

# Projet : Un problème de tomographie discrète

MOGPL : MODÉLISATION ET OPTIMISATION PAR LES GRAPHS ET LA  
PROGRAMMATION LINÉAIRE

REALISÉ PAR  
BECIRSPAHIC LUCAS  
ET  
ADOUM ROBERT

# Contents

<b>Contents</b>	<b>2</b>
I. Raisonnement par programmation dynamique . . . . .	3
1 - Première étape . . . . .	3
II. La PLNE à la rescousse . . . . .	5
1 - Modélisation . . . . .	5
2 - Implantation et tests . . . . .	6

## I. Raisonnement par programmation dynamique

### 1 - Première étape

#### Question 1:

Si l'on a calculé tous les  $T(j, l)$ , pour savoir si il est possible de colorier la ligne  $l_i$  entière avec la séquence entière il suffit de regarder  $T(m - 1, k)$ , si ce dernier vaut vrai alors il est possible de colorier la ligne entière avec la séquence entière. Si il vaut faux alors ce n'est pas possible.

#### Question 2:

- Cas  $l = 0, j \in \{0, \dots, m - 1\}$ : Vrai
- Cas  $l \geq 1, j < s_l - 1$ : Faux
- Cas  $l \geq 1, j = s_l - 1$ :
  - Si  $l = 1$  alors Vrai
  - Si  $l \neq 1$  alors Faux

justification : () Si il n'y a pas de bloc à poser, alors un coloriage est toujours possible.

Si le nombre de cases dont on dispose est inférieur à la taille du bloc, on ne peut pas le poser donc faux.

Si le bloc fait exactement la taille de nos cases, on regarde si il y a un unique bloc à poser. Si ce n'est pas le cas, on renvoi faux.

#### Question 3:

La relation de récurrence permettant de calculer  $T(j, l)$  est la suivante:

$$T(j, l) = T(j - (s_l + 1), l - 1)$$

En effet si l'on se trouve à la case  $j$  et que l'on veut savoir si il est possible de colorier la sous séquence  $(s_1, \dots, s_l)$  il faut pouvoir colorier  $s_l$  case(s) et laisser une case de séparation entre les coloration de  $s_{l-1}$  et  $s_l$ , il faut donc regarder si l'on peut colorier la ligne de la case 0 à  $j - s_l - 1$  avec la sous séquence  $(s_1, \dots, s_{l-1})$

instances	nbCases	time
0	20	0.00042200088501
1	25	0.000617027282715
2	400	0.117752075195
3	481	0.0961720943451
4	625	0.182909011841
5	675	0.199213027954
6	900	0.51091504097
7	1054	0.300116062164
8	1400	0.43498301506
9	2500	5.42304491997
10	9801	8.71296691895

**Question 9** En appliquant notre programme sur l'instance 11, on observe qu'en dépit de la petite taille de l'instance notre algorithme ne colorie rien. En effet quand une case peut être colorié à la fois en blanc et en noir notre algorithme ne fais rien. Si la couleur d'une case ne peut être déterminée de manière exacte grace aux contraintes elle ne sera pas colorié. Ce qui explique pourquoi notre algorithme ne colorie pas correctement l'instance 11.

Une solution à ce problème est d'implémenter un algorithme de backtracking qui une fois la coloration effectuée, observe toutes les cases non coloriées et leur affecte 0 et 1 arbitrairement puis on relance coloration avec la nouvelle grille. On réitère jusqu'à obtenir une grille complète (dans ce cas fin de l'algorithme) ou une grille insolvable. Si la grille ne peut pas être résolue, on retourne

