ISC9000 Connaissance, Raisonnement et Prise de Décision

Résolutions de contraintes et inférence de connaissances implicites

Roger Villemaire

Département d'informatique
UQAM

23 avril 2026

UQÀM
Université du Québec à Montréal

Plan

Plan

Intelligence Artificielle

- L'intelligence artificielle peut être décomposée en un ensemble de *tâches*, qui réalisées conjointement permettrait la conception d'agents artificiels intelligents [1].
- La même idée est mise de l'avant dans les architectures cognitives, à la différence que
  - en IA on pense qu'il est encore prématuré de faire l'intégration de ces différentes approches, car il reste encore beaucoup de progrès à faire et qu'il est donc nécessaire de se limiter à une seule tâche.
- Jusqu'à maitenant nous avons vu : l'apprentissage à l'aide de données, le raisonnement, et aujoud'hui la résolution de contraintes.

1. Stuart J. Russell, Peter Norvig : Artificial Intelligence : A Modern Approach, Prentice Hall, 2010.

Résolution de contraintes

- Il s'agit de trouver des valeurs à des variables pour satisfaire un ensemble de contraintes.
- Mathématiquement, il s'agit de résoudre un système, comme pour les équations !
- C'est donc une approche générale, applicable à toutes sortes de problèmes.
  - En IA, on veut des méthodes algorithmiques rapides pour pouvoir s'en servir, par ex. en planification, robotique, systèmes tutoriels intelligents, jeux, regroupement de données, etc.

Exemples : Disposition des invités

- On veut disposer des invités autour d'une table circulaire. Il y a bien le même nombre de places que de convives, malheureusement certaines personnes ne peuvent pas être assises côte à côte :
  - Béatrice, Charles et François ne peuvent pas être assis à côté d'Albert depuis qu'il a obtenu cette promotion tant convoitée !
  - Béatrice et Charles (ainsi que François et Élise) sont des ex. qu'il vaudrait mieux ne pas asseoir côte à côte !
  - Pour une raison pas très claire, Denis ne peut être placé au côté ni d'Élise, ni de Charles !
  - Finalement, François veut absolument être assis à côté de Béatrice pour lui parler de son tout nouveau projet et ainsi profiter de ses conseils !

Plan

Problème de satisfaction de contraintes

- La situation est représentée sous forme
  - d'un ensemble de *variables* $X, Y, Z, \ldots$ qui peuvent prendre des valeurs dans leurs domaines
    $X \in Dom_X, Y \in Dom_Y, Z \in Dom_Z, \ldots$
  - d'un ensemble $\mathcal{C}$ de *contraintes* que les valeurs choisies des variables doivent satisfaire.
- Il s'agit alors de trouver une solution, c.-à-d. des valeurs précises aux variables pour que toutes les contraintes soient satisfaites.

Inférence et Recherche

- Une méthode centrale pour trouver une solution pour un ensemble de contraintes est d'appliquer les principes suivants :
  - l'*inférence* : déduire des restrictions sur les domaines des variables, tant que c'est possible,
  - la *recherche* : puis essayer les valeurs qu'il reste dans le domaine d'une variable.

Motivation

- Intuitivement la méthode d'inférence et recherche est similaire à celle qu'on utilise en tant qu'être humain,
  - par exemple pour charger sa voiture et partir en vacance,
  - ou encore pour résoudre des jeux logiques (par ex. Sudoku).
- L'inférence et la recherche est la méthode adéquate et complète (sound and complete) dominante :
  - (adéquate) si elle trouve une solution c'en est une,
  - (complète) s'il n'y a pas de solution, la méthode va le déterminer (s'il y a suffisamment de temps !).

Disposition des invités

- On a 6 invités, donc 6 places qu'on numérote $0, 1, 2, 3, 4, 5$
- À chaque personne on doit assigner une place :
  - on a donc six variables A, B, C, D, E, F représentant la place qui sera assignée à Albert, Béatrice, Charles, Denis, Élise et François, respectivement
  - toutes nos variables ont donc *au départ* le même domaine, soit $\{0, 1, 2, 3, 4, 5\}$
- Pour ce qui est des contraintes, elles sont de trois types :
  - X et Y ne doivent pas être assis côte à côte,
  - X et Y doivent être assis côte à côte,
  - $X \neq Y$, pour X et Y des variables différentes, car deux personnes doivent être assises sur des chaises différentes.

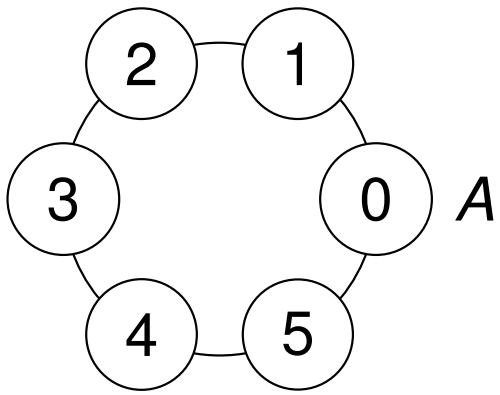
Considérations algorithmiques

- Il est souvent préférable de tenter de réduire le nombre de variables au maximum car :
  - il pourra alors être plus facile de trouver des méthodes d'inférence, réduisant le nombre de possibilités pour les valeurs des variables,
  - il y aura moins de variables, donc potentiellement moins de possibilités à tenter dans la phase de recherche.
- La façon de représenter un problème peut avoir un impact considérable sur la performance. En pratique il est souhaitable de tenter plusieurs représentations et de vérifier expérimentalement la performance à l'aide des meilleurs *solveurs* de contraintes.

