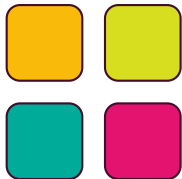


OPC – Partie 3

Modélisation et Satisfiabilité

Gilles Audemard

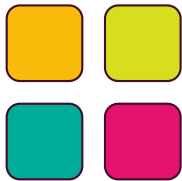
audemard@cril.fr



Introduction

Introduction

- Dernière partie sur le cours d'OPC
- Centrée sur la modélisation vue du côté du problème SAT
- Un petit plan
 - ▶ Le problème SAT
 - ▶ Les démonstrateurs de type CDCL
 - ▶ La modélisation de CSP vers SAT
 - ▶ D'autres types de modélisation
 - ▶ Avantages et inconvénients des diverses approches
 - ▶ Extensions au problème SAT



SAT

Définitions : le problème SAT

$$\begin{array}{l} \overline{x_1} \vee \overline{x_2} \vee x_3 \\ \wedge \qquad \qquad \overline{x_3} \\ \wedge x_1 \vee x_2 \\ \wedge \qquad x_2 \vee x_3 \end{array}$$

- Variables : $x_1 \dots x_3$
- Littéraux : $x_1, \overline{x_1}$
- Clauses : $\overline{x_1} \vee \overline{x_2} \vee x_3$
- Formule CNF
- Problème SAT : existe-il une interprétation des variables qui satisfait la formule ?

Définitions : le problème SAT

$$\begin{array}{l} \overline{x_1} \vee \overline{x_2} \vee x_3 \\ \wedge \qquad \qquad \overline{x_3} \\ \wedge x_1 \vee x_2 \\ \wedge \qquad x_2 \vee x_3 \end{array}$$

| | | |
|-------|-------|-------|
| x_1 | x_2 | x_3 |
| F | F | F |

- Variables : $x_1 \dots x_3$
- Littéraux : $x_1, \overline{x_1}$
- Clauses : $\overline{x_1} \vee \overline{x_2} \vee x_3$
- Formule CNF
- Problème SAT : existe-il une interprétation des variables qui satisfait la formule ?

Définitions : le problème SAT

$$\begin{array}{l} \overline{x_1} \vee \overline{x_2} \vee x_3 \\ \wedge \qquad \qquad \overline{x_3} \\ \wedge x_1 \vee x_2 \\ \wedge \qquad x_2 \vee x_3 \end{array}$$

| | | |
|-------|-------|-------|
| x_1 | x_2 | x_3 |
| F | F | F |

- Variables : $x_1 \dots x_3$
- Littéraux : $x_1, \overline{x_1}$
- Clauses : $\overline{x_1} \vee \overline{x_2} \vee x_3$
- Formule CNF
- Problème SAT : existe-il une interprétation des variables qui satisfait la formule ?

Définitions : le problème SAT

$$\begin{array}{l} \overline{x_1} \vee \overline{x_2} \vee x_3 \\ \wedge \qquad \qquad \overline{x_3} \\ \wedge x_1 \vee x_2 \\ \wedge \qquad x_2 \vee x_3 \end{array}$$

| x_1 | x_2 | x_3 |
|-------|-------|-------|
| F | V | F |

- Variables : $x_1 \dots x_3$
- Littéraux : $x_1, \overline{x_1}$
- Clauses : $\overline{x_1} \vee \overline{x_2} \vee x_3$
- Formule CNF
- Problème SAT : existe-il une interprétation des variables qui satisfait la formule ?

Définitions : le problème SAT

$$\begin{array}{l}
 \overline{x_1} \vee \overline{x_2} \vee x_3 \\
 \wedge \qquad \qquad \overline{x_3} \\
 \wedge x_1 \vee x_2 \\
 \wedge \qquad x_2 \vee x_3
 \end{array}$$

| | | |
|-------|-------|-------|
| x_1 | x_2 | x_3 |
| F | V | F |



- Variables : $x_1 \dots x_3$
- Littéraux : $x_1, \overline{x_1}$
- Clauses : $\overline{x_1} \vee \overline{x_2} \vee x_3$
- Formule CNF
- Problème SAT : existe-il une interprétation des variables qui satisfait la formule ?
- Tester toutes les possibilités : illusoire !

