Méthodes et modélisation pour l'optimisation

M1 informatique, 2016–2017

05 — Modélisation SAT

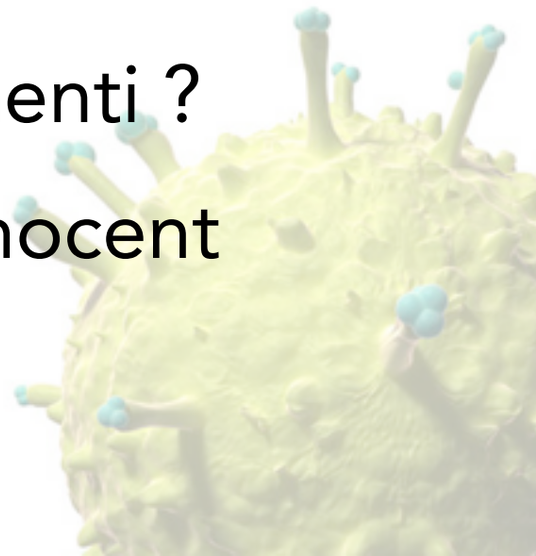
UNIVERSITÉ PARIS-EST
MARNE-LA-VALLÉE

INSTITUT D'ÉLECTRONIQUE
ET D'INFORMATIQUE
GASPARD-MONGE

Mystère

- Trois étudiants Jean, Paulette et Nicolas sont accusés d'avoir introduit un virus dans la salle informatique. Pendant leur audition, ils prétendent ceci :
  - Jean: "*C'est Paulette qui l'a fait et Nicolas est innocent.*"
  - Paulette: "*Si Jean est coupable, alors Nicolas l'est aussi.*"
  - Nicolas: "*Ce n'est pas moi. C'est un des deux autres ou peut-être les deux.*"

1. Est-ce que les trois déclarations sont contradictoires ?
2. Supposons qu'ils soient tous coupables, qui a menti ?
3. Supposons que personne n'ait menti, qui est innocent et qui est coupable ?



Vocabulaire

- Variable booléenne : x prend la valeur 1 (vrai) ou 0 (faux)
- Formule propositionnelle : expression constituée de
  - variables x
  - négations ¬x “NON x”
  - conjonctions $x \wedge y$ “x ET y”
  - disjonctions $x \vee y$ “x OU y”
  - implications $x \rightarrow y$ “SI x, ALORS y”
- Une affectation satisfaisante d’une formule est une affectation de valeurs (0 ou 1) aux variables qui rend la formule vraie
- Une formule est satisfaisable si elle a une affectation satisfaisante

Forme normale conjonctive

- Littéral : x, ¬x
- Clause (disjonction de littéraux) : x ∨ y ∨ ¬z
- Formule en forme normale conjonctive (CNF)
  (conjonction de clauses) : (x ∨ y ∨ ¬z) ∧ (¬x ∨ ¬y ∨ ¬z)

Exercice : quelles formules sont en CNF ?

A. (¬x ∨ ¬y) ∧ (¬z ∨ ¬w)

B. x → y

C. ¬x ∧ y

D. (x ∧ y) ∨ (z ∧ w)

E. (¬(x ∨ y)) ∧ (z ∨ w)

Mise en forme CNF

- Distributivité : $x \vee (y \wedge z) \Leftrightarrow (x \vee y) \wedge (x \vee z)$
- de Morgan : $\neg(x \vee y) \Leftrightarrow \neg x \wedge \neg y$
- Implication : $(x \rightarrow y) \Leftrightarrow \neg x \vee y$
- Associativité : $x \vee (y \vee z) \Leftrightarrow (x \vee y) \vee z \Leftrightarrow x \vee y \vee z$

