 \cap [0, m]$.

D'où $\beta_5 \in [max(inf(I1), 0), min(sup(I1), sup(I3), m)] = [\beta_{5min}, \beta_{5max}] =$
$[max(0, -m + v_2 + v_5 + v_7), min(v_7, v_5 - v_6 + v_7, -m + v_1 + v_3 + v_5 + v_7)]$.

Finalement, le domaine admissible de β_5 est $[\beta_{5min}, \beta_{5max}]$ et pour tout $x_0 \in [\beta_{5min}, \beta_{5max}]$, il existe au moins solution au problème P_k telle que $\beta_5 = x_0$.

En récapitulatif une solution à P_k est reconstruite comme suit :

- Choisir β_5 dans $[max(0, -m + v_2 + v_5 + v_7), min(v_7, v_5 - v_6 + v_7, -m + v_1 + v_3 + v_5 + v_7)]$.
- Choisir $\beta_3 \in [0, m]$ avec $\beta_3 + \beta_5 \in [-m + v_3 + v_5 + v_7, min(v_5 - v_6 + v_7, -m + v_1 + v_3 + v_5 + v_7)]$.

- Choisir β_4 dans $[-m + v_1 + v_4 + v_6, min(v_6 - v_7, -m + v_2 + v_4 + v_6)]$.
- Pour le triplet $(\beta_3, \beta_4, \beta_3)$ choisi, résoudre le système $\mathcal{S}$ pour calculer les autres variables.

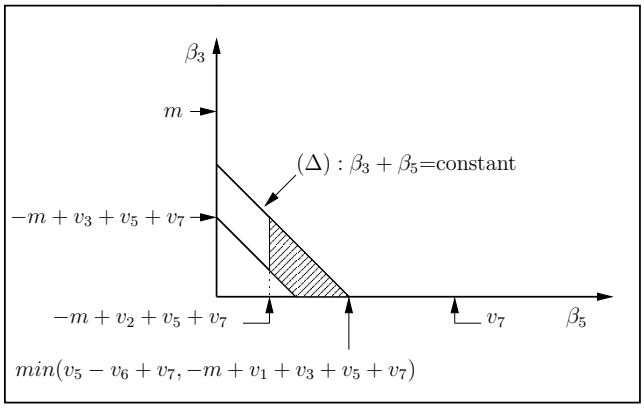


FIG. 7.1 – Domaine admissible de β_3 et β_5

Dans la suite, nous supposons que les sous-problèmes $P_k,\ k = 1, \ldots, W$, ont été résolus.

[B] Etape 2 : Attribution de repos de deux ou trois jours

Dans cette section, nous décidons pour chaque semaine w_k, si l'employé e_i prend un 2-RC ($d\text{–}off_{ik} = 2$) ou un 3-RC ($d\text{–}off_{ik} = 3$). On note par x^k et y^k le nombre d'employés ayant respectivement un 2-RC et un 3-RC au cours de la semaine w_k. On note par c_i le nombre de semaines où l'employé e_i bénéficie d'un 3-RC.

Proposition 52:
Pour que le problème d'attribution de repos de deux ou trois jours admette une solution il faut que :
i) $x^k = 3m - \sum_{j=1}^{7} v_j^k$ *et* $y^k = \sum_{j=1}^{7} v_j^k - 2m,\ k = 1, \ldots, W$,
ii) $c_i = h_i - 2W, i = 1, \ldots, m$.

Preuve:
$2x^k + 3y^k = \sum_{j=1}^{7} v_j^k$ = *le nombre total de jours de repos attribués au cours de la semaine* w_k. $x^k + y^k = m$ *parce que tout employé prend un 2-RC ou un 3-RC par semaine. D'où* $x^k = 3m - \sum_{j=1}^{7} v_j^k$ *et* $y^k = \sum_{j=1}^{7} v_j^k - 2m,\ k = 1, \ldots, W$. $c_i = h_i - 2W, i = 1, \ldots, m$, *car l'employé* e_i *bénéficie d'un 3-RC dans* c_i *semaines et d'un 2-RC dans* $W - c_i$ *semaines. Le nombre total de 3-RC attribués sur l'horizon de planification vaut* $\sum_{k=1}^{W} y^k = \sum_{i=1}^{m} c_i$. ■

$x^k \geq 0$ et $y^k \geq 0$ impliquent que $2m \leq \sum_{j=1}^{7} v_j^k \leq 3m,\ k = 1, \ldots, W$, est une condition nécessaire d'existence d'une solution au problème d'attribution de repos. Nous supposons dans la suite que cette condition est vérifiée.

Cette étape est résolue à l'aide de la recherche d'un flot maximal dans un graphe biparti augmenté $G = (\mathcal{E}, \mathcal{W}, \mathcal{A})$ (voir Figure 7.2). $\mathcal{E} = \{e_i,\ i = 1, \ldots, m\}$ représente

l'ensemble des employés et $\mathcal{W} = \{w_k,\ k = 1, \ldots, W\}$ représente l'ensemble des semaines. Les arcs (e_i, w_k) de $\mathcal{A}$ ont une capacité unitaire. Ajoutons à G une source S et un arc de S à chaque sommet e_i de $\mathcal{E}$. Ajoutons également un puits T et un arc de chaque sommet w_k de $\mathcal{W}$ à T. Les capacités des arcs (S, e_i) sont égales à $c_i,\ i = 1, \ldots, m$, et les capacités des arcs (w_k, T) sont égales à $y^k,\ k = 1, \ldots, W$.

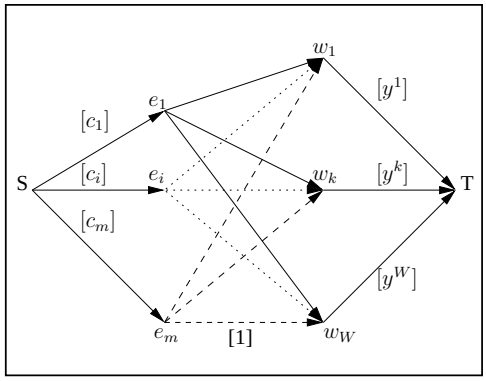


FIG. 7.2 – Attribution des 2-RC et 3-RC

Proposition 53:
Le problème d'attribution de repos de deux ou trois jours admet une solution si et seulement si la valeur du flot maximal dans G est $\sum_{k=1}^{W} y^k = \sum_{i=1}^{m} c_i$.

Preuve:
Il est évident que la résolution du problème d'attribution de repos de deux ou trois jours est équivalent à la recherche d'un flot maximal dans G. L'employé e_i prend un 3-RC dans le semaine w_k $(d-off_{ik} = 3)$ si et seulement si le flot sur l'arc (e_i, w_k) vaut 1. Un flot maximal de valeur $\sum_{i=1}^{m} c_i$ assure que tous les arcs (S, e_i) sont saturés, c'est-à-dire que chaque employé e_i bénéficie de c_i 3-RC. Un flot maximal de valeur $\sum_{k=1}^{W} y^k$ assure que tous les arcs (w_k, T) sont saturés, c'est-à-dire qu'au cours de chaque semaine w_k, y^k 3-RC sont attribués aux employés.

Notons que le problème de recherche d'un flot maximal dans G est équivalent au problème de reconstruction d'une matrice binaire à partir de ses projections orthogonales (voir section 1.2). On peut donc calculer un flot maximal dans G à l'aide d'un algorithme spécifique en temps $O(mW + \max(m \log m, W \log W))$. ■

Rappelons que les étapes 1 et 2 sont résolues indépendemment l'une de l'autre.

[C] Etape 3 : Reconstruction d'une solution du problème de base

Nous supposons que les sous-problèmes $P_k,\ k = 1, \ldots, W$, et le problème d'attribution de repos ont été résolus respectivement dans les étapes 1 et 2. Ainsi, d'une part, on connaît le nombre de 2-RC et 3-RC qui débutent par jour, et d'autre part, on a décidé pour chaque employé s'il prend un 2-RC ou un 3-RC $(d-off_{ik} \in \{2, 3\}\ i = 1, \ldots, m, k = 1, \ldots, W)$ au cours de chaque semaine. Toute attribution arbitraire de dates de début de repos

aux employés qui vérifie les résultats des deux premières étapes est admissible. En effet, pour toute semaine, le nombre d'employés ayant un 2-RC (resp. 3-RC) est égal à la somme de 2-RC (resp. 3-RC) qui débutent au cours de cette semaine car $\sum_{j=1}^{6} \alpha_j^k = x^k$ et $\sum_{j=1}^{6} \beta_j^k = y^k$.

