Méthodes et modélisation pour l'optimisation

M1 informatique, 2019–2020

05 — Modélisation SAT

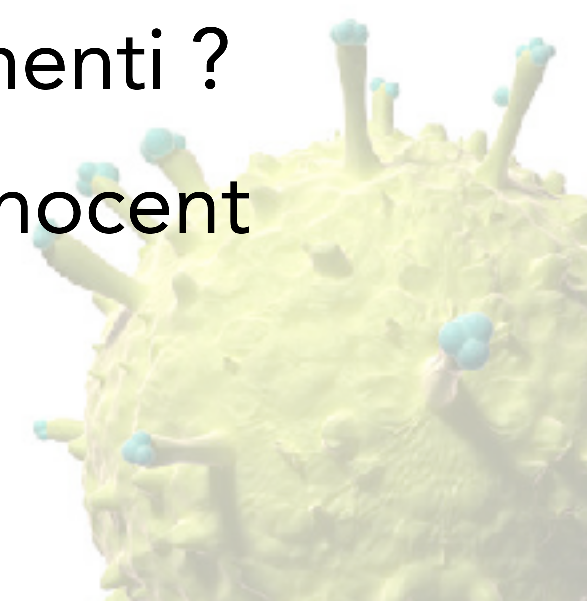
UNIVERSITÉ PARIS-EST
MARNE-LA-VALLÉE

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 $\neg x$ "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 variables aux valeurs (1 ou 0) qui rend la formule vraie (1)
- Une formule est satisfaisable si elle a une affectation satisfaisante

Forme normale conjonctive

- Littéral : x, $\neg x$
- Clause (disjonction de littéraux) : $x \lor y \lor \neg z$
- Formule en forme normale conjonctive (CNF)
  (conjonction de clauses) : $(x \lor y \lor \neg z) \land (\neg x \lor \neg y \lor \neg z)$

Exercice : quelles formules sont en CNF ?

A. $(\neg x \lor \neg y) \land (\neg z \lor \neg w)$

B. $x \rightarrow y$

C. $\neg x \land y$

D. $(x \land y) \lor (z \land w)$

E. $(\neg(x \lor y)) \land (z \lor w)$

Mise en forme CNF

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

Exercice : mettez les formules suivantes en CNF

A. $(x \land y) \lor (z \land w)$

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

C. $(\neg(x \lor y)) \land (z \rightarrow w)$

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 (relativement) petite 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."

