

# Coloration de graph avec SAT

LAURENT Thomas

Master 2 informatique 2019

## 0.1 Ce que contient l'archive

**\_\_main\_\_.py** Le programme qui va parser les graphs en format DIMACS.

**model\_printer.py** Va correctement formater et afficher le modèle obtenue via le solveur SAT.

**tmp.json** Un fichier json contenant un graph (notamment celui de l'Australie).

**sat\_graph.sh** Pour automatiser le déroulement des algorithmes.

**README.pdf** Coucou c'est moi.

## 0.2 Format du graph dans le fichier json

```
{  
  "variables" : ["wa", "nt", "sa", "gld", "nsw", "vc", "tas"],  
  "constraints": [["wa", "nt"], ["wa", "sa"], ..., ["nsw", "vc"]]  
}
```

**variables** étant une liste de tout les noms des points présent dans le graph.

**constraints** la représentation de toutes les arêtes du graph.

## 0.3 Utilisation du script

Voici un exemple:

```
./sat_graph MiniSat_v1.14_linux tmp.json 3
```

**MiniSat\_v1.14\_linux** Est le solveur utilisé en question MiniSat.

**tmp.json** Un fichier au format json valide et étant similaire à l'exemple du dessus.

**3** étant le nombre de couleurs que vous voudrez donner à l'algo. (il n'y a pas vérification si négatif si non numérique)

Le script, sat et python vont générer deux fichiers lors de leurs exécution:

**sat.in** Qui sera le fichier formaté en DIMACS qui est créer par python et lu par le solveur. (le fichier contient 2 lignes de commentaires indiquant le noms des variables et leurs ids, ceci pour aider le *model\_printer*).

**sat.out** Qui sera le modèle de base qui est écrit par le solveur et lu par le *model\_printer* (je ne garantis pas que tout les solveur pourront être lu par le *model\_printer*).

## 0.4 Exemple satisfiable

```
./sat_graph MiniSat_v1.14_linux tmp.json 3
```

```
===== [MINISAT] =====
| Conflicts | ORIGINAL | LEARNT | Progress |
| | Clauses Literals | Limit Clauses Literals Lit/Cl | |
=====
| 0 | 55 117 | 18 0 0 nan | 0.000 % |
=====

restarts : 1
conflicts : 2 (3101 /sec)
decisions : 11 (17054 /sec)
propagations : 51 (79070 /sec)
conflict literals : 5 (0.00 % deleted)
Memory used : 1.68 MB
CPU time : 0.000645 s

SATISFIABLE
===== Model =====
wa prendra la couleur 1
nt prendra la couleur 2
sa prendra la couleur 0
gld prendra la couleur 1
nsw prendra la couleur 2
vc prendra la couleur 1
tas prendra la couleur 2
```

## 0.5 Exemple satisfiable

*./sat\_graph MiniSat\_v1.14\_linux tmp.json 2*

```
===== [MINISAT] =====
| Conflicts | ORIGINAL | LEARNT | Progress |
| | Clauses Literals | Limit Clauses Literals Lit/Cl | |
=====
| 0 | 32 64 | 10 0 0 nan | 0.000 % |
=====

restarts : 1
conflicts : 2 (3101 /sec)
decisions : 1 (17054 /sec)
propagations : 10 (79070 /sec)
conflict literals : 1 (0.00 % deleted)
Memory used : 1.68 MB
CPU time : 0.000645 s

UNSATISFIABLE
```