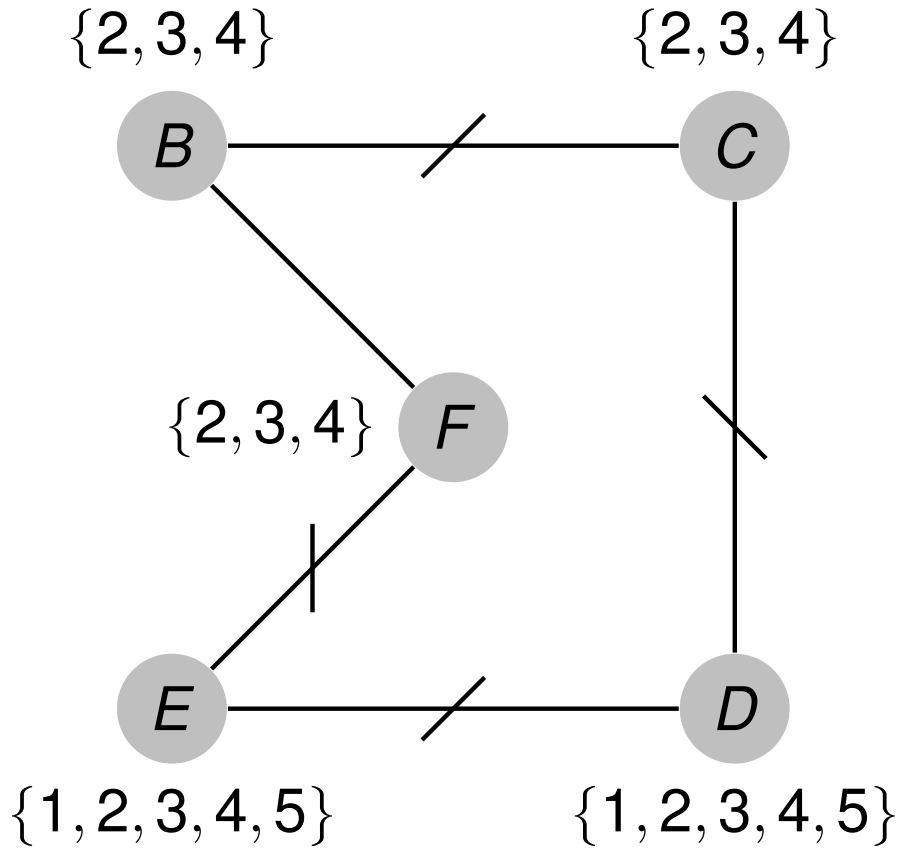
Disposition des invités : Simplification

- Comme la table est circulaire, seule la position relative des invités est importante :
  - On peut donc arbitrairement considérer que $A = 0$, sans changer le fait que le problème ait une solution ou non
  - Il nous reste donc :
    - 5 places à assigner : $1, 2, 3, 4, 5$
    - et les contraintes deviennent :
    - $Dom_B = \{2, 3, 4\}$, $Dom_C = \{2, 3, 4\}$, $Dom_F = \{2, 3, 4\}$,
    - $B + C$, $F + E$, $D + E$, $D + C$ (ne sont pas côte à côte),
    - et $F - B$ (sont côte à côte),
    - de plus toutes les variables doivent prendre des valeurs distinctes dans {1,2,3,4,5}.

Représentation Graphique



La table



Le réseau de contraintes

Plan

Logique propositionnelle

- Des variables de domaine $\{0, 1\}$ ($0 = \textit{faux}$ et $1 = \textit{vrai}$).
- Les formules sont construites, à partir des variables, à l'aide de la négation $\neg$, la conjonction ("et logique") $\wedge$ et de la disjonction $\vee$.
- Exemples :
  - $\neg q \vee r$
  - $p \wedge (\neg q \vee r)$
- Une formule propositionnelle est donc une contrainte sur ses variables : on cherche des valeurs des variables qui satisfont la formule (la rendent vraie).

Problème SAT

- Trouver une solution (des valeurs pour ses variables qui la rende vraie) pour une formule propositionnelle en *forme normale conjonctive* (FNC) :
  - la formule est donc une conjonction (ET) de *clauses*
  - une *clause* est une disjonction de variables et de négations de variables.
- Ceci n'est pas une restriction puisque toute formule propositionnelle peut être transformée en FNC. De plus, la FNC simplifie la gestion algorithmique.

Exemple

- $(p \vee q \vee \neg r) \wedge (\neg p \vee r) \wedge (p \vee \neg q \vee r) \wedge q$ est une FNC
- Une telle formule est normalement représentée de façon suivante

$$\begin{array}{lll} p & q & \bar{r} \\ \bar{p} & r & \\ p & \bar{q} & r \\ q & & \end{array}$$

- Il s'agit donc de chercher à rendre au moins un *littéral* (une variable ou la négation d'une variable) de chaque ligne vrai.
- Évidemment une variable et sa négation doivent prendre des valeurs contraires (une est 0 et l'autre 1).

Discussion

- Le problème SAT est d'une certaine façon le problème de résolution de contrainte le plus simple puisque chaque variable ne peut prendre que deux valeurs (vrai ou faux).
- Néanmoins, tout problème fini (nombre fini de variables et de valeur) peut s'y réduire.
- De plus, sa simplicité aide au développement d'algorithme.
- C'est aussi un problème bien compris au niveau théorique.

