

UNIVERSITÉ LIBRE DE BRUXELLES
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

INFO-F-302 - Logique Informatique
Projet : Le jeu Pattern et Utilisation de
MiniSAT

OOMS Aurélien, SONNET Jean-Baptiste

Table des matières

1	Énumération 3,2	2
1.1	Problème et notation	2
1.1.1	Grille	2
1.1.2	Cases	2
1.2	Parcours et représentation par énumération	3
2	Les Bandes	5
2.1	Problème et notation	5
2.2	Représentation par bandes	6
2.2.1	Observation : position minimale, maximale et sous-espace	6
2.2.2	Énumération récursive	7

Chapitre 1

Énumération 3,2

1.1 Problème et notation

1.1.1 Grille

Le problème est présenté sous forme d'une grille 3×3 contenant au *max* 2 contraintes par ligne ou colonne.

Soit une matrice 3×3 ,

$$\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} \\ x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \end{bmatrix}$$

où chacune des cases $x_{i,j}$ prendra potentiellement une des 3 valeurs : $\{0, 1, -1\}$, respectivement l'inconnu, le noir, le blanc.

On aura par exemple comme problème à résoudre :

$$\begin{array}{ccc} & 2 & 1 & 2 \\ \begin{array}{c} 1 \\ 1 \, 1 \\ 2 \end{array} & \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{array}{|c|c|c|} \hline \blacksquare & \square & ? \\ \hline ? & ? & ? \\ \hline ? & ? & ? \\ \hline \end{array} \end{array}$$

Selon les contraintes précisées, la solution devra donner pour toutes les cases inconnues une valeur de 1 ou -1 :

$$\begin{array}{ccc} & 2 & 1 & 2 \\ \begin{array}{c} 1 \\ 1 \, 1 \\ 2 \end{array} & \begin{bmatrix} 1 & -1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix} & \begin{array}{|c|c|c|} \hline \blacksquare & \square & \square \\ \hline \blacksquare & \square & \blacksquare \\ \hline \square & \blacksquare & \blacksquare \\ \hline \end{array} \end{array}$$

1.1.2 Cases

Chaque case est représentée par une variable x , de sorte que la case à la i^{eme} ligne et à la j^{eme} colonne se note $x_{i,j}$

1.2 Parcours et représentation par énumération

On va parcourir la grille et énumérer les combinaisons possibles premièrement suivant les contraintes de lignes ensuite selon les contraintes de colonnes.

Par ligne (respectivement colonne), on crée si nécessaire une clause représentant les cas possibles quant aux cases dont on ne connaît pas encore la nature.

Pour chaque case contenue dans une ligne donnée :

1. Si elle contient une information, on en crée une clause à part entière qui sera jointe aux autres.
2. Si elle ne contient pas d'informations et conditionnellement à la contrainte, on énumère les possibilités sous forme disjonctive (clause).

On effectue de même par colonne et relativement aux contraintes de la colonne.

L'énumération des cas possibles se formule naturellement à l'aide de la disjonction exclusive (XOR ou \oplus), à chaque étape, il convient de la traduire en forme normale conjonctive (FNC).

$$a \oplus b \equiv (a \wedge \neg b) \vee (\neg a \wedge b) \equiv (a \vee b) \wedge \neg(a \wedge b)$$

a	b	$a \oplus b$	$(a \wedge \neg b) \vee (\neg a \wedge b)$	$(a \vee b) \wedge \neg(a \wedge b)$
0	0	0	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	0	0

Algorithm 1 Énumération selon les contraintes de lignes et de colonnes

```
for ligne  $i \rightarrow n$  do ▷ contraintes de lignes
  for colonne  $j \rightarrow n$  do
    if  $x_{i,j} == \perp$  then
      Créer une nouvelle clause avec  $-x_{i,j}$ 
    else if  $x_{i,j} == \top$  then
      Créer une nouvelle clause avec  $x_{i,j}$ 
    else if  $x_{i,j} == 0$  then
      if  $\nexists clause_i$  then
        Créer  $clause_i$ 
      end if
      Ajouter  $(\vee) x_{i,j}$  à la  $clause_i$ 
    end if
    Joindre  $(\wedge)$  les clauses
  end for
  if  $\exists clause_i$  then
    Joindre  $(\wedge)$  la  $clause_i$ 
  end if
end for
for colonne  $j \rightarrow n$  do ▷ contraintes de colonnes
  for ligne  $i \rightarrow n$  do
    if  $x_{i,j} == \perp$  then
      Créer une nouvelle clause avec  $-x_{i,j}$ 
    else if  $x_{i,j} == \top$  then
      Créer une nouvelle clause avec  $x_{i,j}$ 
    else if  $x_{i,j} == 0$  then
      if  $\nexists clause_j$  then
        Créer  $clause_j$ 
      end if
      Ajouter  $(\vee) x_{i,j}$  à la  $clause_j$ 
    end if
    Joindre  $(\wedge)$  les clauses
  end for
  if  $\exists clause_j$  then
    Joindre  $(\wedge)$  la  $clause_j$ 
  end if
end for
```