Pour limiter la durée des séquences de travail, c'est-à-dire, le nombre de jours de travail consécutifs, nous attribuons les dates de début de repos selon la règle suivante : si le repos de l'employé e_i se termine après celui de l'employé e_j au cours de la semaine w_{k-1} et si e_i et e_j prennent le même nombre de jours de repos au cours de w_k alors le repos de e_i commence après celui de e_j au cours de la semaine w_k. Une solution est alors obtenue par la procédure suivante :

Procédure dates-de-repos

Données : $\{\alpha_j^k, \beta_j^k,\ j = 1, \ldots, 7, k = 1, \ldots, W\}$ (*Calculés par l'étape 1*) ;
$\{d{-}off_{ik},\ i = 1, \ldots, m, k = 1, \ldots, W\}$ (*Calculés par l'étape 2*) ;
pour $k = 1$ **à** W **faire**
1. $X_k \leftarrow \{e_i$ tel que $d{-}off_{ik} = 2\}$;
// employés ayant un 2-RC au cours de la semaine w_k ;
2. $Y_k \leftarrow \{e_i$ tel que $d{-}off_{ik} = 3\}$;
//employés ayant un 3-RC au cours de la semaine w_k ;
// $X_k \cap Y_k = \emptyset$ *et* $|X_k| + |Y_k| = m$
3. trier X_k et Y_k selon la date de fin de repos au cours de w_{k-1} ;
//X_1 *et* Y_1 *sont triés selon les indices* ;
4. pour $j = 1$ **à** 6 **faire**
4.1 attribuer un 2-RC débutant le jour d_j de w_k aux premiers α_j^k employés de X_k et les enlever de X_k ;
4.2 attribuer un 3-RC débutant le jour d_j de w_k aux premiers β_j^k employés de Y_k et les enlever de Y_k ;
fin faire ;
fin faire ;

Concernant l'existence d'une solution au problème de base, nous énonçons le résultat suivant :

Proposition 54:
Le problème de base admet une solution si et seulement si la première et la deuxième étapes admettent des solutions.

Preuve:
La condition est nécessaire car toute solution au problème de base est une solution aux étapes 1 et 2. La condition est suffisante car d'après la troisième étape, toutes solutions aux étapes 1 et 2 peuvent être combinées pour reconstruire une solution au problème de base.

Proposition 55:
La complexité du problème de base est $O(mW + max(m \log m, W \log W))$.

Preuve:
La première étape est résolue en temps $\Theta(W)$ *parce qu'elle consiste à résoudre* W *problèmes* P_k *et chaque* P_k *est résolu en temps constant. En utilisant un algorithme spécifique de reconstruction de matrices binaires (voir section 1.2), la deuxième étape est résolue en temps* $O(mW + \max(m \log m, W \log W))$. *La troisième étape est de complexité* $O(mW)$ *parce qu'au cours de chaque semaine, il y a* m *repos à attribuer aux employés.*

[D] Un exemple

Afin de bien comprendre et assimiler l'approche proposée, nous reprenons, à l'aide d'un exemple, en détail chacune des trois étapes. Considérons une compagnie avec cinq employés, $m = 5$, et un horizon de planification de trois semaines, $W = 3$. Le nombre total de jours de travail par employé et la demande en personnel sont donnés respectivement par $H' = (12, 13, 15, 14, 13)$ et $V' = (4, 3, 2, 3, 4, 3, 3; 4, 4, 3, 3, 2, 3, 3; 4, 4, 4, 3, 2, 2, 4)$. Nous en déduisons le nombre total de jours de repos par employé : $H = (9, 8, 6, 7, 8)$ et le nombre d'employés au repos par jour : $V = (1, 2, 3, 2, 1, 2, 2; 1, 1, 2, 2, 3, 2, 2; 1, 1, 1, 2, 3, 3, 1)$ car $h_i = 21 - h'_i$ et $v_j^k = 5 - v_j'^k$.

Etape 1 : Détaillons la première étape pour le sous-problème P_1. $\beta_5 \in I1 = [0, 2], I2 = [0, 1]$ et $\beta_3 + \beta_5 \in I3 = [1, 1]$. Nous obtenons $\beta_4 = 0$ ou 1 et $(\beta_3, \beta_5) = (1, 0)$ ou $(\beta_3, \beta_5) = (0, 1)$. Le sous-problème P_1 admet quatre solutions selon les valeurs de $(\beta_3, \beta_4, \beta_5)$. Prenons arbitrairement $(\beta_3, \beta_4, \beta_5) = (1, 0, 0)$ et calculons grâce au système $\mathcal{S}$ les autres variables : $\alpha_1 = \alpha_2 = \cdots = \alpha_5 = 0, \alpha_6 = 2, \beta_1 = \beta_2 = 1$. Le résultat complet de la première étape est donné par le tableau 7.2. Les colonnes représentent les jours de l'horizon de planification (trois semaines). La deuxième et la troisième lignes donnent respectivement le nombre de 2-RC et de 3-RC qui commencent par jour.

Jour	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
2-jours de repos	0	0	0	0	0	2	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0
3-jours de repos	1	1	1	0	0	0	0	0	0	1	0	2	0	0	0	0	0	1	1	0	0

TAB. 7.2 – Etape 1

Etape 2 : Calculons, tout d'abord, $y^1 = 3, y^2 = 3, y^3 = 2$ ($y^k = \sum_{j=1}^{7} v_j^k - 2m$) et $c_1 = 3, c_2 = 2, c_3 = 0, c_4 = 1, c_5 = 2$ ($c_i = h_i - 2W$). Ensuite, cherchons un flot maximal dans le graphe de la figure 7.3. Le tableau 7.3 donne le nombre de jours de repos par employé et par semaine.

Etape 3 : Au cours de chaque semaine w_k, $k = 1, 2, 3$, on affecte $\sum_{j=1}^{6} \beta_j^k$ repos 3-RC aux y^k employés ayant un 3-RC et $\sum_{j=1}^{6} \alpha_j^k$ repos 2-RC aux x^k employés ayant un 2-RC. Les dates de début de repos sont attribués selon la règle de la procédure date-de-repos (voir Figure 7.4). Notons que le repos de e_2 dans la semaine w_2 débute avant celui de e_3

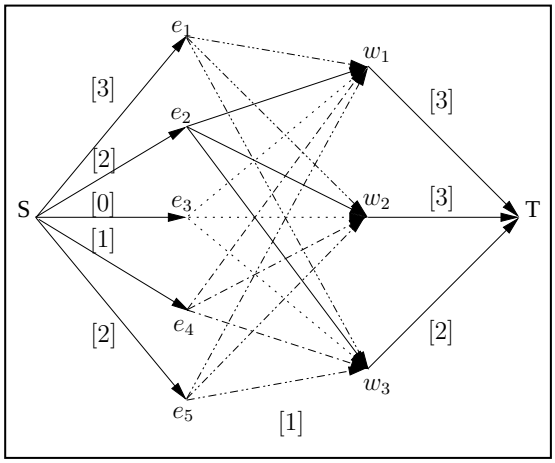


FIG. 7.3 – Etape 2 : graphe biparti

Emp.	**Semaine 1**	**Semaine 2**	**Semaine 3**
1	3	3	3
2	3	2	3
3	2	2	2
4	2	3	2
5	3	3	2

TAB. 7.3 – Etape 2 : nombre de jours de repos par semaine

parce que son repos au cours de la semaine w_1 se termine avant celui de e_3.

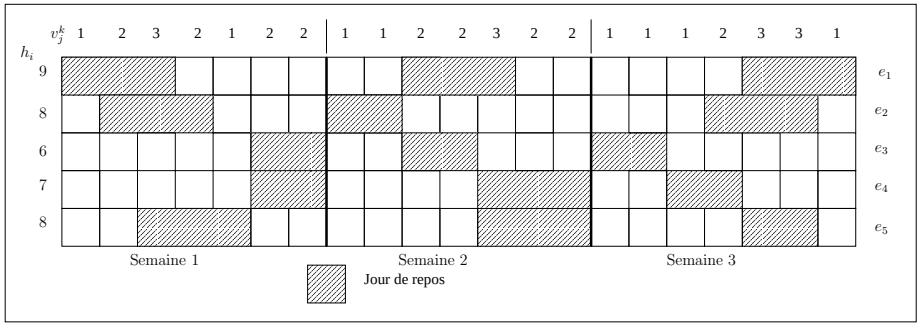


FIG. 7.4 – Etape 3 : solution au problème de base

En conclusion, le problème *P23R* tel qu'il a été défini est résolu en temps polynomial. Sa polynomialité est due au fait que les sous-problèmes P_k sont indépendants. La procédure dates-de-repos permet d'obtenir une solution si et seulement s'il en existe une. Si l'on n'impose pas que les repos soient consécutifs, la complexité du problème devient inconnue car les propriétés du problème *P23R* ne sont plus valides. Nous verrons également que si l'on impose d'autres contraintes sur le placement des repos, le problème restera polynomial dans certains cas et sera de complexité inconnue dans d'autres cas.