$$\neg x_J \lor x_N$$

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

$$\neg x_N,\ x_P \lor x_J$$

La CNF a 4 clauses

$$x_P,\ \neg x_N,\ \neg x_J \lor x_N,$$

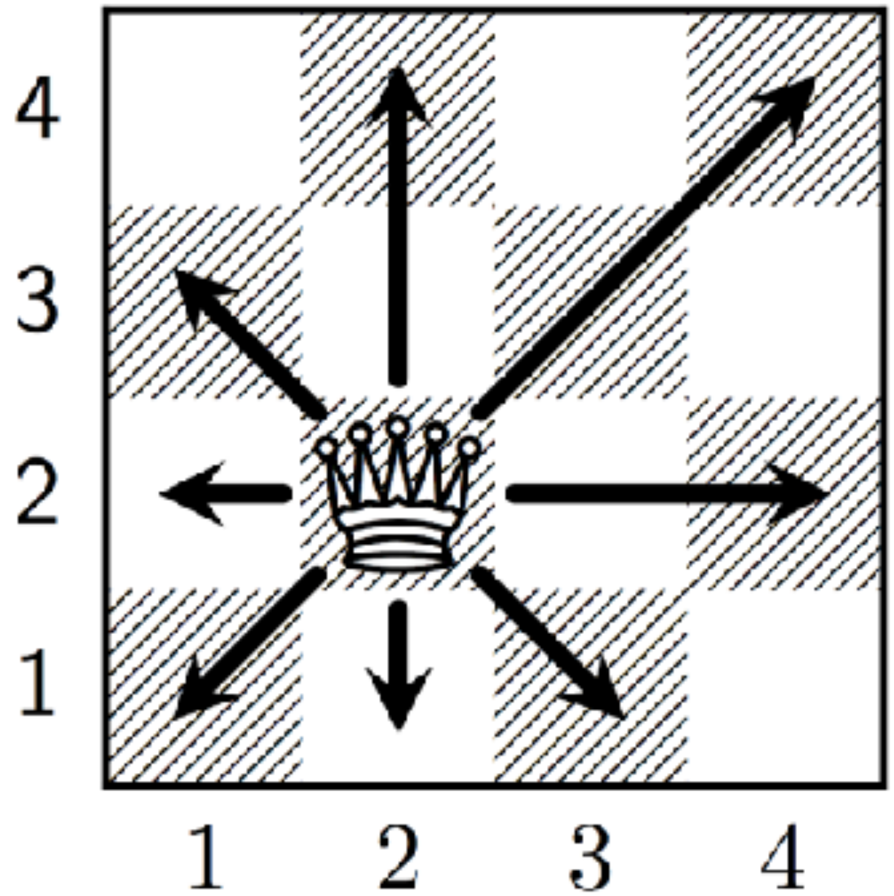
$$x_P \lor x_J$$

Mystère en CNF

Jean	x_P, $\neg x_N$
Paulette	$\neg x_J \vee x_N$
Nicolas	$\neg x_N$, $x_P \vee x_J$

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 ?

4 reines ♕



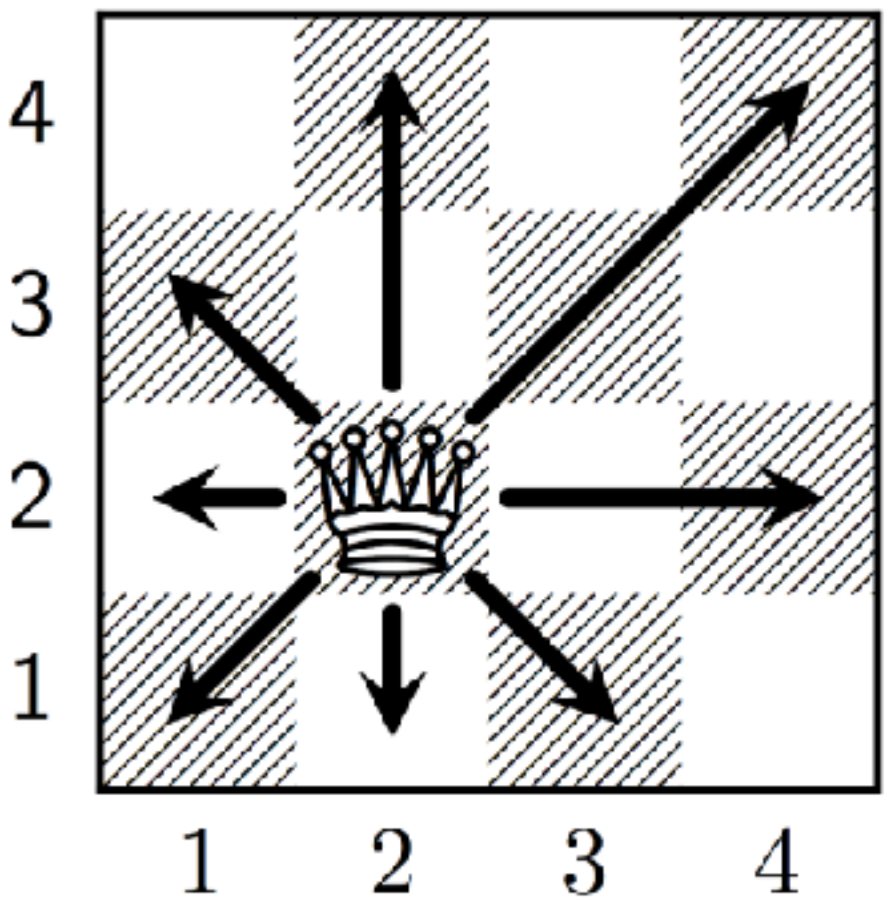
Est-ce qu'il est possible de placer les 4 reines de manière à ce qu'aucune d'entre elles ne soit en prise ?

