

INFO-F-302 : Logique informatique

Projet : Le jeu Pattern et Utilisation d'un solveur SAT

Jean-Sébastien Lerat

14 mai 2013

1 Le jeu Pattern

Le jeu *Pattern* est une grille de 15×15 carreaux initialement vides. En haut et à gauche, un ensemble de chiffres indique la taille des bandes (carreau(x) consécutif(s)) noires contenues dans les lignes et colonnes. Ces bandes sont espacées d'au moins un carreau blanc. Le but du jeu est de compléter entièrement la grille de carreaux noirs et blancs tout en respectant les règles. Un exemple de grille est donné en Figure 1, la solution est donnée en Figure 2. Afin de simplifier le problème, nous allons limiter le nombre de bandes par ligne et colonne à 3.

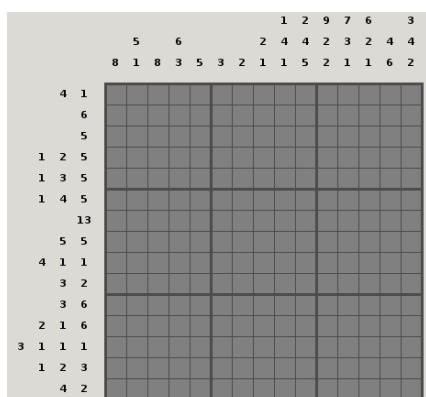


FIGURE 1 – Instance de Pattern

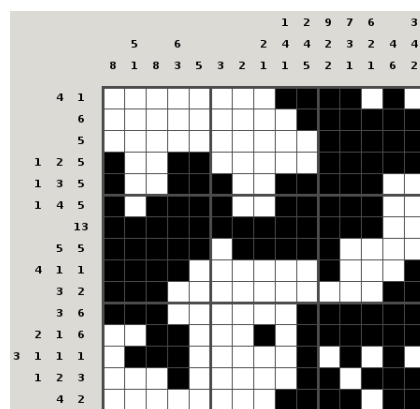


FIGURE 2 – Solution

Format d'une grille

Une grille est représentée par un fichier texte respectant :

- La première ligne contient les dimensions *size*, *tapes* de la grille (15, 3 par défaut : 15 lignes/colonnes et 3 bandes par ligne/colonne) séparés par un/des espace(s).
- Les *size* lignes suivantes contiennent au maximum *tapes* entiers séparés par des espaces qui correspondent aux longueurs des bandes noires des colonnes (de gauche à droite).

- Les *size* lignes suivantes contiennent au maximum *tapes* entiers séparés par des espaces qui correspondent aux longueurs des bandes noires des lignes (de haut en bas).
- Les *size* lignes suivantes contiennent *size* entiers séparés par des espaces qui correspondent à la couleur des carreaux : -1 blanc, 0 vide, 1 noir.

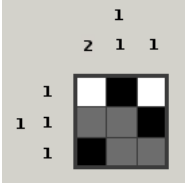
Représentation par des fonctions de longueurs

La représentation d'une grille complétée peut s'effectuer via les fonctions de longueurs L pour les lignes et C pour les colonnes :

$$\begin{aligned} L &: (\text{ligne}, \text{bande}) \rightarrow \mathbb{N} \\ C &: (\text{colonme}, \text{bande}) \rightarrow \mathbb{N} \end{aligned}$$

Pour chaque case $X_{i \in Lignes, j \in Colonnes}$ de la grille, $L(i, b \in LBandes_i)$ (resp. $C(j, b \in CBandes_j)$) équivaut à la longueur de la bande b de la ligne i (resp. colonne j) ssi la bande commence (sens gauche-droite, haut-bas) en $X_{i,j}$.

Exemples

Format de fichier	Représentation en C, L	La grille
<pre> 3 2 2 1 1 1 1 1 1 1 -1 1 -1 0 0 1 1 0 0 </pre>	<pre> C(1,1) = 2 C(2,1) = 1 C(2,2) = 1 C(3,1) = 1 L(1,1) = 1 L(2,1) = 1 L(2,2) = 1 L(3,1) = 1 </pre>	

2 Le solveur Sat MiniSat

MiniSat, disponible sur <http://minisat.se/MiniSat.html>, est un solveur SAT développé en C++ qui travaille avec des formules de logique propositionnelles sous forme normale conjonctive (FNC). MiniSat fournit comme résultat une assignation des variables si le problème est satisfaisable.

Pour l'utiliser, vous devez lui fournir un fichier texte au format DIMACS simplifié. Ce format est le suivant :

- Chaque ligne commençant par un c est un commentaire.
- La première ligne (hors commentaires) doit correspondre à $p \text{ cnf variables clauses}$ où *variables* est le nombre de variables et *clauses* le nombre de clauses.

- Chaque autre ligne est une clause finissant par 0 dont les variables sont séparées par des espaces. Les variables sont représentées par leur numéro d'ordre allant de 1 à *variables*.
- La négation d'une variable est représentée par le signe "-".

Par exemple :

```
c Commentaire.
p cnf 2 3
-1 0
2 0
1 -2 0
```

Remarque : si vous choisissez d'implémenter vos programmes en C++, vous pouvez utiliser la librairie C++ de MiniSat (voir les exemples C++ du cours).