Complexité algorithmique : P

- Un problème est dans la classe P (polynomial) s'il existe un algorithme pour le résoudre dans un temps de calcul (nombre d'étapes) borné par un polynôme en fonction de la taille de l'entrée, par exemple
  - n^2 : $10^2 = 100$, $(2n)^2 = 4n^2$, doubler la taille, temps de calcul $x4$,
  - n^3 : $10^3 = 1000$, $(2n)^3 = 8n^3$, doubler la taille, temps de calcul $x8$,
  - n^k : $10^k = 1000\ldots000$, $(2n)^k = 2^k n^k$, doubler la taille, temps de calcul $x2^k$ (une constante).
- Ceci est très différent de la croissance exponentielle :
  - 10^n : $10^{10} = 10\ 000\ 000\ 000$, $(10)^{2n} = (10)^n \cdot (10)^n$, doubler la taille, temps de calcul augmente d'un facteur égal au temps de calcul original !
- En informatique théorique, polynomial est considéré comme "faisable" ou "rapide".

Complexité algorithmique : *NP*

- Un problème *Prob* est dans la classe *NP* (non-déterministe polynomial) s'il existe un algorithme polynomial *ProbNP* qui prend deux entrées : l'entrée *e* de *Prob* (instance) et une valeur *o* (oracle) et telle que *Prob*(e) si et seulement s'il existe un valeur *o* pour l'oracle et *ProbNP*(e, o) donne la même réponse que *Prob*(e).
  - idée : lorsque j'ai l'oracle, je peux obtenir la solution rapidement
  - mais on n'a pas nécessairement de méthode rapide pour trouver l'oracle !
  - Donc, vérification rapide, mais recherche possiblement difficile.

SAT, *P*, et *NP*

- SAT (existe-t-il une solution), est dans *NP* puisqu'il suffit de prendre la solution comme oracle.
- Je peux vérifier rapidement si c'est une solution.
- Est-ce que je peux trouver rapidement une solution ? Le consensus est non !
- Si on peut trouver un solution à SAT rapidement (en temps polynomial) on aurait que SAT est dans le classe *P*.

$P = NP$

- On se demande donc si SAT est dans P !
  - C'est le plus grand problème ouvert de l'informatique théorique.
  - La plupart des théoriciens pensent que non.
  - Il y a un millions de dollars à gagner pour résoudre cette question (prix du millénaire) !
- En fait, P est inclus dans NP et on sait que s'il y a un problème dans NP qui n'est pas dans P alors SAT est un tel problème (on dit qu'il est NP-complet)
- $P = NP$ demande à trouver une méthode directe, beaucoup plus rapide que la recherche.
- Mais, même si on ne pense pas être en mesure de résoudre $P = NP$ en produisant un algorithme polynomial pour SAT, on peut chercher des algorithmes plus rapides !

Plan

Inférence : Propagation d'unités

- Lors de la recherche d'une solution pour le problème SAT, si tous les littéraux d'une clause, sauf un, sont faux, ce dernier littéral doit nécessairement être vrai.
- La procédure qui met à vrai de tels littéraux, la *propagation d'unités*, est donc une méthode d'inférence pour le problème SAT.

Algorithme DPLL

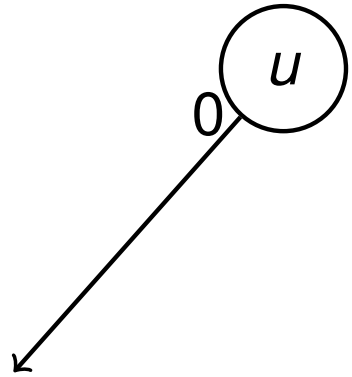
- L'algorithme de Davis, Putnam, Logemann et Loveland procède comme suit :
  - faire la propagation d'unités, si possible
  - si toute les clauses ont au moins un littéral vrai, on a trouvé une solution et la procédure se termine, sinon s'il n'y a pas de *conflit* (une clause dont tous les littéraux sont faux), on choisit une variable et une valeur pour cette variable et on retourne à l'étape précédente
  - s'il y a un conflit, on retourne à la dernière variable choisie pour laquelle il reste une valeur à tenter et on choisit cette nouvelle valeur pour cette variable et on revient à la première étape, sinon on a essayé toutes les possibilités et il n'y a pas de solution.
- C'est un algorithme d'inférence et de recherche pour le problème SAT !

Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$

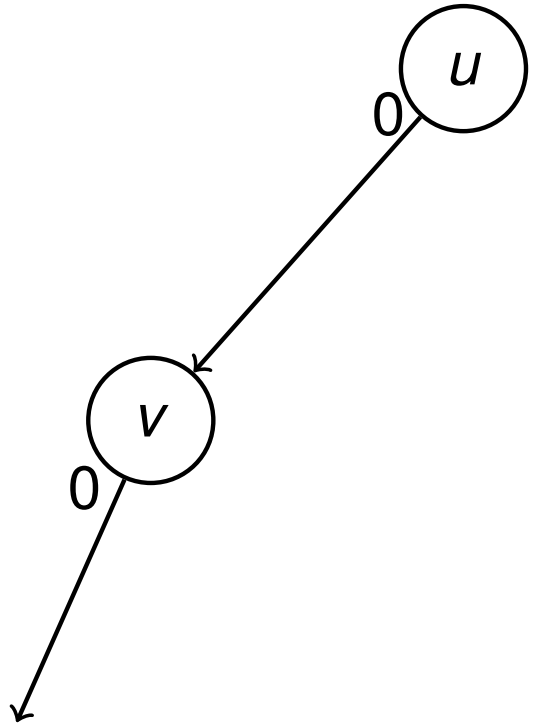
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