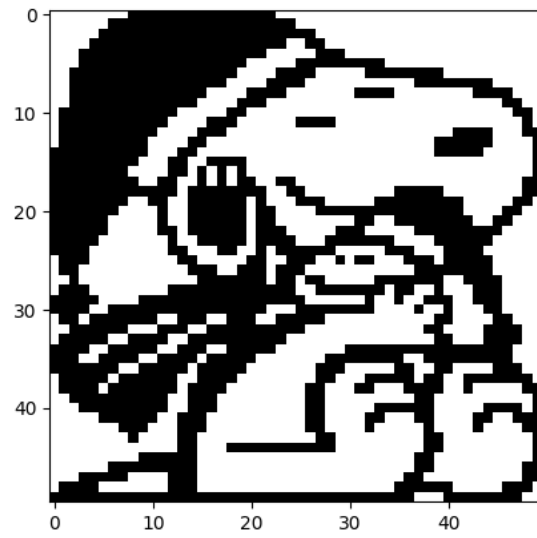


Figure 0.1: Grille de l'instance numéro 9

jusqu'à l'affectation la plus récente et on prend l'autre couleur. On notera que cette algorithme prend beaucoup plus de temps pour résoudre les grilles, une autre approche est d'utiliser la PLNE.

## II. La PLNE à la rescousse

### 1 - Modélisation

#### Question 10:

- $x_{ij}$  vaut 1 si la case (i, j) est coloriée en noir et 0 si coloriée en noir.
- $y_{ij}^t$  vaut 1 si le  $t_{ieme}$  bloc de la ligne  $l_i$  commence à la case (i, j) et 0 sinon.
- $z_{ij}^t$  vaut 1 si le  $t_{ieme}$  bloc de la colonne  $c_j$  commence à la case (i, j) et 0 sinon.

Par conséquent on a :  $y_{ij}^t = 1 \Rightarrow \sum_{k=j}^{j+s_t-1} x_{ik} = s_t$

Et donc  $\sum_{k=j}^{j+s_t-1} x_{ik} = y_{ij}^t \times s_t$

Par conséquent la condition est:  $\sum_{k=j}^{j+s_t-1} x_{ik} \geq y_{ij}^t \times s_t$

Avec le même raisonnement on a pour les colonnes:  $\sum_{k=i}^{i+s_t-1} x_{kj} \geq z_{ij}^t \times s_t$

#### Question 11:

On cherche à exprimer une contrainte qui empêche de poser un bloc  $t+1$  avant que le bloc  $t$

soit posé c'est à dire :  $y_{ij}^t = 1 \Rightarrow \sum_{k=0}^{j+s_t} y_{ik}^{t+1} = 0$

et  $y_{ij}^t = 0 \Rightarrow \sum_{k=j}^{j+s_t} y_{ik}^{t+1} \in \{0, 1\}$

Et donc la condition est:  $y_{ij}^t + \sum_{k=0}^{j+s_t} y_{ik}^{t+1} \leq 1$

Avec le même raisonnement on a pour les colonnes:  $z_{ij}^t + \sum_{k=0}^{i+s_t} z_{kj}^{t+1} \leq 1$

instances	time
11	0.0006639957427978516
12	117.45809602737427
13	1.6546478271484375
14	0.47141194343566895
15	15.691749095916748

