Résolution de problèmes à l'aide d'un solveur SAT

Contents

1 Introduction **2**

2 Eléments de corrections pour les séances 1 et 2 **2**
- 2.1 La licorne . . . 2
  - 2.1.1 Modélisation . . . 2
  - 2.1.2 Validation de la cohérence . . . 3
  - 2.1.3 Questions . . . 3
- 2.2 Coloriage d'un graphe . . . 3
  - 2.2.1 Définition du problème . . . 3
  - 2.2.2 Contraintes . . . 3
  - 2.2.3 Implémentation en Python . . . 3

3 Sudoku **4**
- 3.1 Problème . . . 4
- 3.2 Clauses DIMACS . . . 4

4 Planification à horizon fini : Sokoban simplifié **4**
- 4.1 Variables . . . 4
- 4.2 Formulation des contraintes . . . 5
- 4.3 Clauses DIMACS . . . 5

5 Conclusion **5**

1 Introduction

La résolution de problèmes complexes à l'aide de solveurs SAT (Satisfiability) repose sur la transformation des contraintes en formules logiques exprimées sous forme de clauses. Ce rapport détaille plusieurs exemples pratiques illustrant cette méthode, notamment l'analyse d'une licorne mythique, le coloriage de graphes, la résolution de Sudoku et la planification dans des systèmes dynamiques comme le Sokoban.

2 Eléments de corrections pour les séances 1 et 2

2.1 La licorne

2.1.1 Modélisation

Si la licorne est mythique, alors elle est immortelle ; si elle n'est pas mythique, c'est un mammifère mortel. Si elle est un mammifère ou immortelle, voire les deux, elle possède une corne. Une licorne est magique si elle possède une corne.

Les variables utilisées sont :

- 1 : Mythique (M)
- 2 : Mortelle (T)
- 3 : Mammifère (A)
- 4 : Corne (C)
- 5 : Magique (G)

La base de connaissances (KB) est traduite en clauses DIMACS :

```
-1 -2 0
1 2 0
1 3 0
-3 4 0
2 4 0
-4 5 0
```

2.1.2 Validation de la cohérence

L'application d'un solveur SAT montre que la KB est satisfiable. Il existe donc un modèle possible pour un monde où les licornes existent.

2.1.3 Questions

1. **La licorne a-t-elle une corne ?**

   En ajoutant la clause $-4\ 0$, le solveur retourne `UNSAT`. Par conséquent, la licorne doit avoir une corne.

2. **La licorne est-elle mythique ?**

   La résolution de $KB \cup \{1\ 0\}$ et $KB \cup \{-1\ 0\}$ montre que les deux formules sont satisfiables. Nous ne pouvons donc pas conclure si la licorne est mythique ou non.

2.2 Coloriage d'un graphe

2.2.1 Définition du problème

On considère le graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ où :

- $\mathcal{V} = \{A, B, C, D\}$
- $\mathcal{E} = \{\{A, B\}, \{A, C\}, \{B, C\}, \{B, D\}, \{C, D\}\}$

Les couleurs sont $K = \{r, v, b\}$. On utilise des variables propositionnelles $x_{v,k}$ pour indiquer si un sommet v a la couleur k.

2.2.2 Contraintes

- Chaque sommet $v \in \mathcal{V}$ a au moins une couleur : $\bigvee_{k \in K} x_{v,k}$
- Chaque sommet a au plus une couleur : $\neg x_{v,k} \vee \neg x_{v,k'}$ pour $k \neq k'$
- Deux sommets reliés par une arête ont des couleurs différentes : $\neg x_{v,k} \vee \neg x_{v',k}$

2.2.3 Implémentation en Python

```
import subprocess

def exec_sat_solver(filename, cmd="gophersat", encoding="utf8"):
    result = subprocess.run([cmd, filename], stdout=subprocess.PIPE, check=True,
```

```
string = str(result.stdout)
lines = string.splitlines()

if lines[1] != "s SATISFIABLE":
    return False, {}

model = lines[2][2:].split()

return True, { i2v[abs(int(v))] : int(v) > 0 for v in model if int(v) != 0}
```

3 Sudoku

3.1 Problème

La résolution du Sudoku peut être vue comme un problème de coloriage de graphe. Les cellules représentent les sommets, et les contraintes sur les lignes, colonnes et blocs 3x3 définissent les arêtes. Les couleurs représentent les valeurs possibles (1 à 9).

3.2 Clauses DIMACS

- Chaque cellule contient un chiffre.
- Chaque chiffre apparaît au plus une fois par ligne, colonne et bloc.

L'initialisation du problème injecte les informations de la grille sous forme de clauses unaires.

4 Planification à horizon fini : Sokoban simplifié

4.1 Variables

- $w_{c,t}$: Le personnage est sur la case c à l'instant t.
- $b_{c,t}$: Une caisse est sur la case c à l'instant t.
- $do_{x,t}$: L'action x est réalisée à l'instant t.

4.2 Formulation des contraintes

1. **État initial** : Décrit la position initiale des caisses et du personnage.
2. **État final** : Décrit la position cible des caisses.
3. **Transitions** : Exprime les effets des actions sur l'état du monde.

4.3 Clauses DIMACS

Les contraintes sont traduites sous forme de clauses et générées dynamiquement en fonction de l'horizon T.

5 Conclusion

Ce rapport illustre l'utilisation de solveurs SAT pour résoudre divers problèmes combinatoires, en transformant des contraintes complexes en formules logiques sous forme de clauses DIMACS. Cette méthode montre son efficacité et sa généralité, que ce soit pour modéliser des problèmes théoriques ou pour des applications pratiques comme le Sudoku ou Sokoban.