Projet : Un problème de tomographie discrète

MOGPL : Modélisation et Optimisation par les Graphes et la Programmation Linéaire

Realisé par
BECIRSPAHIC Lucas
et
ADOUM Robert

Contents

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 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 $l = 0$, $j \in \{0, ..., m-1\}$: Vrai
  justification : Si il n'y a pas de bloc à poser, alors un coloriage est toujours possible.
- Cas $l \geqslant 1$, $j < s_l - 1$: Faux
  justification : Si le nombre de cases dont on dispose est inférieur à la taille du bloc, on ne peut pas le poser donc faux.
- Cas $l \geqslant 1$, $j = s_l - 1$:
  - Si $l = 1$ alors Vrai
  - Si $l \neq 1$ alors Faux

  justification : 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:

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

En effet si l'on se trouve à la case j qui est noir, et que l'on veut savoir si il est possible de colorier la sous séquence $(s_1, ..., s_l)$ il faut pouvoir colorier s_l case(s) et laisser une case de séparation entre les coloration de $s_{l-1} ets_l$, il faut donc regarder si l'on peut colorier la ligne de la case 0 à $j - s_l - 1$ avec la sous séquence $(s_1, ..., s_{l-1})$.
En revanche si la case j est blanche, il n'est pas possible de placer le bloc par consequent on regarde si il est possible de placer la sequences sur les bloc précédents, ce qui s'exprimer par la formule : $T(j-1,l)$

Question 4:

1) Dans le cas, ou l'on a pas de bloc, il faut vérifier qu'aucune case n'est coloriées.
2.a) Si il n'y a pas la place pour mettre un bloc, c'est toujours faux peu importe la ligne
2.b) Si $j = sl - 1$ alors il faut vérifier que l'on peut poser le bloc, c'est à dire il n'y a pas de cases noires sur les cases considérées.
2.c)

- Si la première case est blanche, on ne peut pas poser le bloc par conséquent on regarde $T(j-1,l)$
- Si la case j est noire, on regarde si on peut placer le bloc de manière correcte, c'est à dire pas de blanc sur l'emplacement et un blanc apres et avant pour s'assurer que les blocs son bien séparés.
- Si la case n'est pas encore colorié, on traite les deux cas de figures précédent et il suffit qu'un seul soit vrai pour que l'on considère $T(j,l)$ vrai.

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

Figure 0.1: Tableau représentant les résulats de la programmation dynamique

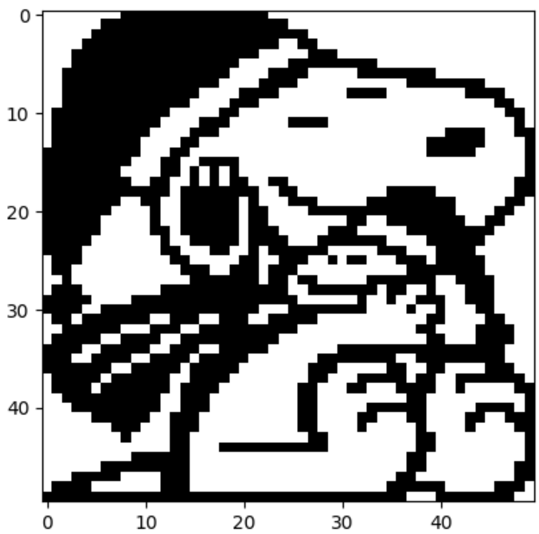

Figure 0.2: Grille de l'instance numéro 9

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 arbirtrairement puis on relance coloration avec la nouvelle grille.

