Projet INF402 : NoriNori

1. Introduction

Ce document présente le projet réalisé dans le cadre de l'INF402. L'objectif est de modéliser un problème type jeu de casse-tête en clause puis de résoudre ces clauses à l'aide d'un SAT-solveur. Nous avons choisi le jeu du Norinori

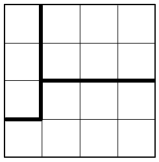
1.1 Rappel des règles

Le jeu du Nornori consiste à remplir une gille carré avec des zones de forme libre en coloriant (ou non) les cases avec les règles suivantes :

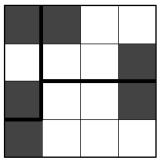
1. Chaque case coloriée doit former des blocs de 2x1 ou 1x2.
2. Deux blocs coloriés ne peuvent pas se toucher entre eux par les côtes (mais ils peuvent par les angles)
3. Chaque zone doit contenir exactement deux cases coloriées noires.

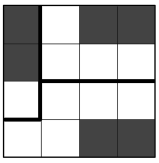
1.2 Exemple

Prenons l'exemple d'une grille de 4x4.



Nous pouvons la résoudre de plusieurs façons, voici deux exemples :





L'exemple est volontairement très simple mais respecte les 3 règles. La difficulté peut être modulée en modifiant la taille de la grille ou le nombre de zones.

2. Modélisation sous forme logique

Afin de trouver une solution au problème, nous devons transformer les règles du Norinori en clauses.

- Soit une grille de taille $n * n$ comportant k zones

- $\forall i \in [1,n], \forall j \in [1,n], \exists k \in [1;\lfloor\frac{n^2}{2}\rfloor] \text{ tel que } x_{i,j,k}$ désigne la case de ligne i et de colonne j et de zone k
- Chaque variable $x_{i,j,k}$ possède une valeur de vérité :
  - 1 : la case $x_{i,j,k}$ est coloriée
  - 0 : la case $x_{i,j,k}$ n'est pas coloriée

Le k désigne la zone à laquelle appartient la case. Plusieurs cases peuvent donc avoir le même indice k. La numérotation des zones k doivent se suivre, il ne peut pas y avoir un k supérieur au nombre de zones.

2.1 Les règles 1 et 2

2.1.1 Logique

Dans cette partie, nous ignorerons volontairement l'indice k car il n'a pas d'utilité.

D'un point de vue logique, ces deux règles sont proches donc peuvent être combinées en une seule et traduites de la manière suivante : "si une case est coloriée alors cette case doit avoir exactement 1 case voisine coloriée qui doit se trouver sur la même ligne ou la même colonne".

Cette règle impose donc un placement en bloc de 2x1 ou 1x2 et empêche de poser deux blocs qui se touchent par un côté.

Pour chaque case $x_{i,j}$, on obtient alors :

$$x_{i,j} \implies \big((x_{i-1,j} \land \neg x_{i,j+1} \land \neg x_{i+1,j} \land \neg x_{i,j-1}) \lor (\neg x_{i-1,j} \land x_{i,j+1} \land \neg x_{i+1,j} \land \neg x_{i,j-1}) \lor (\neg x_{i-1,j} \land \neg x_{i,j+1} \land x_{i+1,j} \land \neg x_{i,j-1}) \lor (\neg x_{i-1,j} \land \neg x_{i,j+1} \land \neg x_{i+1,j} \land x_{i,j-1})\big)$$

On peut ensuite convertir en forme normale conjonctive :

$$(\neg x_{i,j} \lor \neg x_{i-1,j} \lor \neg x_{i,j-1}) \land (\neg x_{i,j} \lor \neg x_{i-1,j} \lor \neg x_{i,j+1}) \land (\neg x_{i,j} \lor \neg x_{i-1,j} \lor \neg x_{i+1,j}) \land (\neg x_{i,j} \lor x_{i-1,j} \lor x_{i,j-1} \lor x_{i,j+1} \lor x_{i+1,j}) \land (\neg x_{i,j} \lor \neg x_{i,j-1} \lor \neg x_{i,j+1}) \land (\neg x_{i,j} \lor \neg x_{i,j-1} \lor \neg x_{i+1,j}) \land (\neg x_{i,j} \lor \neg x_{i,j+1} \lor \neg x_{i+1,j})$$