| Nombre d'instructions | Temps nécessaire |
|--------------------------------------|-----------------------------|
| $2^3 = 8$ | instantané |
| $2^{37} \approx 80 \times 10^9$ | 1 seconde |
| $2^{56} \approx 8 \times 10^{16}$ | ≈ 277 heures |
| $2^{60} \approx 10^{18}$ | 166 jours |
| $2^{128} \approx 340 \times 10^{38}$ | ≥ 3 milliards d'années |

Historique

1960 DAVIS et PUTNAM



Martin DAVIS

Historique

1960 DAVIS et PUTNAM

1962 DAVIS, LOGEMANN et LOVELAND

...



Martin DAVIS

Historique

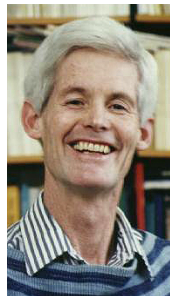
1960 DAVIS et PUTNAM

1962 DAVIS, LOGEMANN et LOVELAND

...

1971 SAT est NP-complet

...



Stephen COOK

Historique

1960 DAVIS et PUTNAM

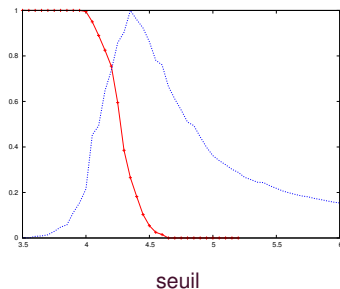
1962 DAVIS, LOGEMANN et LOVELAND

...

1971 SAT est NP-complet

...

1983 Formules aléatoires



Historique

1960 DAVIS et PUTNAM

1962 DAVIS, LOGEMANN et LOVELAND

...

1971 SAT est NP-complet

...

1983 Formules aléatoires

1992 Recherche locale



Historique

1960 DAVIS et PUTNAM

1962 DAVIS, LOGEMANN et LOVELAND

...

1971 SAT est NP-complet

...

1983 Formules aléatoires

1992 Recherche locale

1999 BMC : Utilisation de SAT



Edmund CLARKE

Historique

- 1960 DAVIS et PUTNAM
- 1962 DAVIS, LOGEMANN et LOVELAND
- ...
- 1971 SAT est NP-complet
- ...
- 1983 Formules aléatoires
- 1992 Recherche locale
- 1999 BMC : Utilisation de SAT
- 2001 Solveur CDCL : ZCHAFF



Joao MARQUES-SILVA



Karem SAKALLAH

Historique

1960 DAVIS et PUTNAM

1962 DAVIS, LOGEMANN et LOVELAND

...

1971 SAT est NP-complet

...