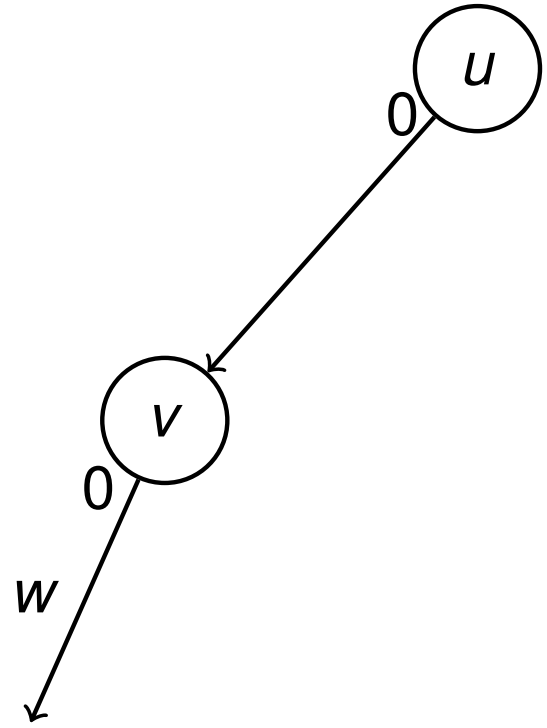
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



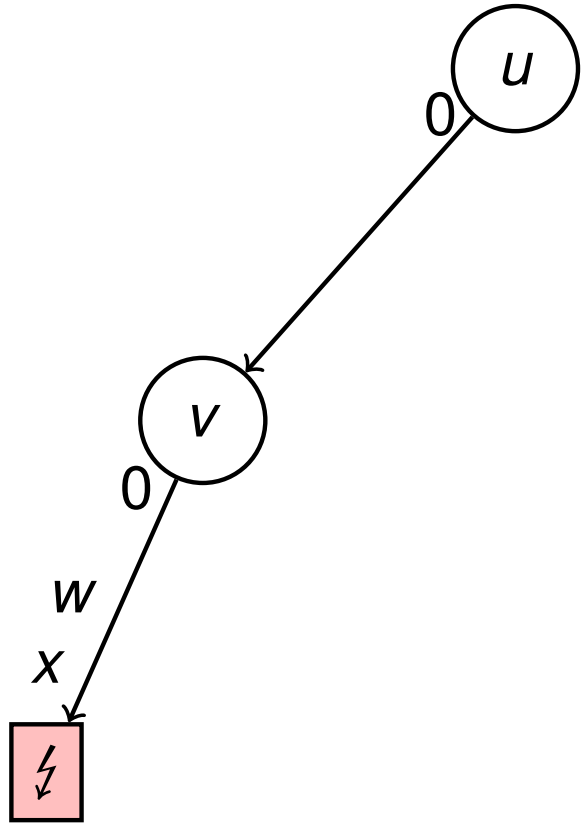
Exemple DPLL

u	v	w
u	w	$\bar{x}$
u	v	x
v	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	w
u	w	$\bar{x}$
$\bar{v}$	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	x
$\bar{u}$	v	w
$\bar{u}$	w	$\bar{x}$
$\bar{u}$	v	x
v	$\bar{w}$	$\bar{x}$



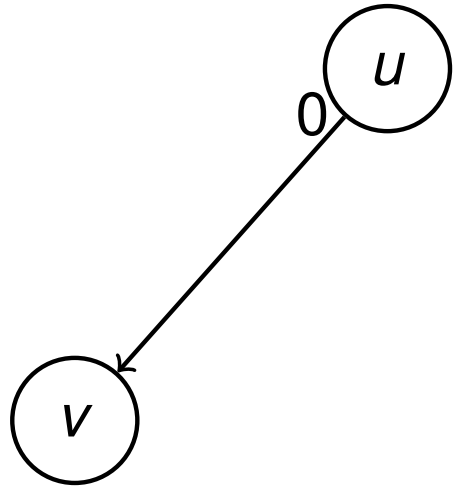
Exemple DPLL

u	v	w
u	w	$\bar{x}$
u	v	x
v	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	w
u	w	$\bar{x}$
$\bar{v}$	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	x
$\bar{u}$	v	w
$\bar{u}$	w	$\bar{x}$
$\bar{u}$	v	x
v	$\bar{w}$	$\bar{x}$



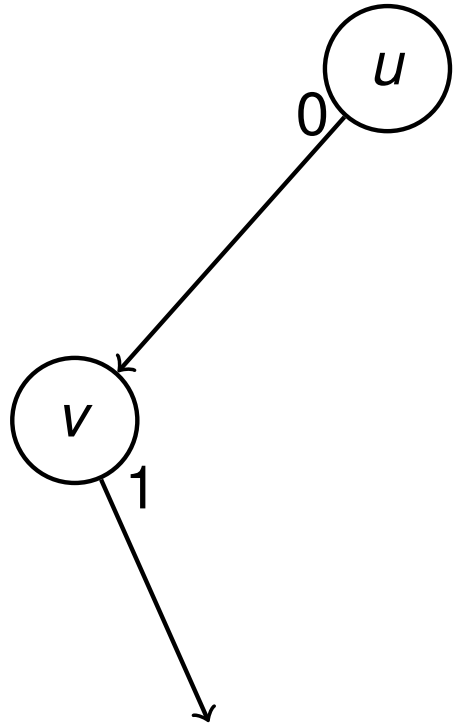
Exemple DPLL

u	v	w
u	w	$\bar{x}$
u	v	x
v	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	w
u	w	$\bar{x}$
$\bar{v}$	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	x
$\bar{u}$	v	w
$\bar{u}$	w	$\bar{x}$
$\bar{u}$	v	x
v	$\bar{w}$	$\bar{x}$