Exercice : mettez les formules suivantes en CNF

A. $(x \wedge y) \vee (z \wedge w)$

B. $\neg(x \rightarrow y)$

C. $(\neg(x \vee y)) \wedge (z \rightarrow w)$

Le problème SAT

- Le problème de trouver une affectation satisfaisante à une formule CNF ou de démontrer qu'il n'y en a aucune s'appelle SAT (satisfaisabilité booléenne)
- SAT est NP-difficile ; tout problème dans NP peut être exprimé comme une formule en CNF
- On décrit souvent l'entrée à SAT par un ensemble de clauses : $(x_1 \vee \neg x_2)$, $(\neg x_1 \vee x_3)$, $(\neg x_2 \vee \neg x_3)$ au lieu de la formule $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$
- Alors, SAT est le problème de trouver une affectation aux variables qui rend toutes les clauses vraies simultanément

Mystère en CNF

"C'est Paulette qui l'a fait et Nicolas est innocent."

$$x_P,\ \neg x_N$$

"Si Jean est coupable, alors Nicolas l'est aussi."

$$\neg x_J \vee x_N$$

"Ce n'est pas moi. C'est un des deux autres ou peut-être les deux."

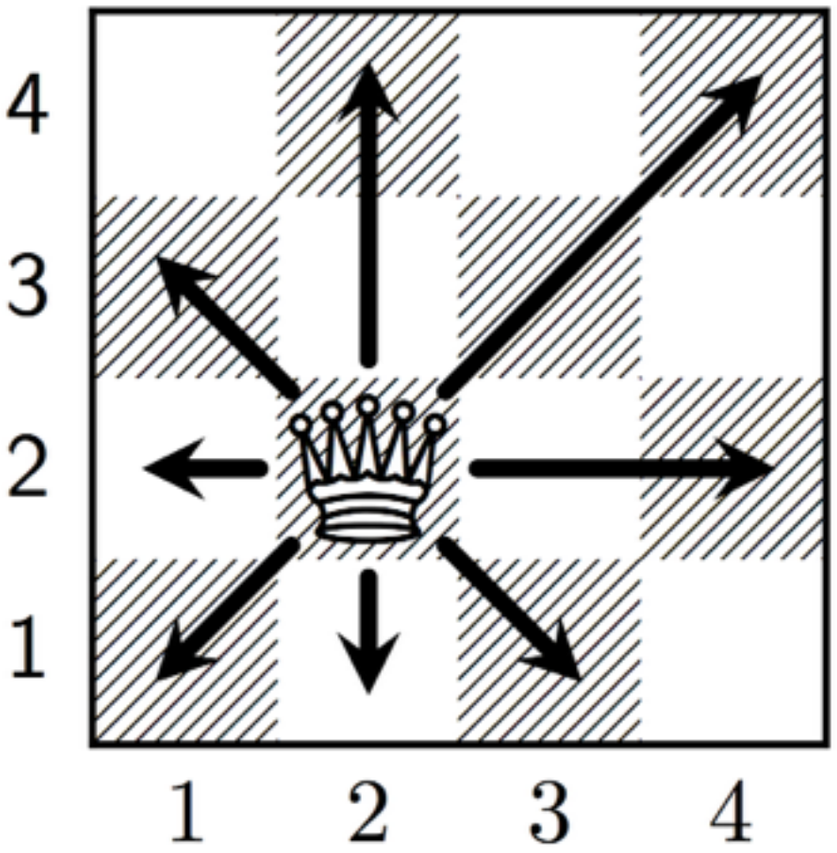
$$\neg x_N,\ x_P \vee x_J$$

La CNF a 4 clauses

$x_P,\ \neg x_N,\ \neg x_J \vee x_N,$

$x_P \vee x_J$

4 reines ♕



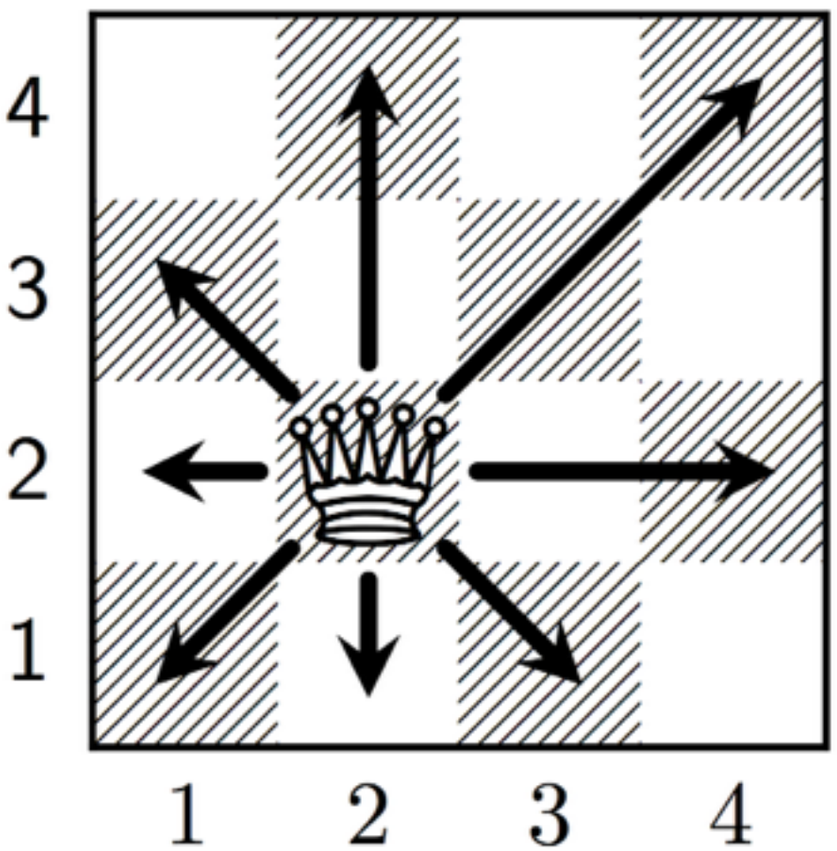
Est-ce qu'il est possible de placer les 4 reines de manière à ce qu'aucune d'entre elles ne soit en prise ?

- Modéliser ce problème par une formule CNF
- C-à-d, trouver une formule CNF telle que chaque solution satisfaisante correspond à un placement valide de 4 reines

4 reines ♕

- Variables x_{ij}

  intention : x_{ij} = 1 s'il y a une reine en rang i, colonne j



1. Au moins une reine dans chaque rang

$$(x_{i1} \vee x_{i2} \vee x_{i3} \vee x_{i4})$$

$i = 1, \ldots, 4$ (4 clauses)

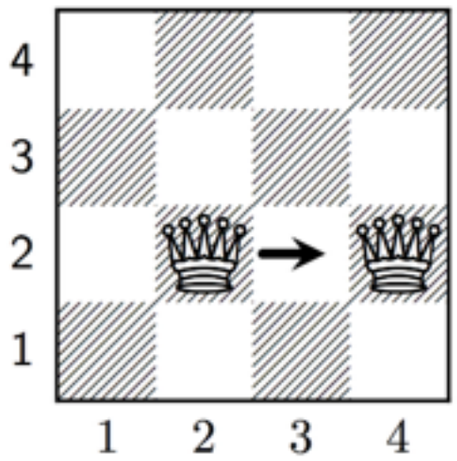
2. Il n'y a pas deux reines à (a,b) et à (c,d)
   si une reine à (a,b) peut attaquer (c,d)

$$(\neg x_{ab} \vee \neg x_{cd})$$

(beaucoup de clauses)

