Un problème de tomographie discrète

Le sujet au format pdf ainsi que les instances sont disponibles sur le site de l'UE :
`http://www-poleia.lip6.fr/~perny/ANDROIDE1718/MOGPL/`

Nous considérons une grille de N lignes numérotées de 0 à $N-1$ et M colonnes numérotées de 0 à $M-1$. Chacune des $N \times M$ cases doit être coloriée en blanc ou en noir. A chaque ligne l_i, $i = 0, \ldots, N-1$, est associée une séquence d'entiers représentant les longueurs des blocs de cases noires de la ligne. De même, à chaque colonne c_j, $j = 0, \ldots, M-1$, est associée une séquence d'entiers représentant les longueurs des blocs de cases noires de la colonne.

Considérons l'exemple suivant :

	1 1		1 2		
3					
1 1 1					
3					

Le 3 de la ligne l_0 signifie que celle-ci contient un bloc de trois cases noires consécutives. La séquence 1 2 de la colonne c_2 signifie qu'elle contient deux blocs : le premier d'une case noire, le deuxième de deux cases noires (les blocs sont séparés d'au moins une case blanche).

Une solution du jeu est :

	1 1		1 2		
3	■	■	■		
1 1 1	■		■		■
3			■	■	■

Le but du projet est de construire, s'il en existe, une solution (un coloriage noir-blanc des cases) répondant aux contraintes.

Plus généralement, chaque ligne/colonne se voit donc assigner une séquence (potentiellement vide) $(s_1, s_2, \ldots, s_k)$ de nombres entiers strictement positifs. Le coloriage doit comporter sur cette ligne/colonne un premier bloc de s_1 cases noires consécutives, puis un deuxième de s_2 cases noires, ... Il n'y a pas d'autres cases noires dans la ligne/colonne. Les blocs doivent être séparés d'au moins une case blanche. Il peut y avoir des cases blanches avant le premier bloc, et/ou après le dernier.

1 Raisonnement par programmation dynamique

Un raisonnement simple permet de déterminer directement la couleur de certaines cases. Par exemple, supposons qu'une ligne l_i ait un unique bloc de taille $s_1 = 10$, et qu'il y ait $M = 15$ colonnes. Alors nécessairement sur cette ligne les cases (i, j) avec $j = 5, \ldots, 9$ sont noires (on rappelle qu'on numérote les indices des lignes/colonnes/cases à partir de 0).

Nous allons dans cette section colorier partiellement des instances en généralisant ce raisonnement pour trouver la couleur de certaines cases.

1.1 Première étape

Considérons une ligne l_i dont la séquence associée est $(s_1, s_2, \ldots, s_k)$. Nous voulons définir un algorithme permettant de déterminer s'il existe une coloriage possible de la ligne avec cette séquence [1].

Pour cela, on définit $T(j, \ell)$ ($j = 0, \ldots, m-1$, $\ell = 1, \ldots, k$) comme vrai s'il est possible de colorier les $j + 1$ premières cases $(i, 0), (i, 1), \ldots, (i, j)$ de la ligne l_i avec la sous-séquence $(s_1, \ldots, s_\ell)$ des ℓ premiers blocs de la ligne l_i.

(Q1) *Si l'on a calculé tous les $T(j, \ell)$, comment savoir s'il est possible de colorier la ligne l_i entière avec la séquence entière ?*

Pour calculer les $T(j, \ell)$, on considère un algorithme de programmation dynamique basé sur les cas suivants :

1. Cas $\ell = 0$ (pas de bloc), $j \in \{0, \ldots, m - 1\}$
2. Cas $\ell \geq 1$ (au moins un bloc)
   (a) $j < s_\ell - 1$
   (b) $j = s_\ell - 1$
   (c) $j > s_\ell - 1$

(Q2) *Pour chacun des cas de base 1, 2a et 2b, indiquez si $T(j, \ell)$ prend la valeur vrai ou faux, éventuellement sous condition.*

Considérons maintenant le cas 2c. Il y a deux possibilités :
- Soit la case (i, j) est blanche ;
- Soit elle est noire, et dans ce cas-là le dernier bloc de longueur s_ℓ est à la fin de la partie de l_i considérée, il se termine donc à la case (i, j).