En supposant que le lundi est le premier jour de la semaine, nous excluons les séquences

suivantes de jours de repos : samedi-dimanche-lundi, dimanche-lundi et dimanche-lundi-mardi. Ceci n'est pas très restrictif et pourrait être évité en supposant que le vendredi est le premier jour de la semaine. Ceci entraînerait qu'il y ait plus d'employés au repos le samedi que le vendredi et plus d'employés au repos le mercredi que le jeudi. Ce qui est vrai dans de nombreuses entreprises françaises.

7.3.5 Problème de repos mensuels

On cherche une solution au problème de base telle que chaque employé bénéficie d'au moins une semaine comportant un 3-RC par mois. L'horizon de planification est composé de M mois dont chacun est constitué de quatre semaines. Nous proposons une procédure polynomiale en trois étapes pour résoudre ce problème. La première étape est similaire à celle du problème de base car on cherche juste à calculer le nombre de repos qui commencent par jour.

La deuxième étape consiste à décider pour chaque employé s'il prend un 2-RC ou un 3-RC au cours de chaque semaine w_k $(d-off_{ik} = 2\,ou\,3,\ i = 1,\dots,m, k = 1,\dots,W)$ en respectant la contrainte d'au moins un 3-RC par mois. Pour répartir les 3-RC entre les semaines, nous cherchons un flot maximal avec des bornes inférieures dans un graphe triparti $G = (\mathcal{E}, \mathcal{ME}, \mathcal{W}, \mathcal{A})$ (voir Figure 7.5). $\mathcal{E} = \{e_i,\ i = 1,\dots,m\}$ est l'ensemble des employés et $\mathcal{W} = \{w_k,\ k = 1,\dots,W\}$ est l'ensemble des semaines. A chaque employé e_i, associons M sommets Me_{il} correspondant aux mois $l = 1,\dots,M$. Ajoutons une source S avec un arc de capacité c_i de S à e_i, $i = 1,\dots,m$, et un puits T avec un arc de capacité y^k de w_k à T, $k = 1,\dots,W$. Ajoutons un arc de capacité $(1,4)$ de e_i à Me_{il} pour $i = 1,\dots,m$ et $l = 1,\dots,M$ **pour assurer qu'au moins un 3-RC est attribué à l'employé** e_i **au cours du mois** l. Ajoutons également un arc de capacité unitaire de Me_{il} à $w_{4l-3}, w_{4l-2}, w_{4l-1}, w_{4l}$ pour $l = 1,\dots,M$, c'est-à-dire, 4 arcs de chaque mois vers les 4 semaines qui le composent. Comme précédemment, le problème d'attribution de 2-RC et 3-RC admet une solution si et seulement si la valeur du flot maximal dans G est égale à $\sum_{k=1}^{W} y^k = \sum_{i=1}^{m} c_i$.

Ce modèle peut se généraliser en remplaçant un mois, séquence de quatre semaines, par n'importe quel ensemble de semaines, consécutives ou non et de cardinalités différentes pour chaque ensemble. L'arc (e_i, Me_{il}) corrspond à un ensemble de z semaines et sa capacité est égale à $(1, z)$. Il y a z arcs de capacité unitaire de Me_{il} vers chacune de ces semaines. L'intérêt pratique de cette extension est de permettre aux employés de prendre au moins un 3-RC pendant une période de vacances scolaires. Dans ce cas, on regroupe l'ensemble des vacances scolaires en "mois".

Une fois les deux premières étapes résolues, on se sert de la procédure dates-de-repos pour trouver une solution complète (étape 3). On démontre de même que le problème de repos mensuels admet une solution si et seulement si les étapes 1 et 2 admettent des solutions.

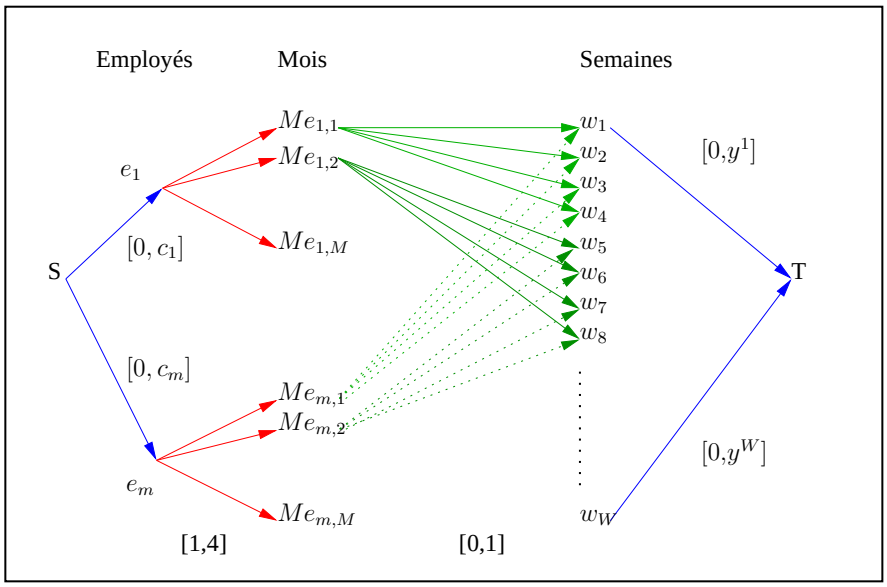


FIG. 7.5 – Problème de repos mensuels

7.3.6 Problème de week-ends mensuels

Au problème de base, on ajoute la contrainte suivante : tout employé bénéficie d'au moins un week-end (suite samedi-dimanche) libre par mois. Cela impose une contrainte supplémentaire reliant les quatre semaines d'un même mois. Nous ne connaissons pas la complexité de ce problème. La décomposition en trois étapes n'est plus valable car les problèmes P_k ne sont plus indépendants.

Cependant, considérons le problème de base (sans la contrainte de week-ends mensuels) et calculons une solution à l'étape 2. Nous pouvons alors tester s'il existe une solution trouvée par l'étape 1 qui vérifie cette nouvelle contrainte. En cas de test positif, nous aurons résolu le problème mais en cas d'échec il nous sera impossible de conclure.

Pour cela, nous cherchons les valeurs admissibles des β_5^k dans $[\beta_{5min}^k, \beta_{5max}^k]$ telles que chaque employé prenne au moins un week-end libre par mois. Pour chaque mois, le calcul des β_5^k se fait à travers la recherche d'un flot compatible dans un graphe triparti $G = (\mathcal{E}, 2\text{–}3\mathcal{W}, \mathcal{D}, \mathcal{A})$ (voir Figure 7.6). $\mathcal{E}$ représente l'ensemble des employés. $2\text{–}3\mathcal{W}$ représente l'ensemble de week-ends libres de 2-RC et de 3-RC. $\mathcal{D}$ représente les quatre dimanches du mois. Ajoutons à G une source S et un puits T.

Rappelons que nous savons pour tout employé s'il prend un 2-RC ou un 3-RC pendant chaque semaine. Pour cela, si l'employé e_i prend un 2-RC (resp. un 3-RC), alors le sommet e_i est relié à tout sommet $2w_k$ (resp. $3w_k$) par un arc de capacité $(0, 1)$. Dans la figure 7.6, l'employé e_1 prend un 2-RC dans la deuxième et la dernière semaines et un 3-RC dans la première et la troisième semaines. La source S est reliée à chaque sommet e_i par un arc de capacité $(1, 4)$ pour assurer au moins un week-end libre pour e_i.