Figure 0.2: Temps PLNE

**Question 12:**

$$\begin{aligned}
\text{Min } z = & \sum_{i=0, j=0}^{N, M} x_{i,j} \\
s.c \quad & \left\{ \begin{array}{l}
\sum_{k=j}^{j+s_t-1} x_{ik} \geq y_{ij}^t \times s_t \mid \forall i \in \{0, 1, 2, \dots, N-1\}, \forall t \in \{1, 2, \dots, k_i\} \\
\sum_{k=i}^{i+s_t-1} x_{kj} \geq z_{ij}^t \times s_t \mid \forall j \in \{0, 1, 2, \dots, M-1\}, \forall t \in \{1, 2, \dots, k_j\} \\
y_{ij}^t + \sum_{k=0}^{j+s_t} y_{ik}^{t+1} \leq 1 \mid \forall i \in \{0, 1, 2, \dots, N-1\}, \forall t \in \{1, 2, \dots, k_i\} \\
z_{ij}^t + \sum_{k=0}^{i+s_t} z_{kj}^{t+1} \leq 1 \mid \forall j \in \{0, 1, 2, \dots, M-1\}, \forall t \in \{1, 2, \dots, k_j\} \\
\sum_{j=0}^{M-1} y_{ij}^t = 1 \mid \forall i \in \{0, 1, 2, \dots, N-1\}, \forall t \in \{1, 2, \dots, k_i\} \\
\sum_{i=0}^{N-1} z_{ij}^t = 1 \mid \forall j \in \{0, 1, 2, \dots, M-1\}, \forall t \in \{1, 2, \dots, k_j\} \\
x_{ij} \in \{0, 1\} \mid \forall i \in \{0, 1, 2, \dots, N-1\}, \forall j \in \{0, 1, 2, \dots, M-1\} \\
y_{ij}^t \in \{0, 1\} \mid \forall i \in \{0, 1, 2, \dots, N-1\}, \forall j \in \{0, 1, 2, \dots, M-1\}, \forall t \in \{1, 2, \dots, k_i\} \\
z_{ij}^t \in \{0, 1\} \mid \forall i \in \{0, 1, 2, \dots, N-1\}, \forall j \in \{0, 1, 2, \dots, M-1\}, \forall t \in \{1, 2, \dots, k_j\}
\end{array} \right.
\end{aligned}$$

## 2 - Implantation et tests

**Question 13:**

(N'oublions pas que j commence à 0 et termine à M-1)

- Pour une ligne  $l_i$  le  $l^{ieme}$  bloc ne peut commencer avant la case  $(i, \sum_{n=1}^{l-1} (s_n + 1))$  , ni commencer après la case  $(i, M - s_l - \sum_{n=l+1}^{k_i} (s_n + 1))$  .

instances	plne-time	dynamique-time
0	0.0007050037384033203	0.0004220008850097656
1	0.0012221336364746094	0.0006170272827148438
2	2.4385571479797363	0.1177520751953125
3	0.10664486885070801	0.09617209434509277
4	9.278510093688965	0.1829090118408203
5	2.123404026031494	0.19921302795410156
6	120.54762291908264	0.5109150409698486
7	0.5048248767852783	0.30011606216430664
8	1.1952688694000244	0.4349830150604248
9	timeout	5.423044919967651
10	timeout	8.712966918945312