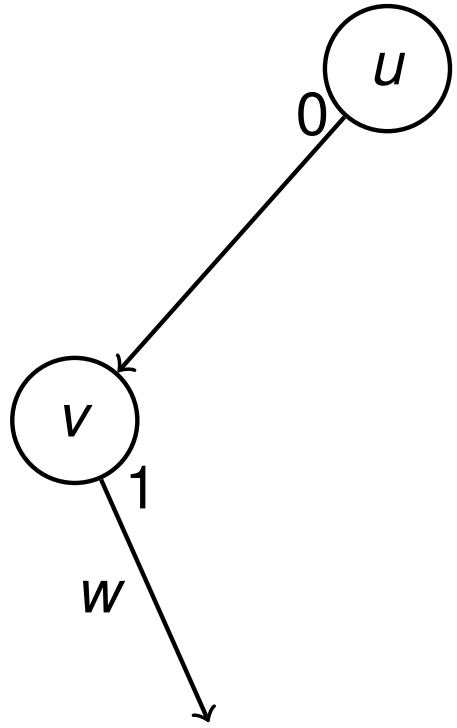
Exemple DPLL

u	v	w
u	w	$\bar{x}$
u	v	x
v	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	w
u	w	$\bar{x}$
$\bar{v}$	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	x
$\bar{u}$	v	w
$\bar{u}$	w	$\bar{x}$
$\bar{u}$	v	x
v	$\bar{w}$	$\bar{x}$



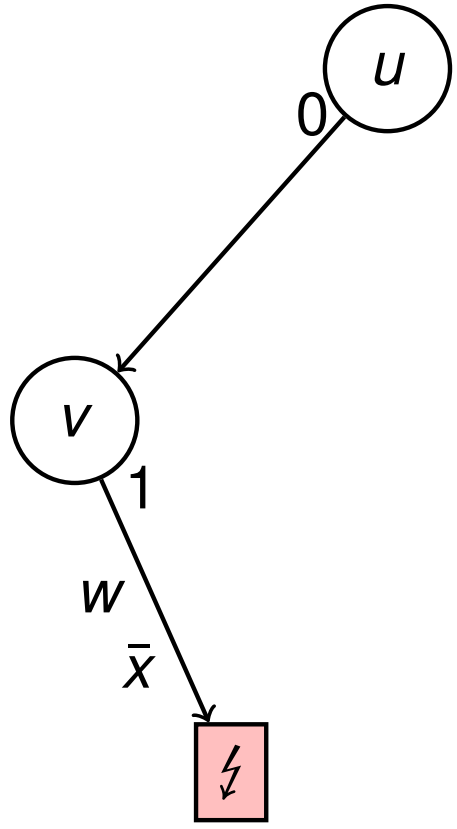
Exemple DPLL

u	v	w
u	w	$\bar{x}$
u	v	x
v	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	w
u	w	$\bar{x}$
$\bar{v}$	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	x
$\bar{u}$	v	w
$\bar{u}$	w	$\bar{x}$
$\bar{u}$	v	x
v	$\bar{w}$	$\bar{x}$



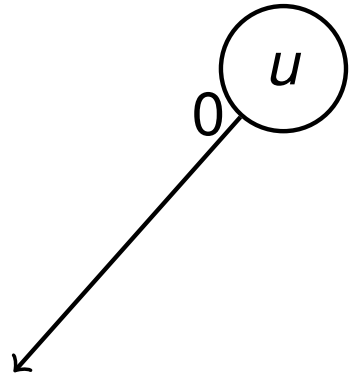
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



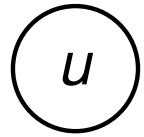
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



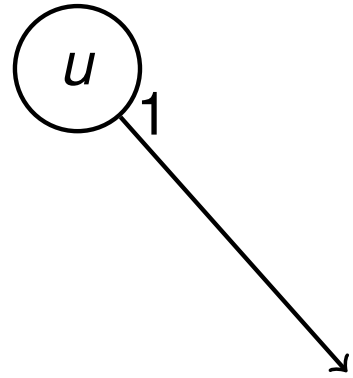
Exemple DPLL

u v w
u w $\bar{x}$
u v x
v $\bar{w}$ $\bar{x}$
u $\bar{v}$ w
u w $\bar{x}$
$\bar{v}$ $\bar{w}$ $\bar{x}$
u $\bar{v}$ x
$\bar{u}$ v w
$\bar{u}$ w $\bar{x}$
$\bar{u}$ v x
v $\bar{w}$ $\bar{x}$



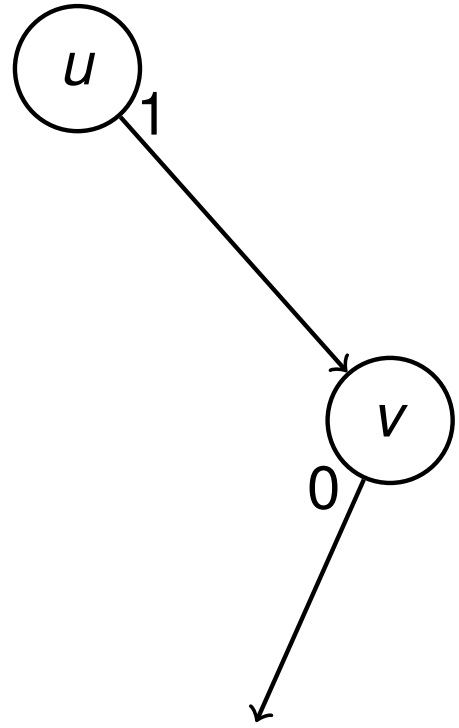
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