1. Initialisation : A <- coloration(M)

2. Tant que la grille n'est pas complete et que le noeud parent n'a pas testé toutes les couleurs:

   a) Si coloration(A) != False:
   Alors choisir arbitrairement une case incomplete c et lui attribuer une couleur

   b) Si coloration(A) == False:
   Alors en retourne à la grille précédente et on attribue l'autre couleur.

c) A <- coloration(A)

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 : $y_{ij}^t = 1 \Rightarrow \sum\limits_{k=j}^{j+s_t-1} x_{ik} = s_t$

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

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

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

Question 11:

On cherche à exprimer une contrainte qui empeche de poser un bloc t+1 avant que le bloc t soit posé c'est à dire : $y_{ij}^t = 1 \Rightarrow \sum\limits_{k=0}^{j+s_t} y_{ik}^{t+1} = 0$

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

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

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

Question 12:

Min z $= \sum\limits_{i=0,j=0}^{N,M} x_{i,j}$

$$s.c \begin{cases} \sum\limits_{k=j}^{j+s_t-1} x_{ik} \geq y_{ij}^t \times s_t \mid \forall i \in \{0,1,2,...,N-1\}, \forall t \in \{1,2,...,k_i\} \\ \sum\limits_{k=i}^{i+s_t-1} x_{kj} \geq z_{ij}^t \times s_t \mid \forall j \in \{0,1,2,...,M-1\}, \forall t \in \{1,2,...,k_j\} \\ y_{ij}^t + \sum\limits_{k=0}^{j+s_t} y_{ik}^{t+1} \leq 1 \mid \forall i \in \{0,1,2,...,N-1\}, \forall t \in \{1,2,...,k_{i-1}\} \\ z_{ij}^t + \sum\limits_{k=0}^{i+s_t} z_{kj}^{t+1} \leq 1 \mid \forall j \in \{0,1,2,...,M-1\}, \forall t \in \{1,2,...,k_{j-1}\} \\ \sum\limits_{j=0}^{M-1} y_{ij}^t = 1 \mid \forall i \in \{0,1,2,...,N-1\}, \forall t \in \{1,2,...,k_i\} \\ \sum\limits_{i=0}^{N-1} z_{ij}^t = 1 \mid \forall j \in \{0,1,2,...,M-1\}, \forall t \in \{1,2,...,k_j\} \\ x_{ij} \in \{0,1\} \mid \forall i \in \{0,1,2,...,N-1\}, \forall j \in \{0,1,2,...,M-1\} \\ y_{ij}^t \in \{0,1\} \mid \forall i \in \{0,1,2,...,N-1\}, \forall j \in \{0,1,2,...,M-1\}, \forall t \in \{1,2,...,k_i\} \\ z_{ij}^t \in \{0,1\} \mid \forall i \in \{0,1,2,...,N-1\}, \forall j \in \{0,1,2,...,M-1\}, \forall t \in \{1,2,...,k_j\} \end{cases}$$

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 $(i, \sum\limits_{n=1}^{l-1}(s_n+1))$, ni commencer après la case $(i, M - s_l - \sum\limits_{n=l+1}^{k_i}(s_n+1))$.