Un sommet $3w_k$ est relié au sommet D_k par un arc de capacité $(\beta_{5min}^k, \beta_{5max}^k)$ car Le domaine admissible de β_5^k est $[\beta_{5min}^k, \beta_{5max}^k]$. Un sommet D_k est relié par un arc de capacité (v_7^k, v_7^k) au puits T dont le flot donne le nombre d'employés au repos le dimanche

de la semaine w_k. On en déduit que chaque sommet $2w_k$ devrait être relié au sommet D_k par un arc de capacité $(0, \infty)$ dont le flot donne la valeur de α_6^k, c'est-à-dire, le nombre d'employés ayant un 2-RC en week-end puisque d'après le système $\mathcal{S}$, $\alpha_6^k + \beta_5^k = v_7^k$.

Le test a une réponse positive, c'est-à-dire, un employé peut bénéficier d'un week-end libre au cours du mois testé, si et seulement si la valeur du flot maximal dans G est $\sum_{k=1}^{4} v_7^k$.

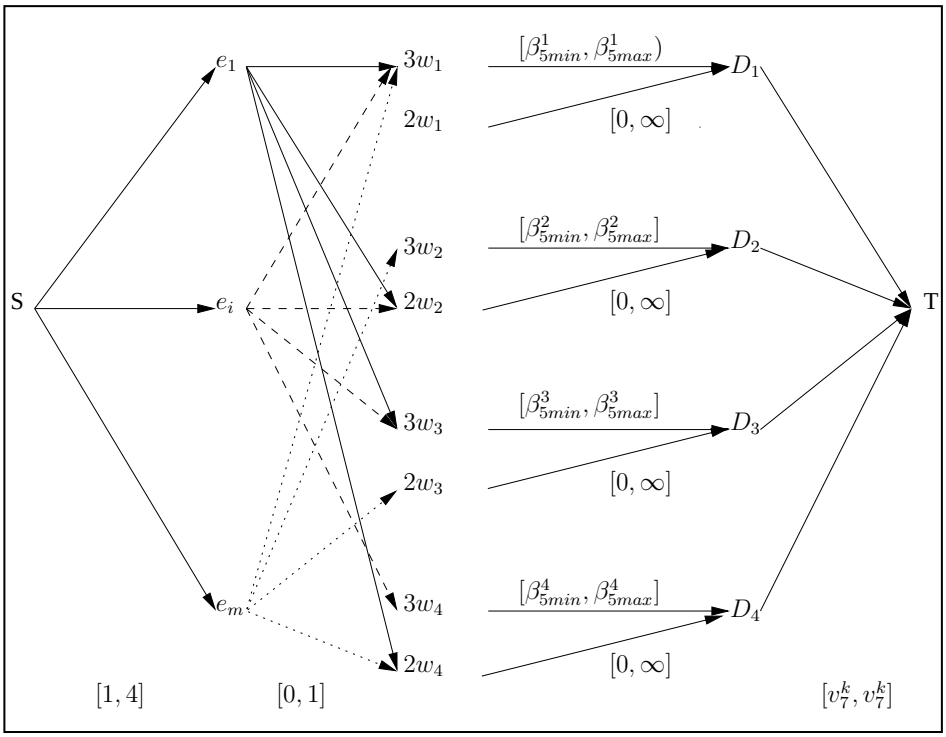


FIG. 7.6 – Test d'existence d'un week-end libre par mois

En récapitulatif, pour une attribution donnée de jours de repos, le test d'existence d'une solution et la résolution du problème de week-ends mensuels se font de la manière suivante :

1. Chercher un flot compatible dans $G = (\mathcal{E}, 2-3\mathcal{W}, \mathcal{D}, \mathcal{A})$ pour chaque mois. Si pour un mois donné, le test a une réponse négative alors l'attribution de repos n'est pas bonne. On pourrait envisager une stratégie pour changer l'attribution,
2. résoudre les P_k. Les valeurs des β_5^k sont les valeurs du flot compatible sur les arcs $(3w_k, D_k)$ du graphe $G = (\mathcal{E}, 2-3\mathcal{W}, \mathcal{D}, \mathcal{A})$ correspondant,
3. si le flot maximal sur l'arc $(e_i, 3w_k)$ (resp. $(e_i, 2w_k)$) est de valeur 1 alors e_i prend un 3-RC (resp. 2-RC) en week-end de w_k,
4. pour les autres employés, calculer les dates de début de repos en suivant la règle établie par la procédure dates-de-repos.

Nous montrons que sous certaines conditions, le problème de week-ends mensuels est polynomial.

Etude d'un cas particulier polynomial

Nous considérons un problème dont les données vérifient les inéquations suivantes :

$$\boxed{m \geq v_2^k + v_5^k + v_7^k,\ v_5^k \geq v_6^k \text{ et } v_1^k + v_3^k + v_5^k \geq m \qquad \mathcal{C}1}$$

C'est le cas particulier où $\beta_{5min}^k = 0$ et $\beta_{5max}^k = v_7^k$ pour tout k.

Proposition 56:
Si la condition $\mathcal{C}1$ est vérifiée alors le problème de week-ends mensuels admet une solution si et seulement si :

i) le problème de base admet une solution,

ii) $m \leq v_7^{4l-3} + v_7^{4l-2} + v_7^{4l-1} + v_7^{4l}$ pour tout mois l.

Preuve:
La condition i) est nécessaire parce que toute solution du problème de week-ends mensuels est aussi une solution au problème de base. La condition ii) est également nécessaire car $v_7^{4l-3} + v_7^{4l-2} + v_7^{4l-1} + v_7^{4l}$ représente le nombre de repos le dimanche pendant le mois l. Ce nombre doit être supérieur au nombre d'employés pour que chacun d'entre eux puisse obtenir au mois un week-end libre.

Réciproquement, soit S une solution au problème de base qui ne respecte pas la contrainte d'au moins un week-end libre par employé : il existe un employé e_a n'ayant pas un week-end libre pendant le mois l. La condition ii) implique qu'il existe un employé e_b qui prend au moins deux week-ends libres pendant ce mois. Soit w_k une semaine du moins l au cours de laquelle e_b prend un week-end libre. Nous montrons qu'il est possible que e_a prenne un week-end libre au cours de w_k à la place de e_b. Rappelons que β_5^k et α_6^k représentent le nombre d'employés ayant respectivement un 3-RC et un 2-RC en week-end de la semaine w_k. L'idée de la preuve est de modifier la valeur de β_5^k dans son domaine admissible $[\beta_{5min}^k, \beta_{5max}^k]$. Trois cas se présentent selon les repos de e_a et e_b dans w_k :

1. *e_a **et** e_b **ont le même nombre de jours de repos dans** w_k : il suffit de permuter les jours de repos attribués à e_a et ceux attribués à e_b dans w_k.*
2. *e_a **prend un 2-RC et** e_b **prend un 3-RC dans** w_k. On a $\beta_5^k > 0$ car e_b prend un 3-RC. Tout d'abord, on incrémente α_6^k de 1, c'est-à-dire, on augmente le nombre de 2-RC en week-end et par la suite, on décrémente β_5^k de 1 ($\hat{\beta}_5^k = \beta_5^k - 1$) car $\alpha_6^k + \beta_5^k = v_7^k$. Puis, on résout le problème $\hat{P}_k$ avec $\hat{\beta}_5^k$. $\hat{\beta}_5^k$ est admissible car $\beta_5^k > 0$ et donc $\hat{P}_k$ admet à nouveau une solution avec $\hat{\beta}_5^k = \beta_5^k - 1$ employés ayant un 3-RC en week-ends et $\alpha_6^k + 1$ employés ayant un 2-RC en week-end. $\hat{S}$ est obtenue à partir de S, en réaffectant les dates de repos au cours de w_k de la manière suivante :*
   - *e_b prend un 3-RC au cours de la semaine au lieu du week-end libre.*
   - *e_a prend un 2-RC en week-end au lieu du 2-RC au cours de la semaine.*
   - *Tout employé, à l'exception de e_b, garde son éventuel week-end libre.*
   - *Les employés n'ayant pas un week-end libre gardent le même nombre de jours de repos sauf que les dates de début des repos peuvent changer car β_j^k et α_j^k ont été modifiés.*

3. e_a ***prend un 3-RC et*** e_b ***prend un 2-RC dans*** w_k. *On a* $\beta_5^k < v_7^k$ *car* e_b *prend un 2-RC en week-end. On utilise la même technique que dans le cas précédent en incrémentant* β_5^k *de* 1 *et en décrémentant* α_6^k *de* 1.

Nous réitérons cet opération jusqu'à ce que chaque employé prenne au moins un week-end libre par mois. ■

On conclut que si $\beta_{5min}^k = 0$ et $\beta_{5max}^k = v_7^k$ pour tout k alors toute solution au problème de base se transforme polynomialement en une solution au problème de week-ends mensuels.