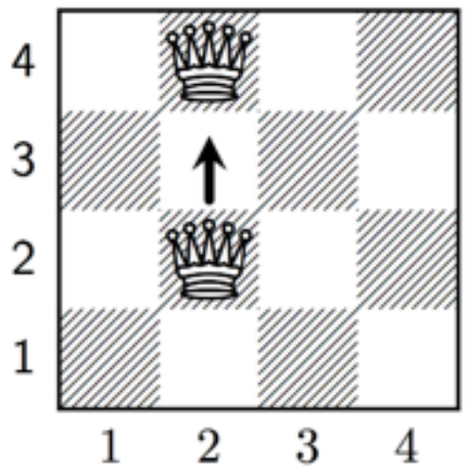
4 reines ♕

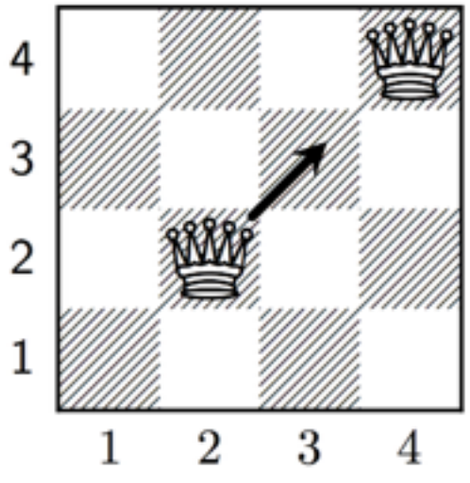
- Quand est-ce qu'une reine à (a,b) peut attaquer (c,d) ?



Attaque horizontale : a = c, b ≠ d

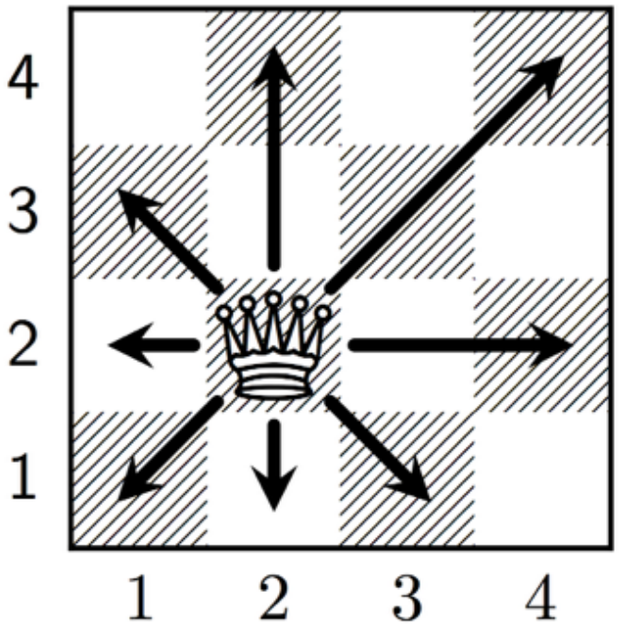
Attaque verticale : a ≠ c, b = d





Attaque diagonale : |a - c| = |b - d|

4 reines en Python



1. Il y a au moins une reine dans chaque rang

```
for i in range(1,5):
  clause = [var(i,j) for j in range(1,5)]
  formula.append(clause)
```

2. Il n'y a pas deux reines à (a,b) et à (c,d) si une reine à (a,b) peut attaquer (c,d)

```
for a in range(1,5):
  for b in range(1,5):
    for c in range(1,5):
      for d in range(1,5):
        if (a,b) < (c,d): # ordre lexicographique
          if ((a == c) or # horizontale
              (b == d) or # verticale
              (abs(a-c) == abs(b-d))): # diagonale
            clause = [neg(var(a,b)),neg(var(c,d))]
            formula.append(clause)
```