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

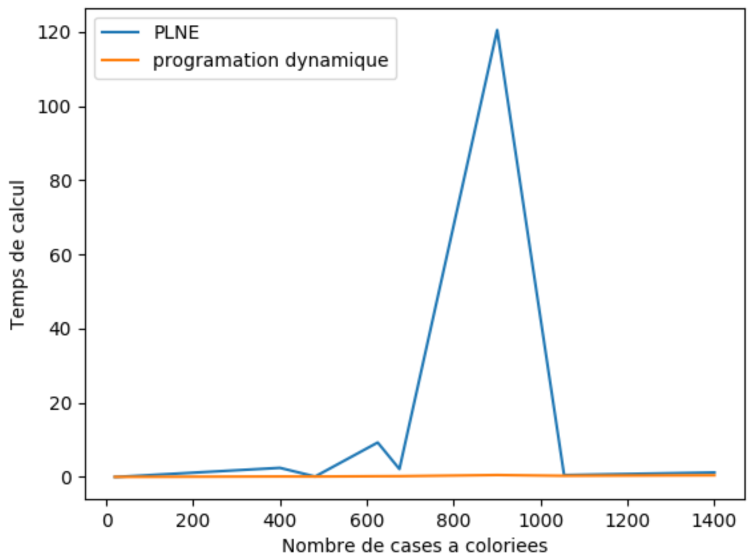

Figure 0.4: Comparaison des deux methodes sur les 8 premières instances

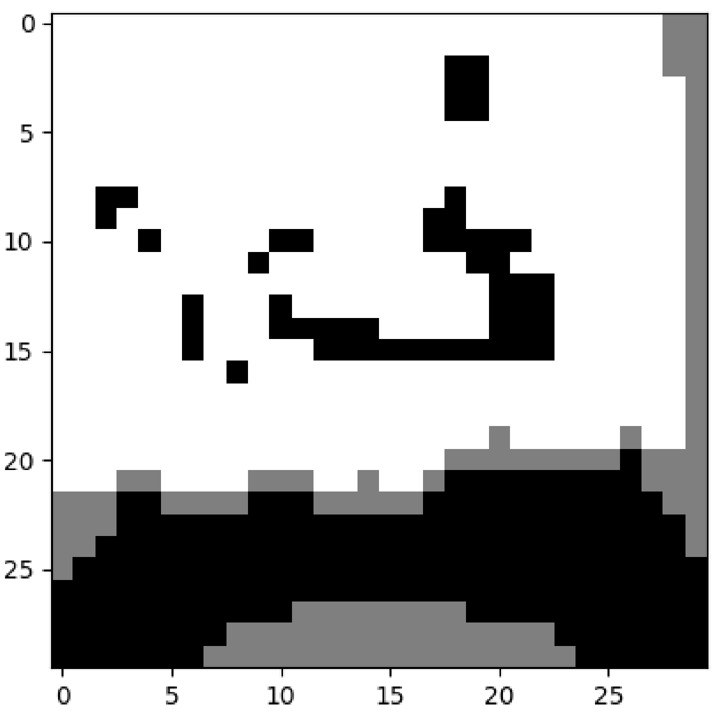

Figure 0.5: Images de l'instance 15 avec la programmation dynamique
le gris correspond aux cases blanches, le noir aux cases noires et le blanc aux cases non déterminées.

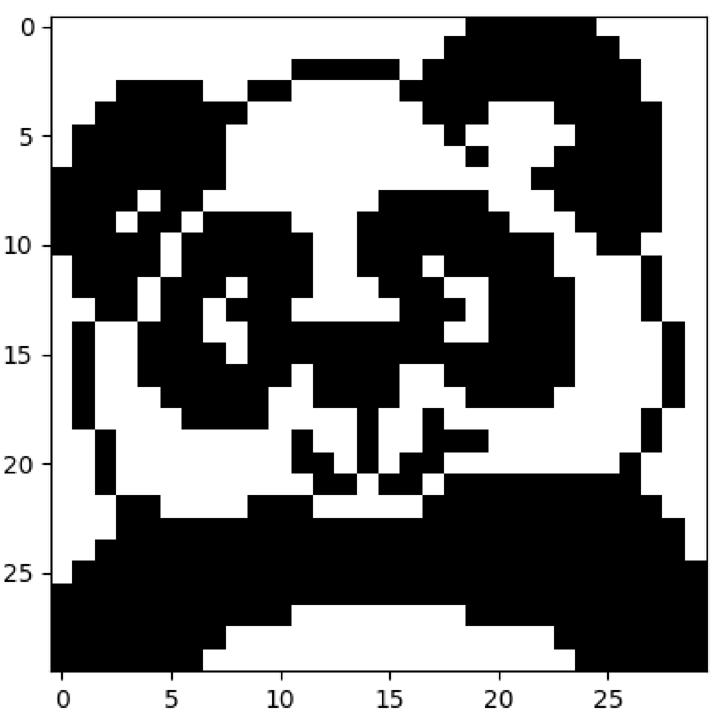

Figure 0.6: Image de l'instance 15 avec la PLNE

instances	plne$_{t}ime$	mix$_{t}ime$	nombre de cases
11	0.0005419254302978516	0.0009791851043701172	8
12	162.9039990901947	1.081050157546997	924
13	2.343104124069214	1.3090941905975342	2025
14	0.7354490756988525	0.824674129486084	1140
15	18.160336017608643	7.860313892364502	900
16	timeout	1650.8602929115295	1750

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