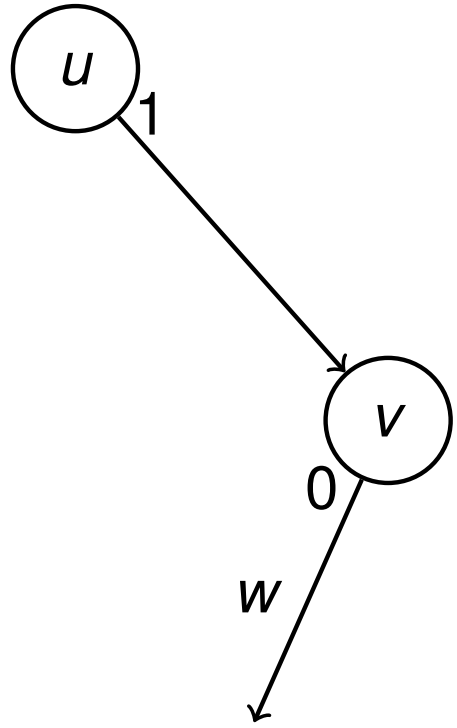
Exemple DPLL

u	v	w
u	w	$\bar{x}$
u	v	x
v	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	w
u	w	$\bar{x}$
$\bar{v}$	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	x
$\bar{u}$	v	w
$\bar{u}$	w	$\bar{x}$
$\bar{u}$	v	x
v	$\bar{w}$	$\bar{x}$



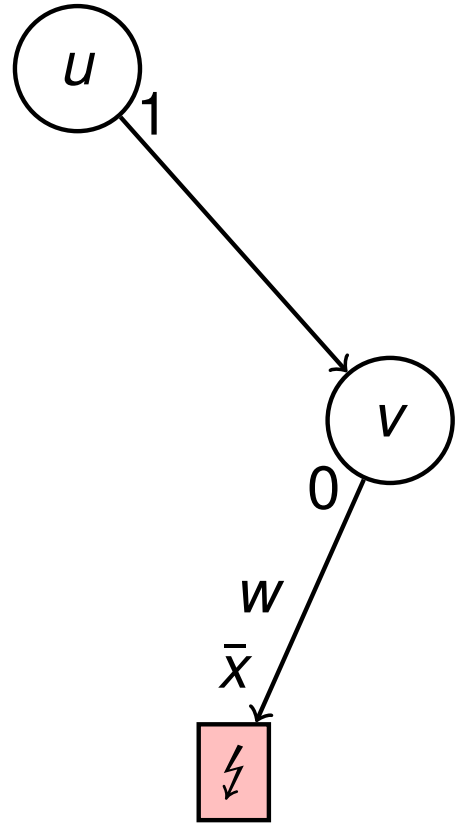
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



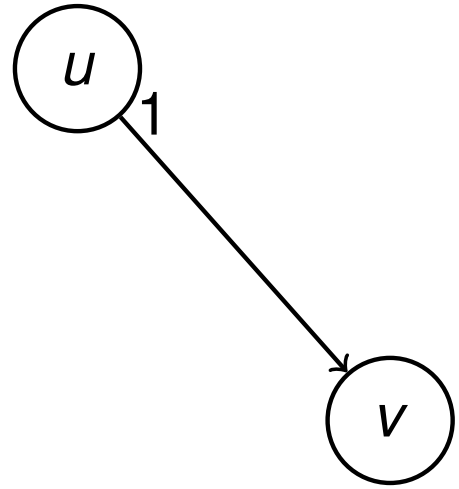
Exemple DPLL

u	v	w
u	w	$\bar{x}$
u	v	x
v	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	w
u	w	$\bar{x}$
$\bar{v}$	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	x
$\bar{u}$	v	w
$\bar{u}$	w	$\bar{x}$
$\bar{u}$	v	x
v	$\bar{w}$	$\bar{x}$



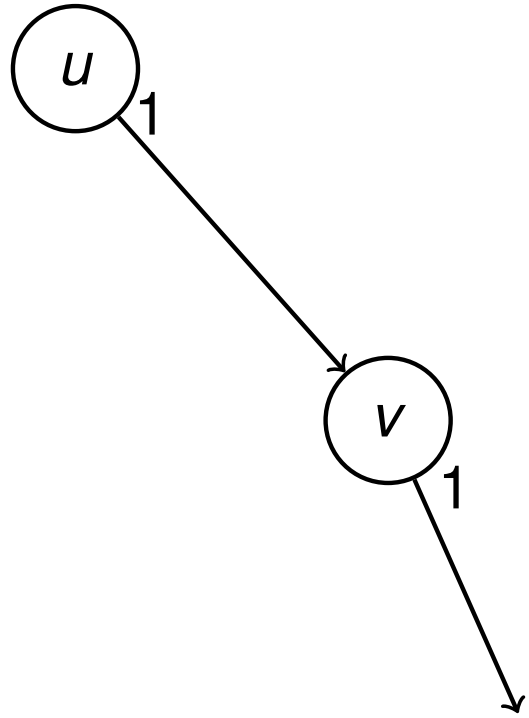
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