- 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 : $x_{ij} = 1$ si la reine i est en rang i, colonne j



1. Au moins une reine dans chaque colonne

   $$(x_{1j} \vee x_{2j} \vee x_{3j} \vee x_{4j})$$

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

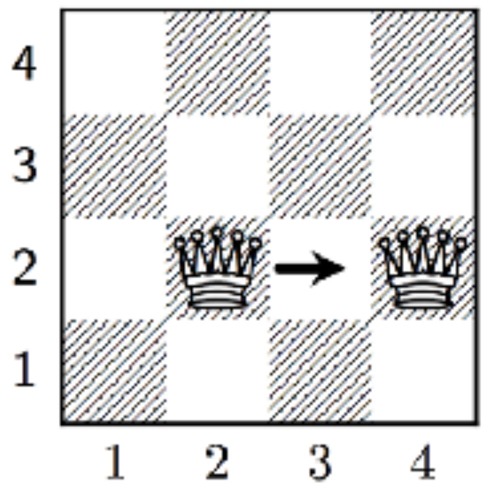
2. Il n'y a pas deux reines à (a,b) et à (c,d) si une reine à (a,b) peut attaquer (c,d)

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

   (beaucoup de clauses)

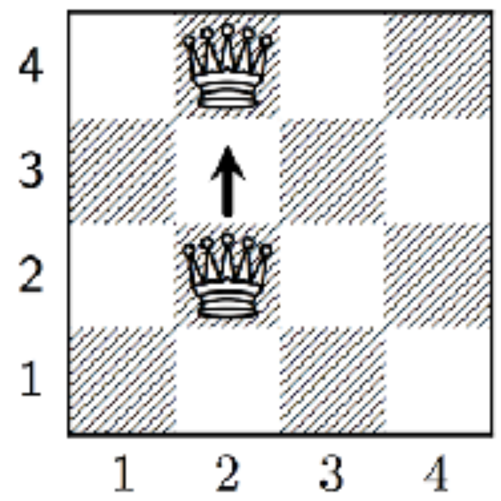
4 reines ♛

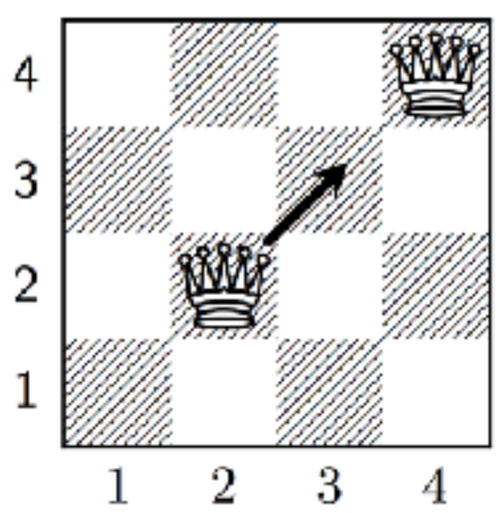
- Quand est-ce qu'une reine à (a,b) peut attaquer (c,d) ?