Macros

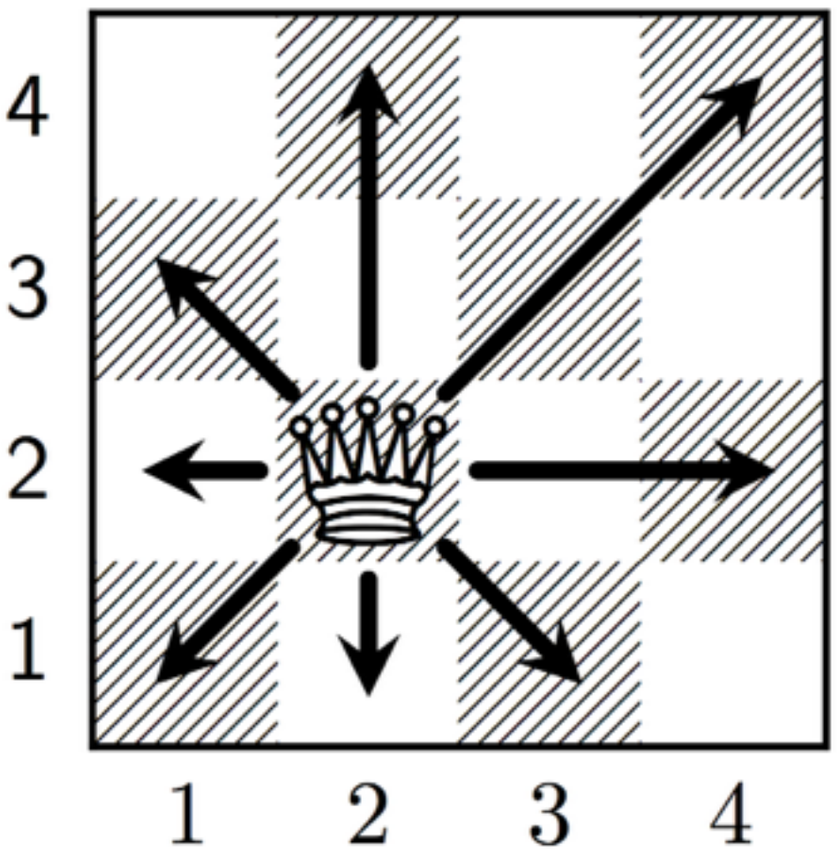
Il y a au moins une variable parmi $x_1, x_2, \ldots, x_n$ qui est vraie

$$\text{AtLeastOne}(x_1, x_2, \ldots, x_n)$$
$$\Leftrightarrow$$
$$(x_1 \vee x_2 \vee \ldots \vee x_n)$$

Il y a au plus une variable parmi $x_1, x_2, \ldots, x_n$ qui est vraie

$$\text{AtMostOne}(x_1, x_2, \ldots, x_n)$$
$$\Leftrightarrow$$
$$(\neg x_1 \vee \neg x_2), (\neg x_1 \vee \neg x_3), \ldots, (\neg x_{n-1} \vee \neg x_n)$$