Remarquons que ce raisonnement n'est pas valable dans le cas général car si $\beta_5^k = \beta_{5min}^k > 0$ où $\beta_5^k = \beta_{5max}^k < v_7^k$ alors les opérations de décrémentations et d'incrémentations de β_5^k ne sont plus valides.

7.4 Planification avec trois ou quatre jours de repos par semaine *(P34R)*

Ce problème est une généralisation du problème *P23R* où chaque employé prend trois (3-RC) ou quatre (4-RC) jours de repos consécutifs par semaine. On peut traiter d'une manière équivalente les problèmes *P45R* et *P56R*. Le principe est toujours le même : dans une première étape, on calcule le nombre de repos débutant chaque jour. Ensuite, on affecte les jours de repos aux employés à l'aide de modèles de flot. Enfin, on combine les résultats précédents pour calculer la date de début de chaque repos.

Comme dans le cas du problème *P23R*, nous étudions trois sous-problèmes :

1. **Le problème de base étendu** respecte les contraintes suivantes : i) v_j^k employés au repos le jour d_j de la semaine w_k, ii) l'employé e_i prend h_i jours de repos sur tout l'horizon de planification et iii) chaque employé bénéficie d'un 3-RC ou d'un 4-RC par semaine.
2. **Le problème de repos mensuels étendu** respecte les contraintes du problème de base étendu et chaque employé bénéficie d'au moins une semaine avec un 4-RC par mois.
3. **Le problème de week-ends mensuels étendu** respecte les contraintes du problème de base étendu et chaque employé bénéficie d'au moins un week-end libre par mois.

Les différentes notations et variables pour les problèmes *P23R* et *P34R* sont dressées dans le tableau 7.4.

7.4.1 Propriétés des données pour *P34R*

Nous dégageons les propriétés des problèmes *P34R*. Nous supposons à nouveau $d_1 =$ lundi est le premier jour de la semaine.

Un employé est au repos une fois par semaine et un repos débutant le jour d_j se termine au plus tard le jour d_{j+3}. D'où aucun employé ne prend simultanément :

Problème	*P23R*	*P34R*
2-RC	2 jours de repos consécutifs	-
3-RC	3 jours de repos consécutifs	3 jours de repos consécutifs
4-RC	-	4 jours de repos consécutifs
x^k	nombre de 2-RC dans w_k	nombre de 3-RC dans w_k
y^k	nombre de 3-RC dans w_k	nombre de 4-RC dans w_k
c_i	nombre de 3-RC pour e_i	nombre de 4-RC pour e_i
α_j^k	nombre de 2-RC commençant d_j	nombre de 3-RC commençant d_j
β_j^k	nombre de 3-RC commençant d_j	nombre de 4-RC commençant d_j

TAB. 7.4 – Variables des problèmes *P23R* et *P34R*

– d_1 et (d_5 ou d_6 ou d_7) comme jours de repos donc
$v_1^k + v_5^k \leq m$ (a4), $v_1^k + v_6^k \leq m$ et $v_1^k + v_7^k \leq m$,
– d_2 et (d_6 ou d_7) comme jours de repos donc
$v_2^k + v_6^k \leq m$ (b4) et $v_2^k + v_7^k \leq m$,
– d_3 et d_7 comme jours de repos donc
$v_3^k + v_7^k \leq m$ (c4),

Chaque employé bénéficie d'au moins trois jours de repos consécutifs par semaine :
– un employé est au repos en d_2 ou d_5 de chaque semaine donc
$m \leq v_2^k + v_5^k$ (d4),
– un employé est au repos en d_3 ou d_6 de chaque semaine donc
$m \leq v_3^k + v_6^k$ (e4),
– un employé est au repos en d_1 ou d_4 ou d_7 de chaque semaine donc
$m \leq v_1^k + v_4^k + v_7^k$ (f4),

Les données sont dites cohérentes si elles vérifient l'ensemble des propriétés.

7.4.2 Problème de base étendu

Nous proposons une procédure polynomiale composée de trois étapes pour résoudre le problème de base étendu. La première étape consiste à calculer le nombre de 3-RC et 4-RC qui commencent le jour d_j de la semaine w_k. La deuxième étape permet de décider pour chaque employé s'il bénéficie d'un 3-RC ou d'un 4-RC au cours de chaque semaine. La dernière étape combine les deux premières étapes pour affecter les 3-RC et les 4-RC aux employés et calculer leurs dates de début.

[A] Etape 1 : Calcul du nombre de repos débutant par jour

Cette étape consiste à déterminer le nombre de 3-RC et 4-RC qui commencent chaque jour. Comme pour les problèmes *P23R*, nous décomposons cette étape en W sous-problèmes $P_k,\ k = 1, \ldots, W$. Le sous-problème P_k consiste, cette fois-ci, à calculer le nombre de repos qui commencent chaque jour de la semaine w_k en satisfaisant la demande tel qu'un employé bénéficie d'un 3-RC ou d'un 4-RC.

Les variables α_j^k et $\beta_j^k,\ j = 1, \ldots, 7$, et $k = 1, \ldots, W$, représentent, dans le cas présent, respectivement le nombre de 3-RC et 4-RC qui commencent le jour d_j de la semaine w_k. Les variables $\alpha_6^k, \alpha_7^k,\ \beta_5^k,\ \beta_6^k,\ \beta_7^k$ sont nulles car un repos (3-RC ou 4-RC) est inclus dans la même semaine.

Tous les sous-problèmes P_k sont résolus de la même façon. Pour simplifier la présentation, nous supprimons l'indice k dans les notations et nous résolvons un sous-problème type.

Les contraintes de la satisfaction de la demande journalière donnent :

$$
\begin{array}{ll}
(1') & v_1 = \alpha_1 + \beta_1 \\
(2') & v_2 = \alpha_1 + \beta_1 + \alpha_2 + \beta_2 \\
(3') & v_3 = \alpha_1 + \beta_1 + \alpha_2 + \beta_2 + \alpha_3 + \beta_3 \\
(4') & v_4 = \beta_1 + \alpha_2 + \beta_2 + \alpha_3 + \beta_3 + \alpha_4 + \beta_4 \\
(5') & v_5 = \beta_2 + \alpha_3 + \beta_3 + \alpha_4 + \beta_4 + \alpha_5 \\
(6') & v_6 = \beta_3 + \alpha_4 + \beta_4 + \alpha_5 \\
(7') & v_7 = \beta_4 + \alpha_5 \\
(8') & \sum_{j=1}^{7}(\alpha_j + \beta_j) = \sum_{j=1}^{6}\alpha_j + \sum_{j=1}^{5}\beta_j = m \\
 & 0 \leq \alpha_j, \beta_j \leq m \quad j = 1, \ldots, 7
\end{array}
$$

Les équations $(1') \ldots (7')$ assurent qu'il y a v_j employés au repos le jour d_j. L'équation $(8')$ garantit que m repos hebdomadaires sont attribués aux employés par semaine. Les contraintes $\alpha_j, \beta_j \leq m,\ j = 1, \ldots, 7$, sont redondantes car d'après l'équation $(8')$ si toutes les variables sont positives alors elles sont toutes inférieures ou égales à m.

Le système établi est de 8 équations linéaires à 9 variables $\alpha_1, \ldots, \alpha_5, \beta_1, \ldots, \beta_4$. On vérifie à l'aide de Scilab que son rang est 8. Ce système est résolu en exprimant toutes les variables en fonction de β_1 :

$$
\mathcal{S}' \begin{cases}
\alpha_1 = v_1 - \beta_1 \\
\alpha_2 = m - v_1 - v_5 \\
\alpha_3 = m - v_2 - v_6 \\
\alpha_4 = m - v_3 - v_7 \\
\alpha_5 = m - v_4 - v_1 + \beta_1 \\
\beta_2 = v_5 + v_2 - m \\
\beta_3 = v_6 + v_3 - m \\
\beta_4 = v_7 + v_4 + v_1 - \beta_1 - m \\
0 \leq \alpha_j, \beta_j \quad j = 1, \ldots, 7
\end{cases}
$$

Nous montrerons que sous la satisfaction des propriétés des données (voir section 7.4.1), le système S' est non vide.

Proposition 57:
Si les données sont cohérentes alors le système S' admet une solution.