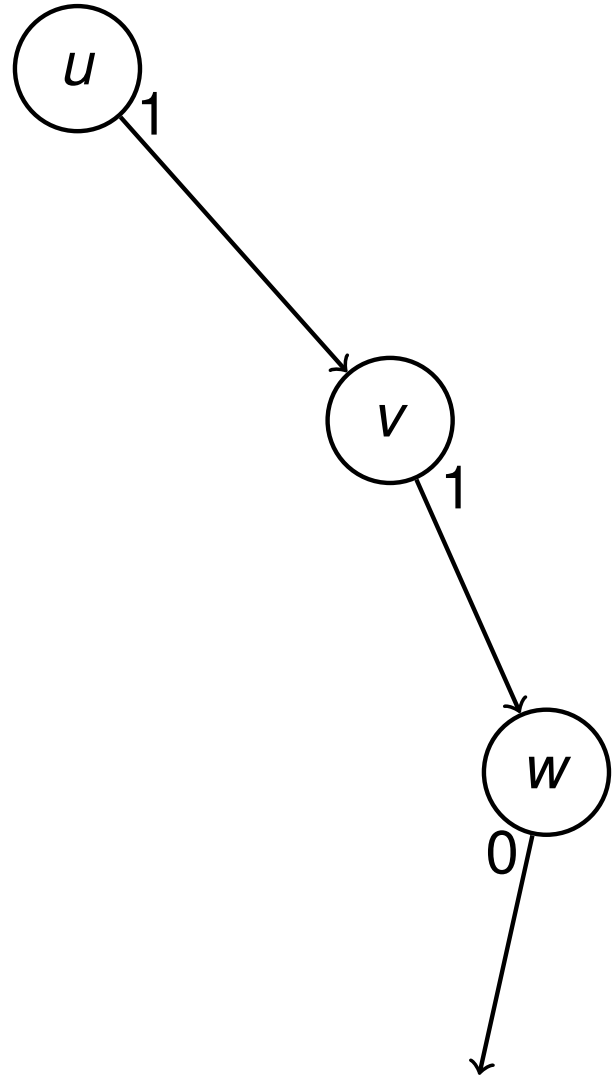
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



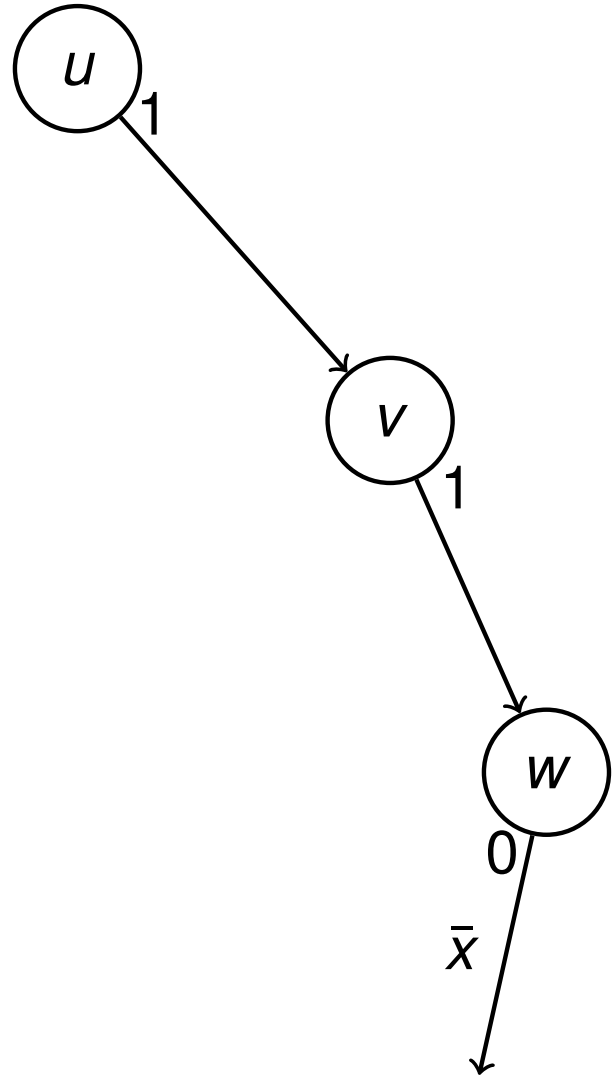
Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



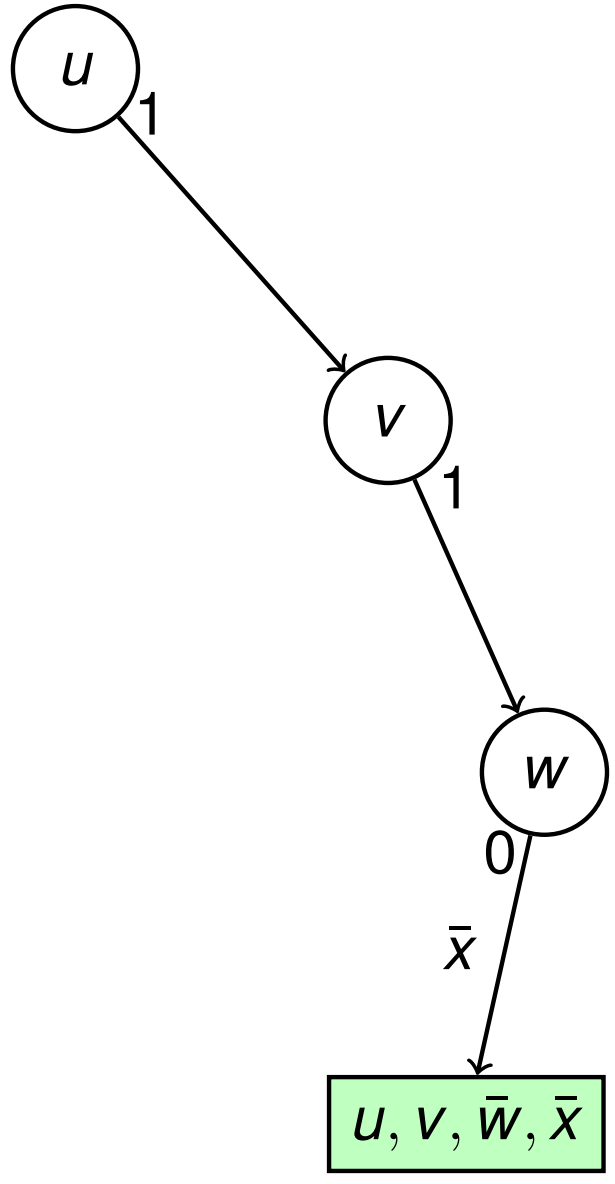
Exemple DPLL

u	v	w
u	w	$\bar{x}$
u	v	x
v	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	w
u	w	$\bar{x}$
$\bar{v}$	$\bar{w}$	$\bar{x}$
u	$\bar{v}$	x
$\bar{u}$	v	w
$\bar{u}$	w	$\bar{x}$
$\bar{u}$	v	x
v	$\bar{w}$	$\bar{x}$