4 reines ♕



1. Au moins une reine dans chaque rang

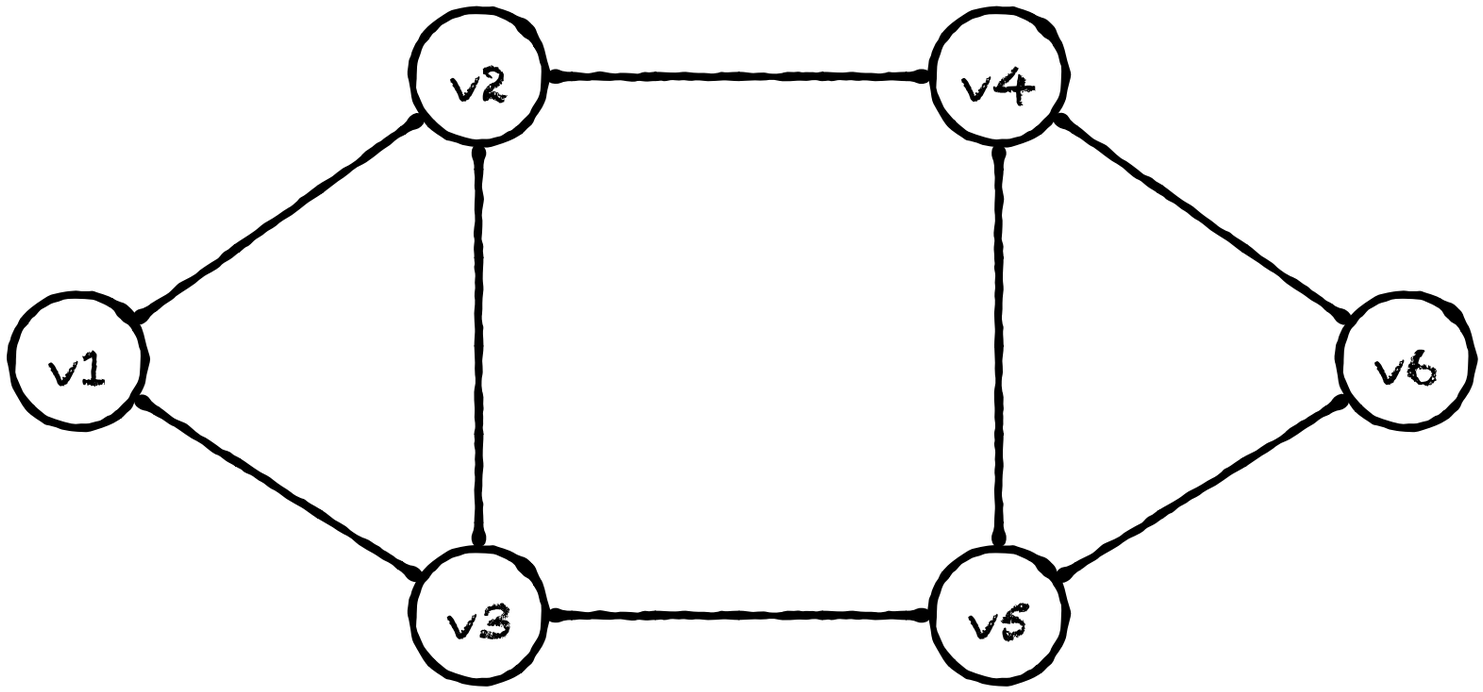
   AtLeastOne(x_{i1}, x_{i2}, x_{i3}, x_{i4})

2. Il n'y a pas deux reines à (a,b) et à (c,d) si une reine à (a,b) peut attaquer (c,d)

   AtMostOne(x_{ab}, x_{cd})

On se permet d'utiliser les macros AtLeastOne et AtMostOne au td et à l'examen.

Coloration de graphe

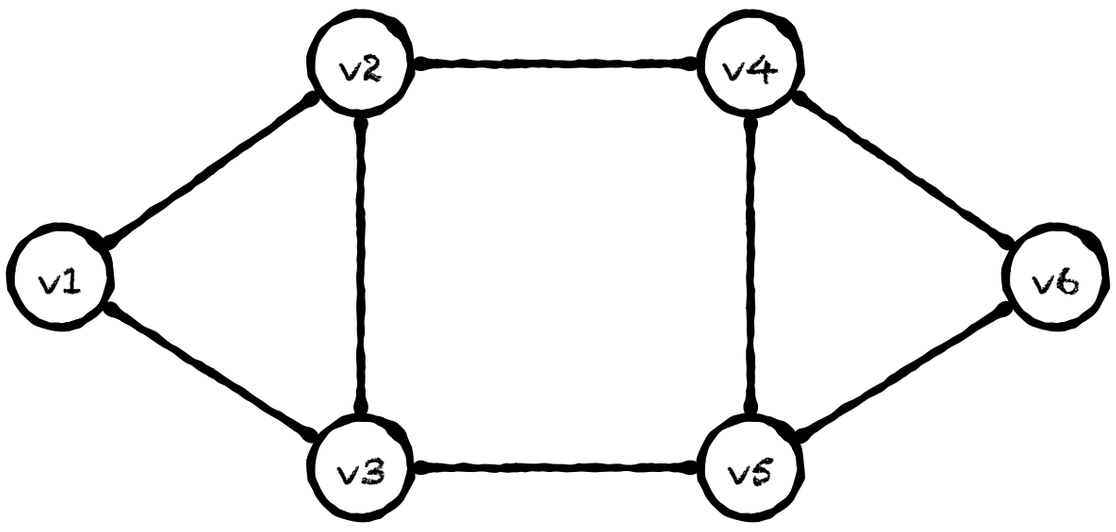


- Le graphe a-t-il une coloration avec trois couleurs ?
- Couleurs : 1, 2, 3
- Introduire une variable x_{ic} pour toute $i = 1, \ldots, 6$ et $c = 1, 2, 3$
- La variable $x_{ic} = 1$ indique que le sommet v_i prend la couleur c