Preuve:
Nous déterminons les valeurs admissibles de β_1 qui assurent que toutes les variables sont

positives. Supposons que β_1 est positive et vérifions que toutes les autres variables le sont :

- $0 \leq \alpha_1 \Leftrightarrow \beta_1 \leq v_1$,
- *(a4)=* $(v_1^k + v_5^k \leq m) \Leftrightarrow 0 \leq \alpha_2$,
- *(b4)=*$(v_2^k + v_6^k \leq m) \Leftrightarrow 0 \leq \alpha_3$,
- *(c4)=*$(v_3^k + v_7^k \leq m) \Leftrightarrow 0 \leq \alpha_4$,
- *(d4)=*$(m \leq v_2^k + v_5^k) \Leftrightarrow 0 \leq \beta_2$,
- *(e4)=*$(m \leq v_3^k + v_6^k) \Leftrightarrow 0 \leq \beta_3$,
- $\alpha_5 \geq 0 \Leftrightarrow \beta_1 \geq v_4+v_1-m$. $\beta_4 \geq 0 \Leftrightarrow \beta_1 \leq v_7+v_4+v_1-m$. *D'où* $\alpha_5, \beta_4 \geq 0$ *est vérifié pour toute valeur de* β_1 *dans* $[max(0, v_4+v_1-m), min(v_1, v_7+v_4+v_1-m)] = I$. *I est non vide car d'une part,* $v_1 \geq 0$ *et* $(f4) = (v_1+v_4+v_7 \geq m) \Leftrightarrow 0 \leq (v_7+v_4+v_1-m)$, *et d'autre part,* $(v_4 + v_1 - m) \leq (v_7 + v_4 + v_1 - m)$ *et* $(v_4 + v_1 - m) \leq v_1$. ■

L'intervalle I représente alors le domaine admissible de β_1. Pour toute valeur de β_1 dans I, on détermine grâce au système $\mathcal{S}'$ les valeurs des autres variables. Dans la suite, nous supposons que les sous-problèmes $P_k,\ k = 1, \ldots, W$, sont résolus.

[B] Etape 2 : Attribution de repos de trois ou quatre jours

Dans cette section, nous décidons pour chaque semaine w_k, si l'employé e_i prend un 3-RC ($d{-}off_{ik} = 3$) ou un 4-RC ($d{-}off_{ik} = 4$). Les variables x^k et y^k donnent maintenant le nombre d'employés ayant respectivement un 3-RC et un 4-RC au cours de la semaine w_k. La variable c_i donne le nombre de semaines où l'employé e_i bénéficie d'un 4-RC.

Proposition 58:
Pour que le problème d'attribution de repos de trois ou quatre jours admette une solution il faut que :

i) $x^k = 4m - \sum_{j=1}^{7} v_j^k$ *et* $y^k = \sum_{j=1}^{7} v_j^k - 3m,\ k = 1, \ldots, W$,

ii) $c_i = h_i - 3W, i = 1, \ldots, m$.

Preuve:
i) est nécessaire car d'une part, le nombre total de jours de repos attribués au cours de la semaine w_k *est* $3x^k+4y^k = \sum_{j=1}^{7} v_j^k$, *et d'autre part,* $x^k+y^k = m$ *puisqu'un employé prend un 3-RC ou un 4-RC par semaine. D'où* $x^k = 4m - \sum_{j=1}^{7} v_j^k$ *et* $y^k = \sum_{j=1}^{7} v_j^k - 3m,\ k = 1, \ldots, W$. *De même, on retrouve* $c_i = h_i - 3W, i = 1, \ldots, m$ *car* $4c_i + 3(W - c_i) = h_i$. ■

$x^k \geq 0$ et $y^k \geq 0$ impliquent que $3m \leq \sum_{j=1}^{7} v_j^k \leq 4m,\ k = 1, \ldots, W$, est une condition nécessaire d'existence d'une solution au problème d'attribution de jours de repos. Nous supposons dans la suite que cette condition est satisfaite.

Cette étape est résolue par la recherche d'un flot maximal dans le graphe G de la figure 7.2 avec les nouvelles valeurs des y^k et c_i. Nous démontrons aussi que le problème d'attribution de jours de repos admet une solution si et seulement si la valeur du flot maximal dans G est $\sum_{k=1}^{W} y^k = \sum_{i=1}^{m} c_i$.

[C] Etape 3 : Reconstruction d'une solution

Si la première et la deuxième étapes sont résolues alors une solution est reconstruite par la procédure dates-de-repos-étendu suivante :

Procédure dates-de-repos-étendu

Données : $\{\alpha_j^k, \beta_j^k,\ j = 1, \ldots, 7, k = 1, \ldots, W\}$ (*Calculés par l'étape 1*) ;
$\{d{-}off_{ik},\ i = 1, \ldots, m, k = 1, \ldots, W\}$ (*Calculés par l'étape 2*) ;
pour $k = 1$ **à** W **faire**
1. $X_k \leftarrow \{e_i$ tel que $d{-}off_{ik} = 3\}$;
// employés ayant un 3-RC au cours de la semaine w_k ;
2. $Y_k \leftarrow \{e_i$ tel que $d{-}off_{ik} = 4\}$;
// employés ayant un 4-RC au cours de la semaine w_k ;
// $X_k \cap Y_k = \emptyset$ *et* $|X_k| + |Y_k| = m$
3. trier X_k et Y_k selon la date de fin de repos au cours de w_{k-1} ;
//X_1 *et* Y_1 *sont triés selon les indices* ;
4. pour $j = 1$ **à** 5 **faire**
4.1 attribuer un 3-RC débutant d_j de w_k aux premiers α_j^k employés de X_k et enlevez les de X_k ;
4.2 attribuer un 4-RC débutant d_j de w_k aux premiers β_j^k employés de Y_k et enlevez les de Y_k ;
fin faire ;
fin faire ;

Enfin, le problème de base étendu est de complexité $O(mW + max(m \log m, W \log W))$ et il admet une solution si et seulement les étapes 1 et 2 admettent chacune une solution. Les démonstrations sont équivalentes à celles proposées pour le problème de base.

7.4.3 Problème de repos mensuels étendu

Nous cherchons une solution au problème de base étendu où chaque employé bénéficie d'au moins un 4-RC par mois. Cette étape est résolue également en trois étapes. La première et la dernière étapes sont similaires à celles du problème de base étendu. La deuxième étape consiste à chercher un flot maximal dans le graphe G de la figure 7.5 avec les nouvelles valeurs des capacités $y^k,\ k = 1, \ldots, W$, et $c_i,\ i = 1, \ldots, m$.

7.4.4 Problème de week-ends mensuels étendu

Au problème de base étendu, on ajoute la contrainte suivante : chaque employé bénéficie d'au moins un week-end libre par mois. Nous établissons les conditions nécessaires et suffisantes d'existence d'une solution pour le problème de week-ends mensuels étendu sous certaines restrictions.

Proposition 59:
Si $m \leq v_1 + v_4$ et $v_7 + v_4 \leq m$ alors le problème de week-ends mensuels étendu admet une solution si et seulement si :

i) le problème de base étendu admet une solution,

ii) $m \leq v_7^{4l-3} + v_7^{4l-2} + v_7^{4l-1} + v_7^{4l}$ pour tout mois l.

Preuve:
i) est nécessaire parce que toute solution du problème de week-ends mensuels étendu est aussi une solution au problème de base étendu.
ii) est aussi nécessaire car $v_7^{4l-3} + v_7^{4l-2} + v_7^{4l-1} + v_7^{4l}$ représente le nombre de repos en week-end pendant le mois l.

Réciproquement, soit S une solution au problème de base étendu qui ne respecte pas la contrainte de week-ends libres. Il existe un employé e_a n'ayant pas de week-ends libres pendant le mois l. D'après la condition ii), il existe alors un employé e_b qui prend au moins deux week-ends libres pendant ce mois. Soit w_k une semaine du moins l au cours de laquelle e_b prend un week-end libre. Nous montrons qu'il est possible que e_a prenne un week-end libre au cours de w_k à la place de e_b.