Attaque horizontale : $a = c$, $b \neq d$

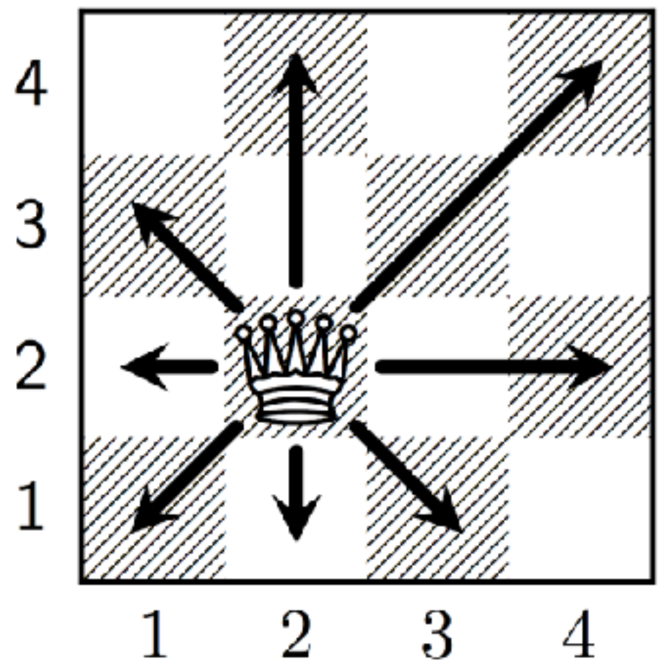
Attaque verticale : $a \neq c$, $b = d$





Attaque diagonale : $|a - c| = |b - d|$, $(a,b) \neq (c,d)$

4 reines en Python



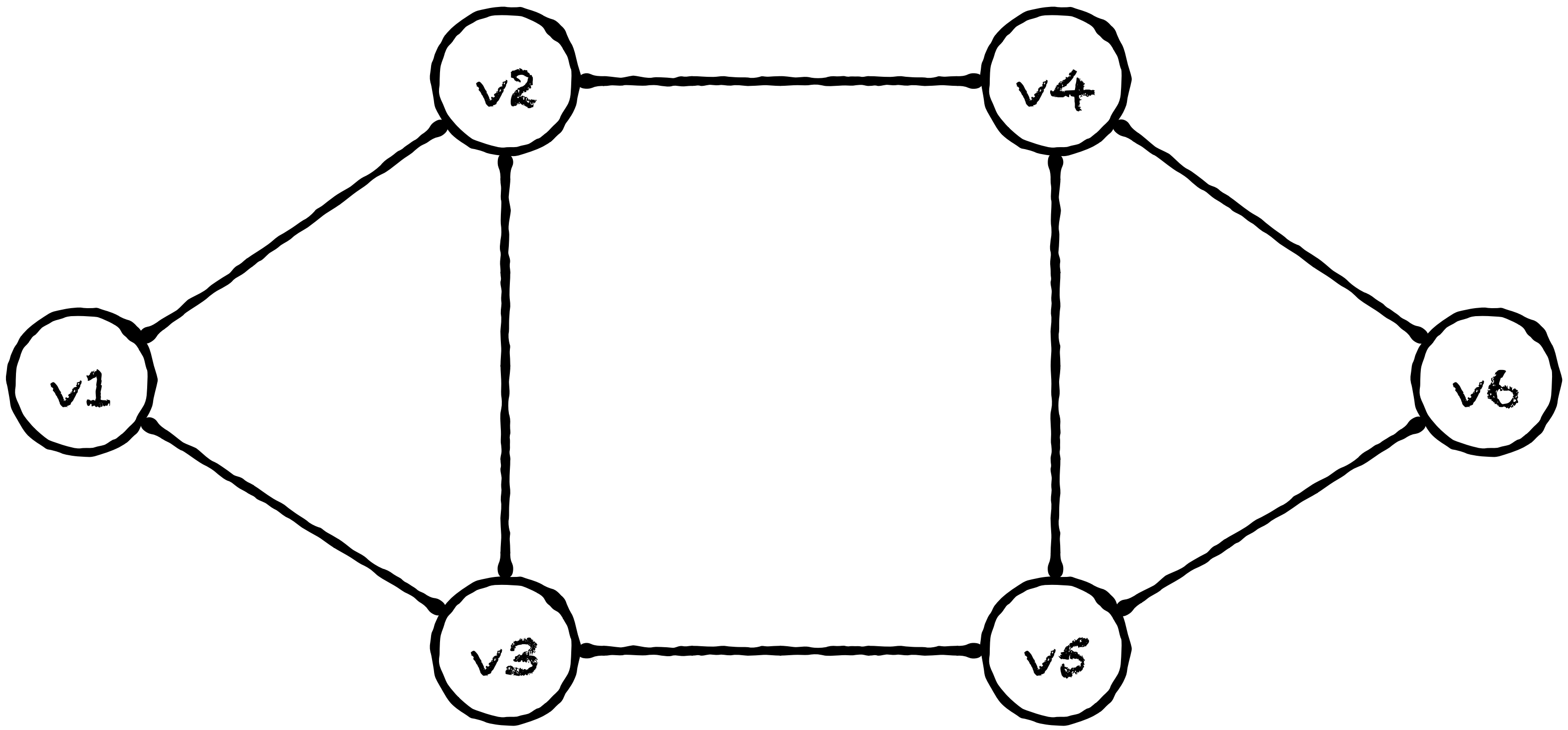
1. Il y a au moins une reine dans chaque colonne

