TP3 : Les reines

1 Introduction

Le problème des reines. Ce problème[1] consiste à placer n reines sur un échiquier de taille $n \times n$ sans qu'elles puissent se menacer entre elles. C'est à dire que pour chaque reine, il ne doit pas y en avoir une autre sur la même ligne, ni sur la même colonne, ni sur les mêmes diagonales qu'elle. Sur un échiquier de taille 8×8 il y a 12 solutions possibles. Au-delà de la taille 26×26, la question est encore ouverte pour trouver toutes les solutions. Dans la Figure 1, une solution pour le cas $n = 5$ est donnée :

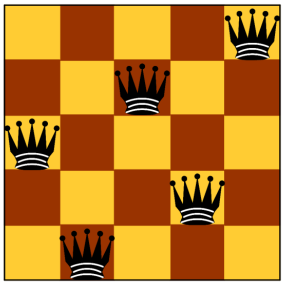

FIGURE 1

Il y a plusieurs approches possibles pour résoudre ce problème. Dans ce TP, nous proposons de représenter le problème des reines comme une grosse expression booléenne, et de trouver ses *modèles* — autrement dit, ses solutions.

SAT-solveurs. Un **SAT-solveur** est un programme informatique dont le but est de résoudre des expressions booléennes. On fournit au programme une certaine expression booléenne $f(x_1, \dots, x_n)$, et le programme détermine si cette expression est **satisfaisable**, c'est-à-dire s'il existe une assignation des variables $(x_1, \dots, x_n)$ pour laquelle $f(x_1, \dots, x_n) = 1$.
Il existe un SAT-solveur installé sur les machines de l'IUT : le programme **picosat**. Il existe d'autres SAT-solveurs connus (Minisat, Z3, etc.) Le format des fichiers d'entrées des SAT-solveurs est le format DIMACS. C'est le format utilisé pour "communiquer" au SAT-solveur l'expression de la fonction booléenne f qu'il doit étudier.

1. `https://oeis.org/A000170`

Rappels sur le format DIMACS. C'est un standard international pour la représentation de formules en forme normale conjonctive. Un fichier en format DIMACS commence par une ligne qui spécifie qu'il s'agit d'une forme normale conjonctive, qui précise combien de variables la formule contient, et de combien de clauses disjonctives elle est constituée. Par exemple : `p cnf 5 3` indique que le fichier contient une formule en forme normale conjonctive, avec 5 variables et 3 clauses. Ensuite, le fichier est composé de plusieurs lignes, une par clause. Chaque ligne contient des entiers positifs et/ou négatifs, et se termine par 0. Un entier positif i indique que la i-ème variable apparaît avec polarité positive dans la clause. Un entier négatif $-i$ indique que la i-ème variable apparaît avec polarité négative dans la clause. Voici un fichier exemple :

```
c Exemple de fichier Dimacs
c (les lignes commençant par c sont des commentaires)

p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

Ce fichier représente donc la formule (à 5 variables booléennes)

$$(x_1 + \overline{x_5} + x_4).(\overline{x_1} + x_5 + x_3 + x_4).(\overline{x_3} + \overline{x_4})$$

Modélisation booléenne du problème des reines. Dans ce TP nous considérons d'abord le cas $n = 4$, c'est-à-dire : 4 reines à placer sur un échiquier de taille 4×4. Nous numérotons les cases de l'échiquier de 1 à 16, de gauche à droite et ligne par ligne, et nous introduisons les variables propositionnelles (booléennes) suivantes :

- X_k : Une reine se trouve sur la case k.

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

2 Exercices

Les exercices 1 à 4 concernent la modélisation du problème, et l'exercice 5 sa résolution à l'aide d'un SAT-solveur. Pour ces exercices, vous n'avez pas besoin d'écrire de programmes en Python. Par contre, une feuille de papier et un stylo s'avèreront utiles. (D'ailleurs pourquoi je le rappelle ? C'est le cas pour *tous* vos TPs de math !)

Exercice 1 (Contraintes de lignes).

1. Écrivez une **clause** (c'est-à-dire une somme de certaines variables X_k et $\overline{X}_k$) qui vaut 0 si les cases 1 et 2 sont toutes les deux occupées, et 1 sinon.
2. Écrivez un *produit de clauses* exprimant le fait que la première ligne doit contenir au plus une reine. (À vous de déterminer combien de clauses sont nécessaires dans ce produit.)
3. Écrivez un produit de clauses traduisant le fait que *toutes les lignes* doivent contenir au plus une reine chacune. Traduisez ces clauses dans un fichier Dimacs.
   Indice : vous devez écrire un total de 24 clauses, soit 24 lignes dans le fichier DIMACS.

Exercice 2 (Contraintes de colonnes).
Sur le même modèle que l'exercice 1, rajoutez à votre fichier DIMACS un ensemble de clauses exprimant le fait que chaque *colonne* doit contenir au plus une reine.
Indice : vous devez écrire un total de 24 clauses, soit 24 lignes supplémentaires dans le fichier DIMACS.

Exercice 3 (Contraintes de diagonales).
Sur le même modèle que l'exercice 1, rajoutez à votre fichier DIMACS un ensemble de clauses exprimant le fait que chaque *diagonale* doit contenir au plus une reine.
Indice : vous devez écrire un total de 28 clauses, soit 28 lignes supplémentaires dans le fichier DIMACS.

Exercice 4 (Contraintes de présence des reines).
Pour le moment, les contraintes que nous avons décrites n'obligent pas à mettre des reines sur l'échiquier ! En effet, elles sont toutes vraies lorsque toutes les variables prennent la valeur 0. Il faut maintenant forcer certaines variables à valoir 1.
Pour cela, écrivez 4 clauses (une par ligne) qui indiquent que chaque ligne doit contenir une reine.

Exercice 5 (Résolution par SAT-solveur).
À ce stade, votre fichier DIMACS contient un encodage du problème des 4 reines. Vous pouvez maintenant utiliser un SAT-solveur (picosat, Minisat ou Z3) pour le résoudre. Par exemple, dans un terminal avec la commande