Nous rappelons que α_5^k et β_4^k donnent le nombre d'employés ayant respectivement un 3-RC et un 4-RC pendant le week-end de la semaine w_k. L'idée de la preuve est de modifier les dates des repos dans la semaine w_k en modifiant la valeur de β_1^k dans son domaine admissible $[\beta_{1min}^k, \beta_{1max}^k] = [v_4^k + v_1^k - m, v_7^k + v_4^k + v_1^k - m]$. Trois cas sont à distinguer selon les repos de e_a et e_b dans w_k :

1. *e_a **et** e_b **ont le même nombre de jours de repos dans** w_k : il suffit de permuter les repos des e_a et e_b dans w_k.*
2. *e_a **prend un 3-RC et** e_b **prend un 4-RC dans** w_k. On a $\beta_1^k < v_7^k + v_4^k + v_1^k - m$ car $\beta_4^k > 0$. Tout d'abord, on incrémente β_1^k de 1, ($\hat{\beta}_1^k = \beta_1^k + 1$), et par la suite, on incrémente de 1 le nombre de 3-RC en week-end et on décrémente de 1 le nombre de 4-RC en week-end (voir les expressions des α_1^k et β_4^k). Puis, on résout le problème $\hat{P}_k$ avec la nouvelle valeur de β_1^k. $\hat{P}_k$ admet toujours une solution $\hat{S}$ car $\hat{\beta}_1^k$ est admissible. Dans $\hat{S}$, les dates de repos sont réaffectées au cours de w_k de la manière suivante :*
   - *e_b ne prend pas un week-end libre.*
   - *e_a prend un 3-RC en week-end.*
   - *Tout employé, à l'exception de e_b, garde son éventuel week-end libre.*
   - *Les employés n'ayant pas un week-end libre gardent le même nombre de jours de repos sauf que les dates de début des repos peuvent changer car β_j^k et α_j^k ont été modifiés.*
3. *e_a **prend un 4-RC et** e_b **prend un 3-RC dans** w_k. On a $\beta_1^k > v_4^k + v_1^k - m$ car $\alpha_5^k > 0$. On utilise la même technique que dans le cas précèdent en décrémentant β_1^k de 1 et par conséquent en décrémentant de 1 le nombre de 3-RC en week-end et en incrémentant de 1 le nombre de 4-RC en week-end.*

Nous réitérons cet opération autant de fois que nécessaire jusqu'à ce que chaque employé prenne au moins un week-end libre par mois. On conclut que sous certaines condi-

tions toute solution au problème de base étendu peut être transformée polynomialement en une solution au problème de week-ends mensuels étendu. ■

La complexité des différents problèmes de planification traités est récapitulée dans le tableau ci-dessous :

Problème	**Base**	**Repos mensuels**	**Week-ends mensuels**
P23R	P	P	?, P sous conditions
P34R	P	P	?, P sous conditions

TAB. 7.5 – Complexité des problèmes *P23R* et *P34R*

7.5 Conclusion

Dans ce chapitre, nous nous sommes attachés à la résolution du problème de planification de personnel en considérant non seulement l'effectif journalier imposé par les besoins de l'entreprise, mais aussi le nombre de jours de travail de chaque employé. Nous avons résolu polynomialement ce problème sous différentes strategies de placement de repos à l'aide, entre autres, d'algorithmes de flots. Cependant, il n'est pas évident d'appliquer les techniques proposées pour résoudre le problème lorsque l'horizon ne peut pas être décomposé en semaines indépendantes. Pour ce faire, on décompose l'horizon en semaines indépendantes commençant par lundi et on résout le problème de base avec $d_1 =$ lundi. S'il existe une solution, nous aurons résolu le problème mais en cas d'échec il faudra changer le premier jour des semaines et résoudre de nouveau le problème de base avec $d_1 =$ mardi. Si le problème de base n'admet aucune une solution quelle que soit la valeur de d_1 dans l'ensemble de jours de la semaine alors il nous sera impossible de conclure.

Bibliographie

[1] H. Alfares. Four day workweek scheduling with two or three consecutive days off. *Journal of Mathematical Modelling and Algorithms*, 2(1) :67–80, 2003.

[2] H. Alfares. Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127 :147–177, 2004.

[3] T. Aykin. Optimal shift with multiple break windows. *Management Science*, 31(4) :591–602, 1996.

[4] K. R. Baker. Scheduling a full-time workforce to meet cyclic staffing requirements. *Management Science*, 12(20) :1561–1568, 1972.

[5] S.E. Bechtold and L.W. Jacobs. Implicit modelling of flexible break assignments in optimal shift. *Management Science*, 36(11) :1339–1351, 1990.

[6] R. N. Burns and M. W. Carter. Workforce size and single shift schedules with variable demands. *Management Science*, 31 :599–607, 1985.

[7] C.T. Cezik, G.O. Gunluk, and L. Luss. An integer programming model for the weekly tour scheduling problem. *Naval Research Logistics*, 48(7) :607–624, 2001.

[8] G.B. Dantzig. A comment on edie's traffic delay at toll booths. *Operations Research*, 2(3) :339–341, 1954.

[9] R. Hung. Multiple-shift workforce scheduling under the 3-4 workweek with different weekday and weekend labor requirements. *Management Science*, 40(2) :280–284, 1994.

[10] S.L. Moondra. An lp model for work force scheduling in banks. *Management Science*, 7(4) :299–301, 1979.

[11] N. Musliu, J. Gärtner, and W. Slany. Efficient generation of rotating workforce schedules. *Technical Report DBAI-TR-2000-35, Technische Universität Wien*, 2000.

[12] A. Partouche. A hybrid method for the general rostering problem with a mixed and homogeneous workforce. Technical report, Cahier du lamsade N° 151, Université Paris-Dauphine, 1997.

[13] G. M. Thompson. Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science*, 41(4) :595–607, 1995.

[14] R.D Tiberwala, D. Philippe, and J. Browne. Optimal scheduling of two consecutive idle periods. *Management Science*, 19(1) :71–75, 1972.

Conclusion

Nous avons étudié dans ce manuscrit des problèmes de reconstruction d'objets discrets à partir de leurs projections orthogonales sous différents types de contraintes. Tout au long de ce travail, nous nous sommes intéressés à l'aspect algorithmique de la résolution des problèmes. Nous avons réussi à démontrer l'aspect polynomial de plusieurs problèmes traités. Lorsque les problèmes ont une complexité inconnue, nous nous sommes attachés à les comparer et à traiter des cas particuliers. Le but était double : résoudre des problèmes particuliers qui couvrent une grande partie des applications et appréhender par ce biais la difficulté à résoudre le cas général.

Nous avons considéré dans ce travail quatre types de problèmes : la reconstruction de matrices binaires, la reconstruction de tableaux colorés, le packing et le pavage de barres et la planification de jours de repos. Ces problèmes ont été classés et leur terminologie est donnée dans un glossaire (voir page xiii). Les chapitres 2 et 3 ont été consacrés aux problèmes de reconstruction de matrices binaires sous différentes contraintes d'adjacence et de périodicité. Nous avons montré que $M4adj(H)$ est polynomial, $M6adj(H)$ est NP-complet et $M8adj(H,V)$ est plus difficile que $T3C$. Un axe de recherche intéressant serait la détermination de la complexité du problème $M4adj(H,V)$. Nous avons également montré que les sous-problèmes $MAP(1,1)$ et $MAP(0,p)$ sont polynomiaux. Le cas général (p,q) périodique reste un problème non résolu.

Dans la deuxième partie (chapitre 4), nous avons étudié le problème de reconstruction de tableaux colorés. Nous avons formulé le problème $T3C$ par un programme quadratique continu. Nous avons traité plusieurs cas polynomiaux. Nous avons aussi étudié le problématique de reconstruction d'images avec niveaux de gris.

La troisième partie de notre travail (chapitres 5 et 6) a été consacrée aux problèmes de pavage et packing de dominos et de barres. Nous avons réussi à montrer l'aspect polynomial des problèmes étudiés lorsqu'une seule projection orthogonale est prise en compte. Le résultat principal de cette partie est la polynomialité du problème $PacBHMin(H,V)$. L'importance du problème $PacBHMin(H,V)$ provient, d'une part de sa généralité (deux projections et un écart minimal), et d'autre part, de ses applications. Lorsqu'on remplace l'écart minimal par un écart maximal, le problème est de complexité inconnue. C'est pourquoi, nous avons traité des sous-problèmes obtenus en relâchant une projection orthogonale ou en supposant que la projection horizontale est constante ou en considérant un écart maximal unitaire. Lorsque les barres sont de longueurs quelconques, le problème de packing associé, $PacHMax(H,V)$ reste également de complexité inconnue.

Dans le dernier chapitre, nous avons étudié un problème de planification de personnel

sous différentes stratégies de placement de jours repos. Nous avons traité trois problèmes : le problème de "base" de planification, le problème de planification de repos mensuels et le problème de planification de week-ends mensuels. Nous avons montré que les deux premiers problèmes sont polynomiaux en proposant une résolution très simple fondée sur des calculs algébriques et des algorithmes de flot. Le problème de week-ends mensuels est de complexité inconnue mais sous certaines restrictions, il est polynomial.