(Q3) *Exprimez une relation de récurrence permettant de calculer $T(j, \ell)$ dans le cas 2c.*

(Q4) *Codez l'algorithme, puis testez-le.*

1.2 Généralisation

En réalité, dans l'algorithme de résolution que nous présentons dans la section 1.3, nous allons colorier les cases de la grille au fur et à mesure. Nous aurons donc, à un moment donné, une ligne avec certaines cases coloriées en blanc, d'autres en noir, et d'autres non encore coloriées.

1. On pourrait répondre très simplement à cette question sans recourir à la programmation dynamique, mais il s'agit de préparer la généralisation où on considérera une ligne dont certaines cases sont déjà coloriées en noir ou en blanc (cases imposées).

Il faut donc modifier l'algorithme précédent afin qu'étant donné une séquence $(s_1, \ldots, s_k)$ et une ligne l_i avec certaines cases déjà coloriées en blanc ou en noir, il indique si une coloration de cette ligne est possible.

(Q5) *Modifiez chacun des cas de l'algorithme précédent afin qu'il prenne en compte les cases déjà coloriées.*

(Q6) *Codez l'algorithme, puis testez-le.*

1.3 Propagation

L'algorithme de résolution consiste alors à partir d'une grille non coloriée, puis à tester chaque ligne et chaque colonne pour tenter de colorier des cases en se servant de la programmation dynamique. Ainsi, pour chaque case (i, j) non encore coloriée de la ligne i, on peut :

- tester si elle peut être coloriée en blanc (la colorier en blanc, et tester si la ligne l_i a une réponse positive) ;
- tester de même si elle peut être coloriée en noir ;
- si les deux tests échouent, le puzzle n'a pas de solution. Si les deux sont OK, on ne peut rien déduire. En revanche, si par exemple le test blanc échoue mais le test noir réussit, cela veut dire que la case peut être coloriée en noir.

On itère cela tant que des changements sont possibles. Un pseudo-code vous est donné en annexe.

(Q7) *Codez l'algorithme de propagation. Votre programme prendra en entrée un fichier txt dont chaque ligne correspond à la séquence associée à une ligne ou une colonne de la matrice. Le symbole # indique que l'on passe de la description des lignes à celle des colonnes. Par exemple, le contenu du fichier encodant l'instance de l'introduction est indiqué ci-dessous. L'algorithme devra permettre une* visualisation *du coloriage obtenu en sortie, avec des cases de couleur noire, blanche ou indéterminée. Vous vérifierez que votre programme résout correctement l'instance 0.txt, qui correspond à l'exemple de l'introduction.*

```
3

1 1 1
3
#
1 1
1
1 2
1
2
```

1.4 Tests

(Q8) *Appliquez votre programme sur les instances 1.txt à 10.txt*[2]*. Vous indiquerez dans le rapport les temps de résolution dans un tableau. Vous fournirez dans le rapport la grille obtenue pour l'instance 9.txt.*

2. Les instances, renumérotées, sont tirées du site `webpbn.com`.

(Q9) *Appliquez votre programme sur l'instance 11.txt. Que remarquez-vous ? Expliquez.*

2 La PLNE à la rescousse

Pour résoudre les grilles plus complexes, nous allons faire appel à la Pogrammation Linéaire en Nombres Entiers (PLNE).

2.1 Modélisation

On définit $N \times M$ variables x_{ij} ($i = 0, \ldots, N-1$, $j = 0, \ldots, M-1$) valant 1 si la case (i, j) est coloriée en noir, 0 si elle est coloriée en blanc.

Pour chaque ligne l_i possédant une séquence $(s_1, s_2, \ldots, s_k)$, on définit kM variables $y_{ij}^1, \ldots, y_{ij}^k$, $j = 0, \ldots, M-1$. La variable y_{ij}^t vaut 1 si le t^{ieme} bloc de la ligne l_i commence à la case (i, j) (et 0 sinon).

On définit de même kN variables z_{ij}^t. La variable z_{ij}^t vaut 1 si le t^{ieme} bloc de la colonne c_j commence à la case (i, j) (et 0 sinon).