Figure 0.3: Comparaison des temps pour les 2 methodes

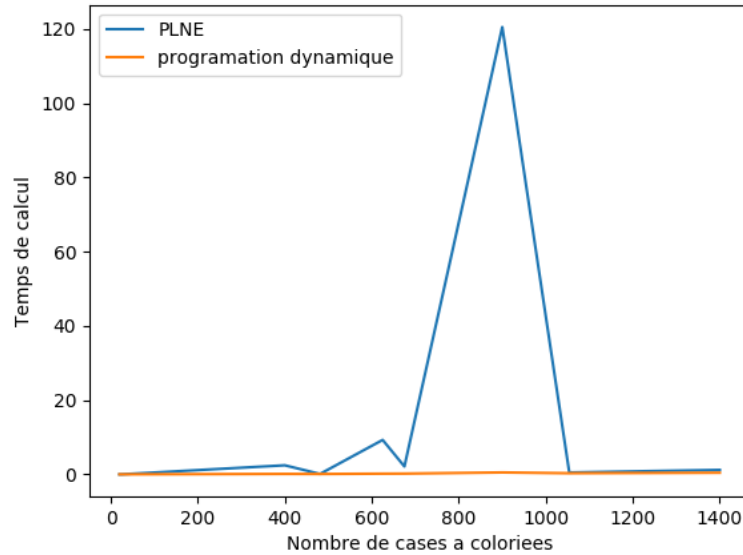


Figure 0.4: Comparaison des deux methodes sur les 8 premières instances

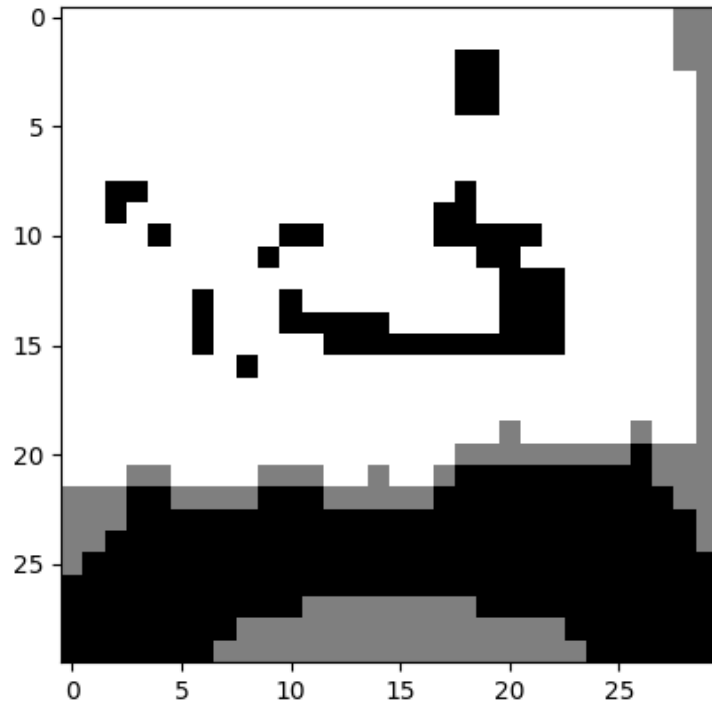


Figure 0.5: Images de l'instance 15 avec la programmation dynamique

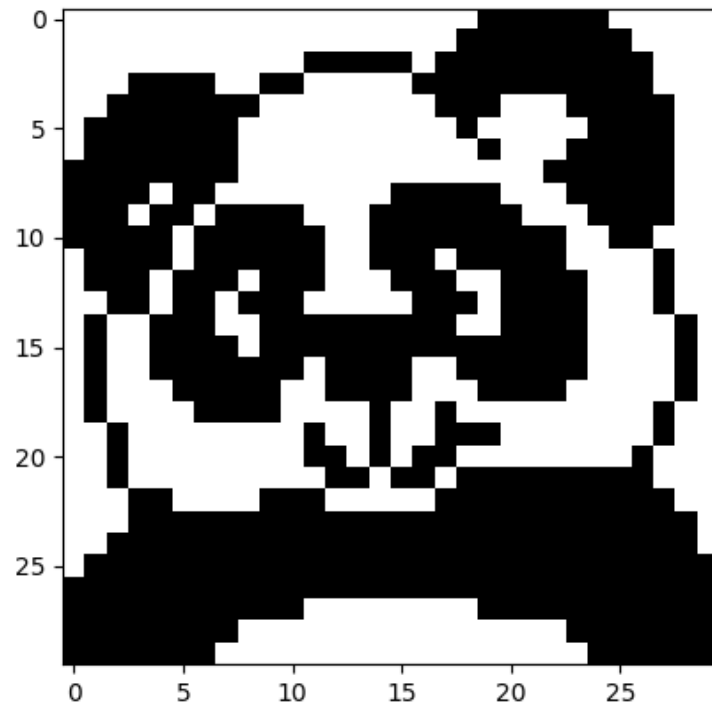


Figure 0.6: Image de l'instance 15 avec la PLNE



instances	plne-time	mix-time
11	0.000541925430298	0.00097918510437
12	162.90399909	1.08105015755
13	2.34310412407	1.3090941906
14	0.735449075699	0.824674129486
15	18.1603360176	7.86031389236
16	timeout	1650.86029291

Figure 0.7: Comparaison des temps entre la plne pure et avec une initialisation avec la programmation dynamique(mix)