Exemple DPLL

$u \quad v \quad w$
$u \quad w \quad \bar{x}$
$u \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad w$
$u \quad w \quad \bar{x}$
$\bar{v} \quad \bar{w} \quad \bar{x}$
$u \quad \bar{v} \quad x$
$\bar{u} \quad v \quad w$
$\bar{u} \quad w \quad \bar{x}$
$\bar{u} \quad v \quad x$
$v \quad \bar{w} \quad \bar{x}$



Remarques

- La propagation d'unités de l'algorithme DPLL permet d'éviter de devoir tenter toutes les valeurs possibles pour toutes les variables, ce qui serait considérablement plus long.
- Les solveurs SAT modernes utilisent néanmoins des méthodes plus sophistiquées, comme par exemple l'algorithme CDCL (Conflit Directed Clauses Learning), qui permet de déterminer une *raison* d'un conflit et de faire un saut-arrière, en général important, réduisant d'autant le nombre d'assignations explorées.
- En pratique de tels *solveurs* sont très efficaces et largement utilisés car tout problème fini, et certains infinis, peut être réduit à SAT.
- Ceci même si tous les algorithmes connus sont de complexités exponentielles (problème $P = NP$).

Représentation

- N'importe quel problème fini peut s'encoder en SAT.
- Il y a beaucoup de méthodes, aux performances variées, mais la plus simple consiste à :
- introduire, pour chaque variable v du problème et chacune des valeurs a de son domaine dom_v
  - une variables propositionnelle $p_{v,a}$ qui signifiera $v = a$,
- ajouter des clauses pour indiquer que v doit prendre une valeur de son domaine
  - $p_{v,a_1} \vee \cdots \vee p_{v,a_n}$ pour $dom_v = \{a_1, \ldots, a_n\}$
- mais une seule
  - $\neg p_{v,a} \vee \neg p_{v,b}$, pour tous $a \neq b$ de dom_v,
- ainsi que des clauses pour encoder les contraintes,
  - $\neg p_{v_1,a_1} \vee \cdots \vee \neg p_{v_n,a_n}$ si $v_1 = a_1, \ldots, v_n = a_n$ est interdite.

Plan

Optimisation

- Souvent on ne veut pas, ou même ne peut pas, satisfaire toutes les contraintes,
- on va donc plutôt viser à maximiser le nombre de contraintes satisfaites.
- Il s'agit alors d'un problème d'*optimisation* où on a
  - des contraintes *fortes* (hard) qui doivent être absolument vérifiées,
  - des contraintes *faibles* (soft) qui ont des *poids*, et
  - on cherche à maximiser la somme des poids des contraintes faibles satisfaites.

Exemples : Disposition des invités

- Dans le problème de la disposition des invités autour d'une table,
  - la contrainte que deux invités ne peuvent pas être assis sur la même chaise sera évidemment forte,
  - pour les autres, elles peuvent être soit fortes, soit faibles avec une pondération qui permettra d'en privilégier certaines.

Problème Max-SAT

- Trouver une solution optimale pour une formule propositionnelle en *forme normale conjonctive* (FNC) :
  - dont certaines clauses sont des contraintes fortes,
  - et les autres des contraintes faibles pondérées (en général) par des entiers.
- Comme dans le cas de SAT, tout problème d'optimisation de contraintes fini peut être traduit en Max-SAT.
- Il s'agit donc, comme pour SAT, d'une méthode générale qui a donné lieu à des développements algorithmiques importants.

Algorithmes Max-SAT

- La plupart des algorithmes Max-SAT les plus performants sont basés sur les solveurs SAT puisque ceux-ci peuvent, lorsqu'il n'a pas de solution, retourner
  - un unsat-core qui est un ensemble de clauses qui n'ont pas de solution.
- Comme cet ensemble de clauses est en général petit, et qu'il faut en violer au moins une pour trouver une solution, une méthode pour max-SAT est donc d'ajouter une contrainte à cet effet et de réitérer.
- Il est aussi possible de d'abord chercher à satisfaire toutes les contraintes fortes et d'ajouter des contraintes de cardinalités pour augmenter le nombre de contraintes faibles satisfaites (plutôt la somme des pondérations).

Plan

Science expérimentale

- Dans tous les cas, tant pour SAT que pour max-SAT, les avancées algorithmiques ont été largement stimulées par les compétitions ayant lieu lors des conférences scientifiques.
- Il s'agit donc de développer des algorithmes qui donnent de bonnes performances, sur des intances représentatives, et non de solutionner $P = NP$!
- Malgré le fait que tous les algorithmes sont en temps exponentiel, les avancées des dernières décennies on permis de résoudre, en pratique, de nombreux problèmes pour lesquels on n'avait pas de méthodes aussi efficaces.
- Il s'agit donc d'avancées algorithmiques pratiques, évaluées expérimentalement.