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

Coloration de graphe

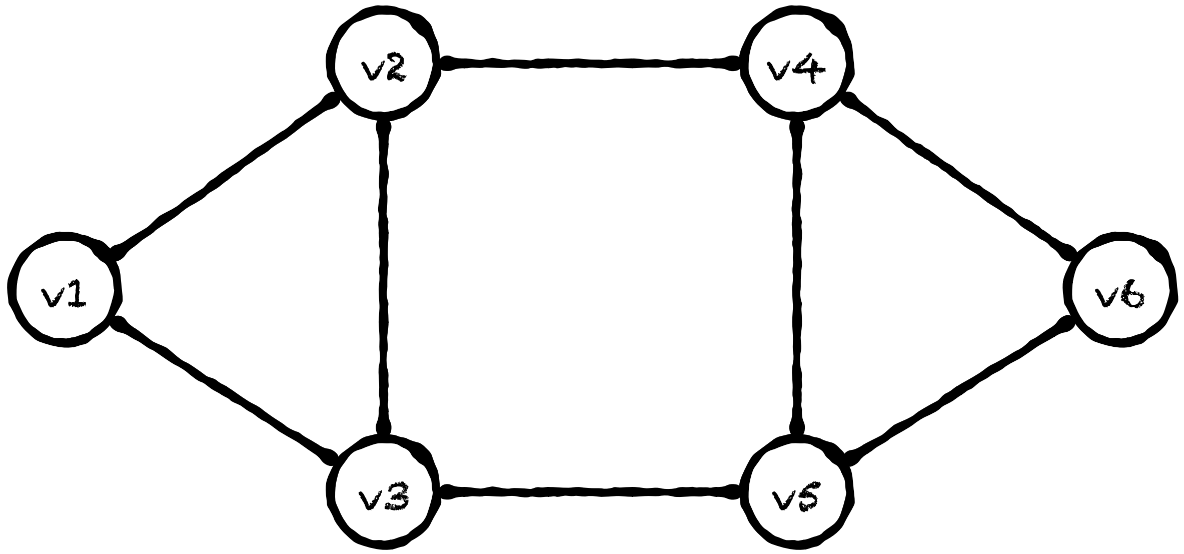


- Le graphe a-t-il une coloration avec 3 couleurs ?
- Couleurs : 1, 2, 3
- Introduire une variable x_{ic} pour toute $i = 1, \ldots, 6$ et $c = 1, 2, 3$
- La variable $x_{ic} = 1$ indique que le sommet v_i prend la couleur c

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

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

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

   $$(\neg x_{i1} \vee \neg x_{i2}),$$
   $$(\neg x_{i1} \vee \neg x_{i3}),$$
   $$(\neg x_{i2} \vee \neg x_{i3})$$

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

   $$(\neg x_{ic} \vee \neg x_{jc})$$

"Macros"