Un challenge est de déterminer la complexité des problèmes restés ouverts après ce travail. Il serait intéressant d'associer des problèmes d'optimisation à ces problèmes de décision comme cela a été fait dans la section 4.3.2 pour le problème TkC. On pourrait alors envisager de leur apporter des solutions approchées.

L'ensemble des résultats significatifs de complexité démontrés dans cette thèse est récapitulé dans le tableau suivant :

Problème	**Complexité**
$M4adj(H)$	P
$M2.4adj(H,V)$	P
$M3.4adj(H,V)$	P
$M4adj(H,V)$	?, P si $m \leq 3$
$M6adj(H,V)$	NP
$M8adj(H,V)$	$\geq$ T3C
$MAP(1,1)$	P
$MAP(0,p)$	P
$PavDUH(H,V)$	P
$PacDBH(H,V)$	?
$PavDBH(H,V)$	P
$PavDBO(H,V)$	$\Leftrightarrow$ Packing de dominos
$PacBHMin(H,V)$	P
$PacBHMax(V)$	P
$PacBHMax(H)$	P
$PacBHMax(H=c,V)$	P
$PacHMaxu(H,V)$	P
$Pac23BH(H,V)$	$\geq$ T3C
$PacHC(H,V)$	P
$PacHMax(H)$	P
$PacHMax(V)$	P
$PacHMax(H,V)$	?
Problème de base	P
Problème de repos mensuels	P
Problème de week-ends mensuels	?, P sous conditions

TAB. 7.6 – Synthèse des résultats de complexité. P : polynomial

Bibliographie

[1] H. Alfares. Four day workweek scheduling with two or three consecutive days off. *Journal of Mathematical Modelling and Algorithms*, 2(1) :67–80, 2003.

[2] H. Alfares. Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127 :147–177, 2004.

[3] T. Aykin. Optimal shift with multiple break windows. *Management Science*, 31(4) :591–602, 1996.

[4] K. R. Baker. Scheduling a full-time workforce to meet cyclic staffing requirements. *Management Science*, 12(20) :1561–1568, 1972.

[5] E. Barcucci and A. Del Lungo. Reconstructing convex polyominoes from their horizontal and vertical projections. *Theoretical computer science*, 155(1) :321–347, 1996.

[6] S.E. Bechtold and L.W. Jacobs. Implicit modelling of flexible break assignments in optimal shift. *Management Science*, 36(11) :1339–1351, 1990.

[7] R. N. Burns and M. W. Carter. Workforce size and single shift schedules with variable demands. *Management Science*, 31 :599–607, 1985.

[8] C.T. Cezik, G.O. Gunluk, and L. Luss. An integer programming model for the weekly tour scheduling problem. *Naval Research Logistics*, 48(7) :607–624, 2001.

[9] M. Chrobak, P. Couperus, C. Dürr, and G. Woeginger. A note on tiling under tomographic constraints. *Theoretical Computer Science*, 290 :2125–2136, 2003.

[10] M. Chrobak and C. Dürr. Reconstructing hv-convex polyominoes from orthogonal projections. *Information Processing Letters*, 69 :283–289, 1999.

[11] M. Chrobak and C. Dürr. Reconstructing polyatomic structures from x-rays : Np completness proof for three atoms. *Theoretical computer science*, 259(1) :81–98, 2001.

[12] M.C. Costa, D. de Werra, and C. Picouleau. On some special cases of an image reconstruction. Technical report, Cedric-Cnam, 2002.

[13] G.B. Dantzig. A comment on edie's traffic delay at toll booths. *Operations Research*, 2(3) :339–341, 1954.

[14] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computionnal complexity of reconstructing lattice sets from their x-rays. *Discrete Mathematics*, 202 :45–71, 1999.

[15] R.J. Gardner, P. Gritzmann, and D. Prangenberg. On the computational complexity of determining polyatomic structures by x-rays. *Theoretical computer science*, 233 :91–106, 2000.

[16] R. M. Haber. Term rank of 0,1 matrices. *Rend. Sem. Mat. Univ. padova*, 30 :24–51, 1960.

[17] R. Hung. Multiple-shift workforce scheduling under the 3-4 workweek with different weekday and weekend labor requirements. *Management Science*, 40(2) :280–284, 1994.

[18] A. Kuba and G.T. Hermann. Discrete tomography : a historical overview. In *Discrete Tomography : Foundations, Algorithms and Applications*, pages 3–33. Birkhauser, 1999.

[19] A. Del Lungo, A. Frosini, M. Nivat, and L. Vuillon. Reconstruction under periodicity constraints. *ICALP*, pages 38–56, 2002.

[20] S.L. Moondra. An lp model for work force scheduling in banks. *Management Science*, 7(4) :299–301, 1979.

[21] N. Musliu, J. Gärtner, and W. Slany. Efficient generation of rotating workforce schedules. *Technical Report DBAI-TR-2000-35, Technische Universität Wien*, 2000.

[22] A. Partouche. A hybrid method for the general rostering problem with a mixed and homogeneous workforce. Technical report, Cahier du lamsade N° 151, Université Paris-Dauphine, 1997.

[23] C. Picouleau. Reconstruction of a coloured domino tiling from its projections. Technical report, Cedric-Cnam, 2001.

[24] C. Picouleau. Reconstruction of domino tiling from its two orthogonal projections. *Theoretical computer science*, 255(1) :437–447, 2001.

[25] H.J. Ryser. Combinatorial properties of matrices of zeros and ones. *Canad. J. Math*, 9 :371–377, 1957.

[26] G. M. Thompson. Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science*, 41(4) :595–607, 1995.

[27] R.D Tiberwala, D. Philippe, and J. Browne. Optimal scheduling of two consecutive idle periods. *Management Science*, 19(1) :71–75, 1972.

[28] B. Wang and F. Zhang. On the precise number of (0,1)-matrices in u(r,s). *Discrete Mathematics*, 187 :211–220, 1998.

[29] G.J. Woeginger. The reconstruction of polyominoes from their orthogonal projections. *Information Processing Letters*, 77(5-6) :225–229, 2001.

Titre : Résolution de problèmes de tomographie discrète. Application à la planification de personnel.

Résumé
Ce travail de thèse est une contribution à l'étude des problèmes de tomographie discrète. Ces contributions sont à la fois théoriques et pratiques, illustrées par des applications à la planification de personnel et à la reconstruction d'images discrètes.

- Au niveau théorique, nous avons déterminé la complexité et proposé des algorithmes polynomiaux pour résoudre des cas assez généraux pour les problèmes suivants :
  - Packing et pavage par des dominos et des barres.
  - Reconstruction de matrices binaires avec contraintes d'adjacence.
  - Reconstruction de matrices alternées périodiques.
  - Reconstruction de tableaux colorés.
- Au niveau pratique :
  - nous avons proposé une approche originale pour le problème de planification de personnel avec 2-3 ou 3-4 jours de repos consécutifs par semaine.
  - nous avons proposé une heuristique pour aider à la reconstruction des images avec niveaux de gris.

Nous ouvrons aussi de nouvelles perspectives pour l'étude des problèmes de tomographie discrète.

Mots clés : tomographie discrète, planification de personnel, packing, pavage, problème de reconstruction, algorithme polynomial, contrainte périodique, contrainte alternée périodique, contrainte d'adjacence.

Title : Solving problems of discrete tomography. Application in workforce scheduling.

Abstract
This thesis consists in developping the study of discrete tomography problems. My contributions have both theoretic and experimental aspects which are illustrated by applications in workforce scheduling and in image reconstructing.

- Theoric contributions. We have determined the complexity and proposed polynomial algorithms to solve general cases of the following problems :
  - Tiling with dominoes and bars.
  - Reconstructing binary matrices under adjacency constraints.
  - Reconstructing binary matrices under alternate periodicity constraints.
  - Reconstructing colored tables.
- Experimental contributions.
  - We have proposed an original approach to solve the problem of days-off scheduling with 2-3 or 3-4 consecutive days off per week.
  - We have proposed an heuristic to help in reconstruting gray-scale images.

Finally, we open some theoretic perspectives to study other discrete tomography problems.

Keywords : Discrete Tomograpy, Workforce Scheduling, Bar Tiling, Reconstruction Problem, Polynomial Time Algorithm, Periodical Constraint, Alternate Periodical Constraint, Adjacency Constraint.