Le fichier de sortie quand à lui respectera le format de MiniSat qui est également un fichier texte. La première ligne se composera du mot clé UNSAT si le problème n'est pas satisfaisable et SAT si le problème est satisfaisable. Dans ce dernier cas, la seconde ligne est une instance de ces variables (comme les clauses au format DIMACS).

3 Questions

Pour chaque question, il vous est demandé d'écrire un fichier – nommé selon le format *Questioni.py* pour Python, *Questioni.bin* pour le binaire C++ (accompagné d'un Makefile), ou *Questioni.class* pour Java – qui va prendre en paramètre (1) une grille au format défini dans la Section 1 (2) un nom pour le fichier de sortie (3) des paramètres supplémentaires si la question l'exige.

Question 1 (Énumération 3,2) Pour cette question, aucune implémentation n'est requise. Par contre, vous devez expliquer dans le rapport votre représentation par énumération, comment vous représentez ce problème en FNC. Les grilles sont en 3×3 contenant au maximum 2 contraintes par lignes/colonnes. Afin de résoudre ce problème, il vous est demandé d'utiliser des variables $X_{\text{ligne}, \text{colonne}}$ représentant chaque case aux coordonnées ligne, colonne. Cette variable X vaut vrai lorsque la case est noir. Nous considérons que $X_{\text{ligne}, \text{colonne}}$ correspond à une case blanche lorsque $X_{\text{ligne}, \text{colonne}}$ est faux.

Question 2 (Les bandes) Pour cette question, aucune implémentation n'est requise. Par contre, vous devez expliquer dans le rapport votre représentation par bandes, comment vous représentez ce problème en FNC. Lors de l'exercice précédent, vous avez dû énumérer tous les cas possibles. La complexité de ce type d'implémentation n'est clairement pas performante. C'est pourquoi nous allons introduire de nouvelles variables $LBande_{\text{pos}, ID, \text{start}}$ pour les lignes (resp. $CBande_{\text{pos}, ID, \text{start}}$ pour les colonnes). Celles-ci vont correspondre à la présence de la bande noire ID de la ligne (resp. colonne) pos , commençant à la colonne (resp. ligne) start . Par exemple, sur la Figure 2 : $LBande_{4,2,4} = \top$ et $L(4,2) = 2$ ce qui implique que

$$\forall j \in [4, 4 + L(4, 2)[, X_{4,j} = \top \wedge X_{4,4+L(4,2)} = \perp$$

Question 3 (Résolution 3,2) *En utilisant MiniSat, écrivez un programme en Python, C++ ou Java qui va résoudre une grille 3×3 contenant au maximum 2 contraintes par lignes/colonnes. Pour cette implémentation, ainsi que pour les suivantes, vous ne pouvez pas employer l'énumération des cases mais vous devez utiliser la représentation en bandes. S'il existe une solution, votre fichier de sortie devra respecter le format de la Section 1. Sinon, le fichier de sortie sera vide. Expliquez votre code et comment vous représentez le problème de résolution de grille (les contraintes, la grille, variables, ...) en FNC.*

Question 4 (Résolution size,tapes) *En utilisant MiniSat, écrivez un programme en Python, C++ ou Java qui va résoudre une grille $size \times size$ contenant au maximum tapes contraintes par lignes/colonnes. S'il existe une solution, votre fichier de sortie devra respecter le format de la Section 1. Sinon, le fichier de sortie sera vide. Expliquez votre code et comment vous représentez le problème de résolution de grille (les contraintes, la grille, variables, ...) en FNC.*

Question 5 (Bonus : Unicité) *En utilisant MiniSat, écrivez un programme Python, C++ ou Java qui va vérifier l'unicité d'une grille.*

Votre fichier de sortie contiendra soit une (si la solution est unique) soit deux solutions au format de la Section 1 (sinon). Expliquez votre code et comment vous représentez le problème d'unicité de la solution (les contraintes, la grille, variables, ...) en FNC.

Question 6 (Taille variable) *Assurez-vous de généraliser vos réponses précédentes pour des grilles de tailles variables lorsqu'il y a lieu. Vous devez utiliser la numérotation alphabétique précédée d'un 6 (i.g. 6a, 6b, 6c, ...). Expliquez l'impact de la taille sur le temps de calcul/complexité (en termes de contraintes et variables).*

Question 7 (Bonus : Génération) *En utilisant MiniSat, écrivez un programme Python, C++ ou Java qui va générer une grille telle qu'il existe au moins une solution. Attention, les grilles générées ne doivent pas être aléatoires mais il faut générer ces grilles comme une instance du problème SAT. Ajoutez un paramètre entier n pour que votre script génère n grilles différentes. Votre fichier de sortie contiendra toutes les grilles générées au format de la Section 1. Expliquez votre code et comment vous représentez le problème de génération de grille (les contraintes, la grille, variables, ...) en FNC et comment vous générez les n grilles.*

Modalités

Le projet se fait en binôme, il est à rendre au secrétariat étudiant au plus tard pour le **21 Mai 2012** midi 12h (fuseau horaire de Bruxelles). Il doit comprendre un rapport papier qui répond aux questions et les codes en annexe.

Envoyez également une version électronique dans un dossier compressé (format ZIP) portant les NetIDs des deux étudiants par e-mail à Jean-Sébastien Lerat (prénom.nom@ulb.ac.be).