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

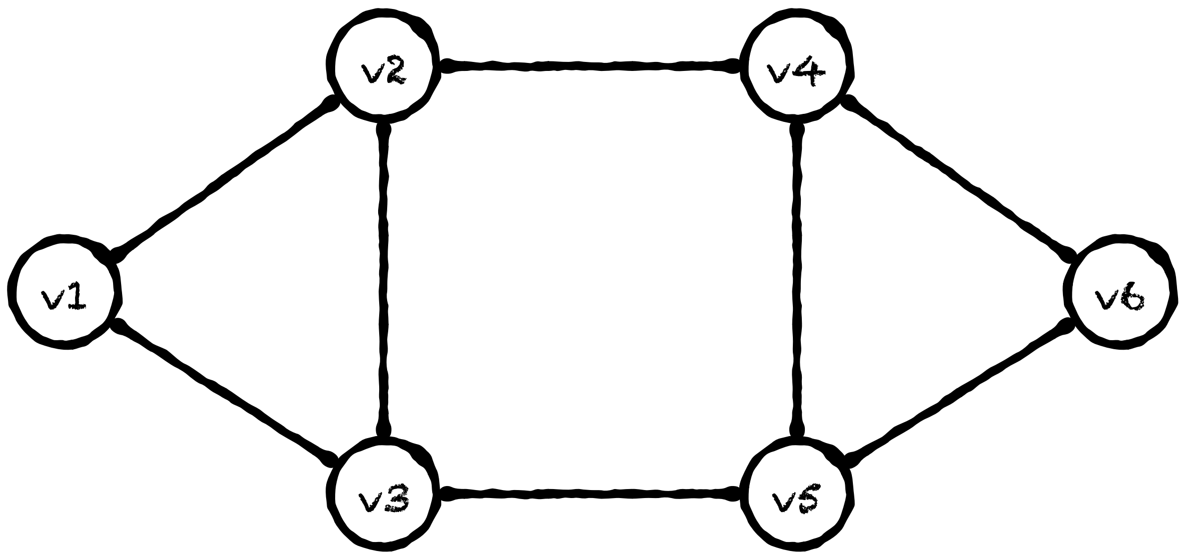
$$\text{AtLeastOne}(x_1, x_2, \ldots, x_n)$$
$$\Leftrightarrow$$
$$(x_1 \lor x_2 \lor \ldots \lor x_n)$$

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

$$\text{AtMostOne}(x_1, x_2, \ldots, x_n)$$
$$\Leftrightarrow$$
$$(\neg x_1 \lor \neg x_2), (\neg x_1 \lor \neg x_3), \ldots, (\neg x_{n-1} \lor \neg x_n) \quad \textit{(toute paire)}$$

Coloration de graphe

- x_{ic} = 1 si le sommet v_i prend la couleur c



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

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

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

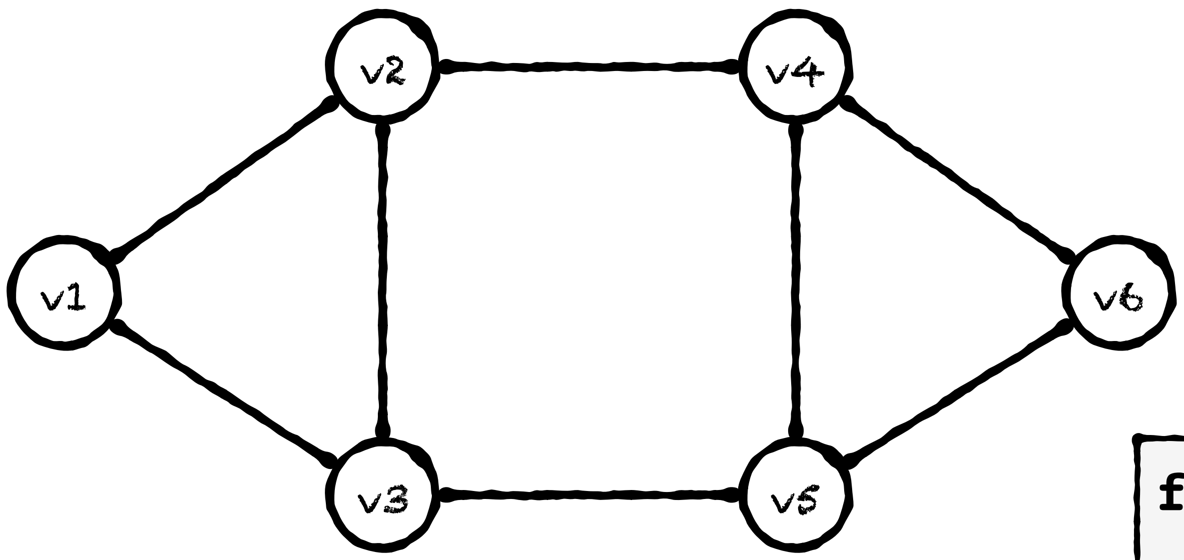
   AtMostOne(x_{i1}, x_{i2}, x_{i3})