Chapitre 2

Les Bandes

2.1 Problème et notation

Une suite de cases noires représente une *bande*.

Une bande appartient à une ligne ou colonne et commence à une position donnée. Comme il peut y avoir plusieurs bandes sur une même ligne ou colonne, chaque bande doit être identifiable.

On représente une bande par

$$LBande_{pos,ID,start}$$

où,

L|C précise s'il s'agit d'une ligne ou d'une colonne.

pos désigne le numéro de la ligne ou la colonne où est située la bande.

ID désigne, s'il existe plusieurs bandes sur la ligne ou colonne, la bande que l'on considère.

start désigne le numéro de colonne ou ligne à partir de laquelle commence la bande.

On définit deux fonctions donnant la longueur d'une bande :

$L(pos, ID)$ donnant la longueur de la ID^{eme} bande à la ligne pos .

$C(pos, ID)$ donnant la longueur de la ID^{eme} bande à la colonne pos .

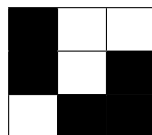


TABLE 2.1 – Grille 3×3

On aura pour la grille 2.1 :

$$LBande_{3,1,2} = \top$$

$$L(3, \underset{3}{2}) = 2$$

et

$$x_{3,j} = \top, \forall j \in [3, 2 + L(3, 2)]$$

$$\wedge$$

$$x_{3,2+L(3,2)+1} = \perp$$

2.2 Représentation par bandes

2.2.1 Observation : position minimale, maximale et sous-espace

Soit une colonne d'une longueur de 8 cases avec pour contrainte 3 bandes de taille 1, 2 et 1.

La première bande pourra prendre la position minimum $x_{1,0}$ et maximum $x_{1,\alpha}$.

La valeur de α peut être obtenue à partir du nombre de contraintes, de la taille des différentes bandes et de la règle selon laquelle deux bandes sont séparés d'au moins une case.

Dans l'exemple, on obtient :

$c = 3$, où c est le nombre de contraintes

$l = 8$ est la longueur de la colonne

$L_{total} = 4$, est la longueur cumulée des bandes dans la colonne

$\alpha = l - (L_{total} + (c - 1))$













0			
1			
2			
3			
4			
5			
6			
7			
	<i>min</i>		<i>max</i>

TABLE 2.2 – position *max* et *min* sur une colonne 8×1 avec la contrainte 1 2 1

On voit dans le tableau 2.2 qu'il est possible de connaître *a priori* le nombre restreint de positions possibles pour la première bande.

On voit aussi que chaque possibilité délimite un *sous-espace* (zone grisée) au sein duquel devront être placées les bandes restantes.

Il est alors possible d'appliquer la même analyse quant à la détermination de la position de la bande suivante au sein du sous-espace.

Une fois les positions de la seconde bande déterminées, elles délimiteront l'espace restant pour la dernière bande.

Suite à cela, il est possible d'élaborer une méthode récursive, ou faussement récursive, mais au nombre d'itérations réduit.

2.2.2 Énumération récursive

						$\updownarrow L(1, 1) = 1$	$x_{1,1} = \top$
						+1	$x_{1,2} = \perp$
						# itération	$x_{1,3} = \perp$
						$\updownarrow L(1, 2) = 2$	$x_{1,4} = \top$
							$x_{1,5} = \top$
						+1	$x_{1,6} = \perp$
						$\updownarrow L(1, 3) = 1$	$x_{1,7} = \top$
							$x_{1,8} = \perp$
•		r_1		r_1		r_3	récursion

TABLE 2.3 – Énumération récursive sur une colonne 8×1 avec la contrainte 1 2 1

Algorithm 2 Énumération des bandes

```

for ligne  $i \rightarrow n$  do
  for colonne  $j \rightarrow n$  do nop();
  end for
end for

```