```
picosat reines.cnf --all
```

(où `reines.cnf` désigne le fichier DIMACS contenant l'encodage des contraintes).

- Commentez le résultat, trouvez les deux solutions pour ce problème.
- Au vu des résultats, comment pourriez-vous améliorer votre modèle ?

Exercice 6 (Le problème à n reines).
Pour le problème à 4 reines ci-dessus, il y avait un total de 24+24+28+4=80 clauses ; vous avez donc pu écrire le fichier DIMACS correspondant à la main. Mais si la taille n de l'échiquier augmente, le nombre de clauses augmente rapidement, et une approche à la main n'est plus possible. Le but de cet exercice est donc d'écrire un programme Python (aah enfin !) qui *génère automatiquement le fichier DIMACS* représentant le problème à n reines.

1. Question préliminaire. Allez voir sur internet comment *écrire dans un fichier texte* avec Python (par exemple dans la doc officielle). Remarque : il y a plusieurs méthodes disponibles ; la méthode la plus sûre est celle basée sur la syntaxe « `with open("filename") as f` », mais les autres méthodes sont aussi acceptables.

   Afin de tester votre méthode, créez depuis Python un fichier texte intitulé `supercalcul.txt` et qui contient les 2 lignes suivantes :

   ```
   Le nombre 37/19 vaut approximativement:
   [sur cette ligne, l'écriture décimale du nombre 37/19]
   ```

2. On numérote toujours les cases de l'échiquier selon la convention précédente : de gauche à droite et ligne par ligne.
   - Combien va-t-il y avoir de variables booléennes dans le problème à n reines ?
   - Si une case se trouve à la ligne i et à la colonne j dans un échiquier de taille n, calculez son numéro de case, en fonction de n, i et j.
   - Implémentez cette relation sous la forme d'une fonction

     ```
     def case(i,j):
     ```

     qui renvoie le numéro k de la case située en position (i, j). Cette fonction vous sera utile pour la question suivante.

3. Écrivez un programme Python qui, étant donnée une certaine taille d'échiquier n, génère automatiquement un fichier DIMACS représentant les contraintes du problème à n reines. Testez ensuite un SAT-solveur sur le fichier que vous venez de générer. Combien y a-t-il de solutions en fonction de n ? À partir de quelle taille n le SAT-solveur commence-t-il à "ramer" ?