2.1.2 Algorithme

```
clauses = []
pour i allant de 1 à n:
  pour j allant de 1 à n:
    #on fait au cas par cas pour gérer les cases qui "dépassent" du plateau
    si i = 1 && j = 1: #case en haut a gauche x_{i-1,j} et x_{i,j-1} n'existent pas
      clauses = clauses + (!x_{i,j}  || x_{i,j+1} || x_{i+1,j}) && (!x_{i,j} || !x_{i,j+1} || !x_{i+1,j})
    sinon si i = 1 && j = n: #case en haut à droite x_{i-1,j} et x_{i,j+1} n'existent pas
      clauses = clauses + (!x_{i,j}  || x_{i,j-1} || x_{i+1,j}) && (!x_{i,j} || !x_{i,j-1} || !x_{i+1,j})
    sinon si i = n && j.= 1: #case en bas a gauche x_{i+1,j} et x_{i,j-1} n'existent pas
      clauses = clauses + (!x_{i,j}  || x_{i,j+1} || x_{i-1,j}) && (!x_{i,j} || !x_{i,j+1} || !x_{i-1,j})
    sinon si i = n && j = n: #case en bas à droite x_{i+1,j} et x_{i,j+1} n'existent pas
      clauses = clauses + (!x_{i,j}  || x_{i,j-1} || x_{i-1,j}) && (!x_{i,j} || !x_{i,j-1} || !x_{i-1,j})
    sinon si i = 1 #cases sur la premiere ligne x_{i-1,j} n'existe pas
      clauses = clauses + (!x_{i,j}  || x_{i,j-1} || x_{i,j+1} || x_{i+1,j}) && (!x_{i,j} || !x_{i,j-1} || !x_{i,j+1}) && (!x_{i,j} || !x_{
    sinon si i = n #cases sur la derniere ligne x_{i+1,j} n'existe pas
      clauses = clauses + (!x_{i,j}  || x_{i,j-1} || x_{i,j+1} || x_{i-1,j}) && (!x_{i,j} || !x_{i,j-1} || !x_{i,j+1}) && (!x_{i,j} || !x_{
    sinon si j = 1 #cases sur la premiere colonne x_{i, j_1} n'existe pas
      clauses = clauses + (!x_{i,j}  || x_{i+1,j} || x_{i,j+1} || x_{i-1,j}) && (!x_{i,j} || !x_{i+1,j} || !x_{i,j+1}) && (!x_{i,j} || !x_{
    sinon si j = n #cases sur la dernière colonne x_{i, j+1} n'existe pas
      clauses = clauses + (!x_{i,j}  || x_{i+1,j} || x_{i,j-1} || x_{i-1,j}) && (!x_{i,j} || !x_{i+1,j} || !x_{i,j-1}) && (!x_{i,j} || !x_{
    sinon #toutes les autres cases
      clauses = clauses + clause trouvée en 2.1.1
retourner clauses
```

2.2 La règle 3

2.2.1 Logique

Afin de vérifier qu'il existe seulement 2 cases coloriées par zone, on peut prendre 2 cases appartenant à la même zone, puis si les 2 sont vraies (coloriées), alors toutes les autres cases de la zone doivent être fausses (non coloriées).

D'un point de vue logique, on obtient alors :

$(x_{i_1,j_1,k} \wedge x_{i_2,j_2,k}) \implies (\neg x_{i_3,j_3,k} \wedge \neg x_{i4,j4,k} \wedge$ .....toutes les cases d'indice k) avec $i_1 \neq i_2 \neq i_3 \neq i_4$.... et $j_1 \neq j_2 \neq j_3 \neq j_4$.....

On peut ensuite convertir en forme normale conjonctive

$(\neg x_{i_1,j_1,k} \vee \neg x_{i_2,j_2,k} \vee \neg x_{i_3,j_3,k}) \wedge (\neg x_{i_1,j_1,k} \vee \neg x_{i_2,j_2,k} \vee \neg x_{i_4,j_4,k}) \wedge (\neg x_{i_1,j_1,k} \vee \neg x_{i_2,j_2,k} \vee \neg x_{i_5,j_5,k}) \wedge$ ..... avec toutes les cases restantes d'indice k

On effectue ensuite la conjonction de toutes les clauses générées pour chaque paire de cases dans la zone k, puis enfin la conjonction avec les clauses de chaque zone pour $k \in [1; \lfloor \frac{n^2}{2} \rfloor]$. Nous avons donc désormais toutes les clauses de la règle 3

2.2.2 Algorithme

Algorithme rudimentaire à paufiner :

```
#on recupere les clauses de la zone k = 1
clauses = []
cases = []
pour i allant de 1 à n: # on récupère d'abord toutes les cases situées dans la zone k
  pour j allant de 1 à n:
    si le k de x_{i,j,k} = 1 alors:
      cases = cases + x_{i,j,k}

#on passe désormais à la génération de clauses
pour case1 parmi cases:
  pour case2 parmi cases:
    si case1 ≠ case2:
      pour case3 parmi cases:
        si case3 ≠ case1 && case3 ≠ case2:
          clauses = clauses + (!case1 || !case2 || !case3)
retourner clauses

#on répète ensuite l'opération avec différents k pour couvrir toutes les zones k dans kEns
```