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

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

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

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:
  lst = [var(i,0),var(i,1),var(i,2)]
  clauses = at_least_one(lst)
  cnf.extend(clauses)
  clauses = at_most_one(lst)
  cnf.extend(clauses)
```

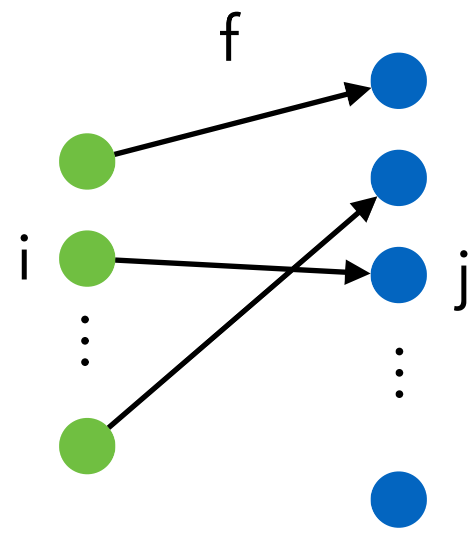
3. Les deux sommets d'une arête (v_i, v_j) ne prennent pas la même couleur c

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

Formaliser des applications

- On a souvent besoin de formaliser une application

$$f : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\}$$



  - affecter les colonnes aux reines
  - affecter les couleurs aux sommets
- Variables de décision *intention* : $x_{ij} = 1$ si $f(i) = j$
- Pour que f soit une application (et non pas seulement une relation binaire), on introduit des contraintes qui disent

"pour tout i, il y a exactement un j t.q. f(i) = j"

$$\text{AtMostOne}(x_{i1}, x_{i2}, \ldots, x_{im})$$

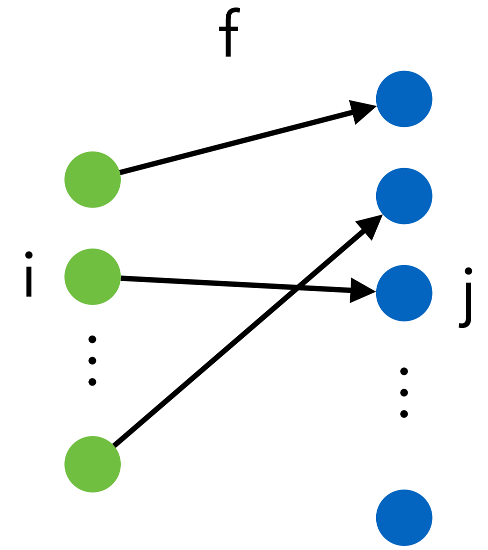
$$\text{AtLeastOne}(x_{i1}, x_{i2}, \ldots, x_{im})$$

Formaliser des injections

- Supposer que l'application $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$ est injective
  - pas deux reines dans une même colonne
- Pour imposer cette condition supplémentaire sur f, on introduit des contraintes qui disent

"pour tout j, il y a au plus un i t.q. f(i) = j"

$$\text{AtMostOne}(x_{1j}, x_{2j}, \ldots, x_{nj})$$



AtMost-k(...), k > 1

- Comment peut-on exprimer qu'au plus 2 d'entre les variables x, y, z, u, v sont vraies ?
- "Si je sélectionne 3 d'entre elles, au moins 1 est fausse"

$$\text{AtMost-2}(x, y, z, u, v)$$
$$\Leftrightarrow$$
$$(\neg x \vee \neg y \vee \neg z), (\neg x \vee \neg y \vee \neg u), (\neg x \vee \neg y \vee \neg v), \ldots, (\neg z \vee \neg u \vee \neg v)$$

- Combien de clauses ?

  sélection de 3 parmi 5 = $(5\times4\times3)/(3\times2\times1) = 10$

- Combien de clauses pour n variables ?

  sélection de 3 parmi n = $n(n-1)(n-2)/6 = O(n^3)$

AtMost-k($x_1, \ldots, x_n$), $k > 1$

- On peut utiliser moins de clauses en introduisant $n \times k$ variables auxiliaires y_{ij} pour $i = 1, \ldots, n$ et $j = 1, \ldots, k$
- L'idée est de poser $x_i \rightarrow y_{i1} \vee y_{i2} \vee \ldots \vee y_{ik}$
- Au plus k parmi les y_{ij} peuvent être vraies
  - au plus 1 dans chaque colonne