(Q10) *Formulez une contrainte qui exprime le fait que si le t^{ieme} bloc de la ligne l_i commence à la case (i, j), alors les cases (i, j) à $(i, j + s_t - 1)$ sont noires. Faire de même pour les colonnes (en adaptant bien sûr).*

(Q11) *Formulez une contrainte qui exprime le décalage nécessaire entre le début des blocs s_t et s_{t+1} de la ligne l_i : si le t^{ieme} bloc commence à la case (i, j), alors le $(t+1)^{ieme}$ ne peut pas commencer avant la case $(i, j + s_t + 1)$.*

(Q12) *Formulez alors le problème de coloriage comme un PLNE (attention, les familles de contraintes des questions 10 et 11 ne sont pas suffisantes à elles seules pour assurer la validité d'une solution).*

2.2 Implantation et tests

Avant d'implanter, limitons le nombre de variables. Par exemple, si une ligne l_i a une séquence $(3, 2, 2)$ et qu'il y a 15 colonnes, alors le deuxième bloc de la ligne l_i ne peut commencer avant la case $(i, 4)$, ni commencer après la case $(i, 10)$.

(Q13) *Exprimez de manière générale, pour une ligne l_i, un intervalle hors duquel le ℓ^{ieme} bloc ($l \in \{1, \ldots, k\}$) ne peut commencer.*

(Q14) *Implantez la résolution par PLNE. Vous vérifierez que votre programme résout correctement l'instance 11.txt.*

Pour la résolution du PLNE, vous utiliserez le logiciel Gurobi vu en séance de TME.

(Q15) *Résolvez les instances 1.txt à 16.txt avec un* timeout *de 2 minutes. Donnez les temps de calcul dans un tableau. Pour les instances 12.txt à 16.txt, appliquez également la méthode de la section 1. Commentez. Vous fournirez dans le rapport la grille obtenue avec chacune des deux méthodes pour l'instance 15.txt.*

3 Pour aller plus loin (facultatif)

Pour aller plus loin, vous pouvez étendre votre projet en proposant des améliorations ou compléments, par exemple :

— Implantez une méthode globale en utilisant d'abord le prétraitement puis la PLNE sur les cases non déterminées (à comparer avec l'emploi de la PLNE seule) ;
— Evaluez la complexité au pire cas et la complexité empirique de la méthode de la section 1 en fonction de la taille de la grille ;
— Améliorations libres ...

4 Travail demandé

Le travail se fait obligatoirement en binôme. Les binômes seront communiqués au chargé de TD au plus tard mi-octobre. Le langage de développement est libre.

4.1 Rapport

Chaque binôme doit rédiger un rapport (de 5 pages environ, et dans tous les cas moins de 10 pages) répondant aux différentes questions de l'énoncé (quelques explications et commentaires peuvent ne pas être superflus).

Le rapport (format pdf) et le code (commenté) sont à envoyer par email au chargé de TD avant le **dimanche 10 décembre** 23h59.

4.2 Soutenance

Une soutenance aura lieu la semaine du 18 décembre en salle machine (sur le créneau de la dernière séance de TD-TME). Chaque binôme doit donc avoir son code prêt à être exécuté sur son compte à la PPTI.

Chaque binôme présentera en 5 minutes son travail (simplement sur la base de son code, pas de transparent ou autre), puis répondra aux questions.

NB : il pourra être demandé d'exécuter le code sur des instances autres que celle du projet. Votre programme devra donc pouvoir résoudre des instances fournies sur des fichiers txt (au format spécifié dans le projet).

Il est impératif de remettre le rapport et d'effectuer la soutenance dans le groupe où vous êtes inscrit.

5 Annexe : pseudo-code de l'algorithme de propagation

Dans le pseudo-code ci-dessous, on note A la grille à colorier, A_i la ligne d'indice i de A, et A^j la colonne d'indice j de A.

Algorithme 1 Algorithme de résolution partielle

```
1:  procédure COLORATION(A)
2:      LignesAVoir ← {0, ..., N − 1}
3:      ColonnesAVoir ← {0, ..., M − 1}
4:      tant que LignesAVoir ≠ ∅ ou ColonnesAVoir ≠ ∅ faire
5:          pour i dans LignesAVoir faire
6:              Tester par programmation dynamique les cases non coloriées de la ligne A_i
7:              Nouveaux ← {j : j est une nouvelle case coloriée}
8:              ColonnesAVoir ← ColonnesAVoir ∪ Nouveaux
9:              LignesAVoir ← LignesAVoir − {i}
10:         fin pour
11:         pour j dans ColonnesAVoir faire
12:             Tester par programmation dynamique les cases non coloriées de la colonne A^j
13:             Nouveaux ← {i : i est une nouvelle case coloriée}
14:             LignesAVoir ← LignesAVoir ∪ Nouveaux
15:             ColonnesAVoir ← ColonnesAVoir − {j}
16:         fin pour
17:     fin tant que
18: fin procédure
```