Coloration de graphe

- Variables x_{ic}

  intention : $x_{ic} = 1$ si le sommet v_i prend la couleur c



1. Chaque sommet $i = 1, \ldots, 6$ prend au moins une couleur

   AtLeastOne(x_{i1}, x_{i2}, x_{i3})

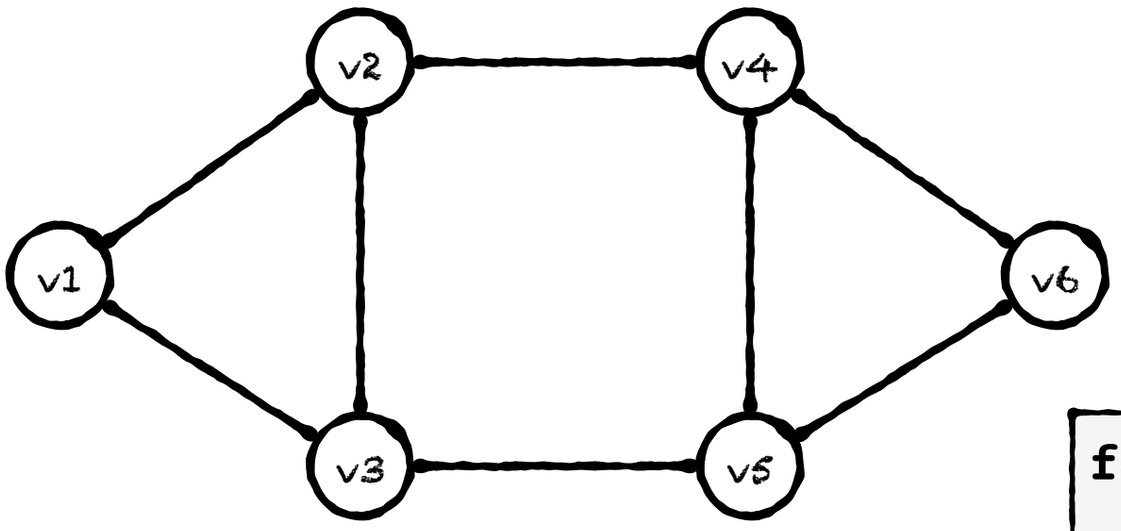
2. Chaque sommet $i = 1, \ldots, 6$ prend au plus une couleur

   AtMostOne(x_{i1}, x_{i2}, x_{i3})

3. Les deux sommets d'une arête (v_i, v_j) ne prennent pas la même couleur c

   AtMostOne(x_{ic}, x_{jc})

Coloration de graphe en Python



1. Chaque sommet i = 1, …, 6
   prend au moins une couleur
2. Chaque sommet i = 1, …, 6
   prend au plus une couleur

```
for i in nodes:
  vars = [var(i,0),var(i,1),var(i,2)]
  clauses = at_least_one(vars)
  formula.extend(clauses)
  clauses = at_most_one(vars)
  formula.extend(clauses)
```

3. Les deux sommets d'une arête (v_i, v_j)
   ne prennent pas la même couleur c

```
for e in edges:
  i, j = e
  for c in range(3):
    clauses = at_most_one([var(i,c), var(j,c)])
    formula.extend(clauses)
```

Emploi du temps

Cours	Enseignant
MMPO	Thapper
Complexité	Nicaud
MMPO	Hubard
Compression	Nicaud
...	...

	Lun	Mar	Mer	Jeu	Ven
08h30 10h30					
10h45 12h45					
14h00 16h00					
16h15 18h15					

Disponibilités	
Thapper	lun, mar, mer 10h45-16h00
Hubard	lun, ven
...	...