$$\begin{matrix} y_{11} & y_{12} & \ldots & y_{1j} & \ldots & y_{1k} \\ y_{21} & y_{22} & \ldots & y_{2j} & \ldots & y_{2k} \\ \ldots & \ldots & & \ldots & & \ldots \\ y_{i1} & y_{i2} & \ldots & y_{ij} & \ldots & y_{ik} \\ & & & & & \\ y_{n1} & y_{n2} & \ldots & y_{nj} & \ldots & y_{nk} \end{matrix}$$

AtMostOne($y_{1j}, y_{2j}, \ldots, y_{nj}$)

AtMost-k(x_1, …, x_n), k > 1

- $n \times k$ variables auxiliaires
- $k \times AMO(n) + n = O(k \times n^2)$ clauses
- $AMO(n)$ = le nombre de clauses de AtMostOne(x_1, …, x_n)

$$x_i \rightarrow y_{i1} \vee y_{i2} \vee \dots \vee y_{ik}$$

$$\begin{array}{cccccc}
y_{11} & y_{12} & \dots & y_{1j} & \dots & y_{1k} \\
y_{21} & y_{22} & \dots & y_{2j} & \dots & y_{2k} \\
\dots & \dots & & \dots & & \dots \\
y_{i1} & y_{i2} & \dots & y_{ij} & \dots & y_{ik} \\
 & & & & & \\
y_{n1} & y_{n2} & \dots & y_{nj} & \dots & y_{nk}
\end{array}$$

AtMostOne(y_{1j}, y_{2j}, …, y_{nj})

AtLeast-k(x_1, …, x_n), k > 1

- On introduit n×k variables auxiliaires y_{ij} pour i = 1, …, n, j = 1, …, k et on pose $y_{i1} \vee y_{i2} \vee \ldots \vee y_{ik} \rightarrow x_i$
- Au moins k rangs doivent être non-zéro
  - **au moins** 1 y_{ij} vraie dans chaque colonne
  - au plus 1 y_{ij} vraie dans chaque rang

$$
\begin{array}{r|cccccc}
 & y_{11} & y_{12} & \ldots & y_{1j} & \ldots & y_{1k} \\
 & y_{21} & y_{22} & \ldots & y_{2j} & \ldots & y_{2k} \\
 & \ldots & \ldots & & \ldots & & \ldots \\
\text{AtMostOne}(y_{i1}, y_{i2}, \ldots, y_{ik}) & y_{i1} & y_{i2} & \ldots & y_{ij} & \ldots & y_{ik} \\
 & & & & & & \\
 & y_{n1} & y_{n2} & \ldots & y_{nj} & \ldots & y_{nk}
\end{array}
$$

AtLeastOne(y_{1j}, y_{2j}, …, y_{nj})

AtLeast-k(x_1, …, x_n), k > 1

- n×k variables auxiliaires
- k + n×AMO(k) + n×k = O(n×k²) clauses
- 3n+2 clauses pour k = 2

$$y_{i1} \lor y_{i2} \lor \dots \lor y_{ik} \rightarrow x_i$$

$$\begin{array}{cccccc}
y_{11} & y_{12} & \dots & y_{1j} & \dots & y_{1k} \\
y_{21} & y_{22} & \dots & y_{2j} & \dots & y_{2k} \\
\dots & \dots & & \dots & & \dots \\
y_{i1} & y_{i2} & \dots & y_{ij} & \dots & y_{ik} \\
 & & & & & \\
y_{n1} & y_{n2} & \dots & y_{nj} & \dots & y_{nk}
\end{array}$$

AtMostOne(y_{i1}, y_{i2}, …, y_{ik})

AtLeastOne(y_{1j}, y_{2j}, …, y_{nj})

AtMostOne(n)

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

- Cette construction utilise $n(n-1)/2 = O(n^2)$ clauses
- Il y a d'autres constructions de AtMostOne qui utilisent moins de clauses en introduisant des variables auxiliaires
  - Exercice au td : $O(n)$ clauses, $O(n)$ variables auxiliares
- $\Rightarrow$ $O(n\times k)$ clauses et $O(n\times k)$ variables auxiliares pour

  AtMost-k(x_1, …, x_n) et AtLeast-k(x_1, …, x_n)