1983 Formules aléatoires

1992 Recherche locale

1999 BMC : Utilisation de SAT

2001 Solveur CDCL : ZCHAFF



Lintao ZHANG

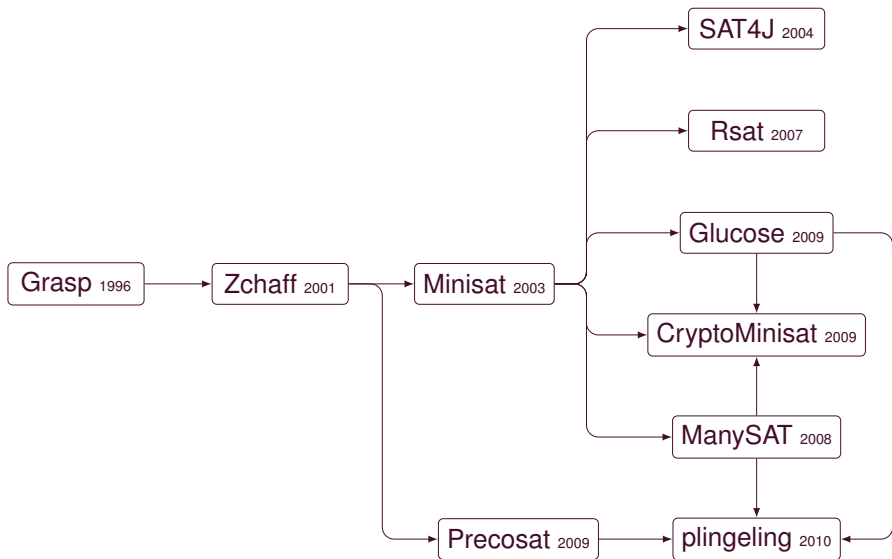


Sharad MALIK

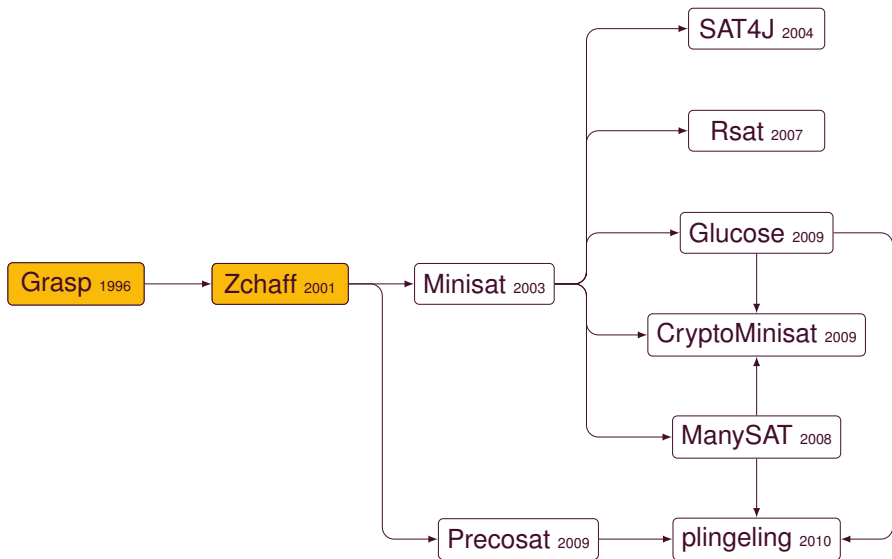
Le tournant des années 2000

- Ces deux articles marquent un tournant !
 - ▶ A. Biere, A. Cimatti, E. Clarke, et Y. Zhu. "*Symbolic Model Checking without BDDs*". 1999.
 - ▶ M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang et S. Malik. « *Chaff : engineering an efficient SAT solver*. 2001.
- On passe de l'utilisation de SAT pour montrer qu'un problème est NP-complet et donc intraitable
- A l'utilisation de SAT pour résoudre un problème intraitable en théorie

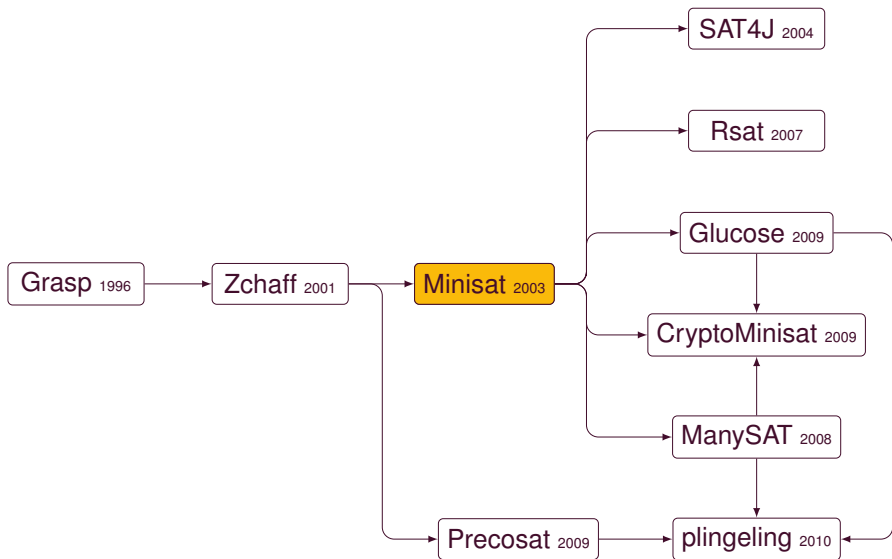
Quelques démonstrateurs SAT



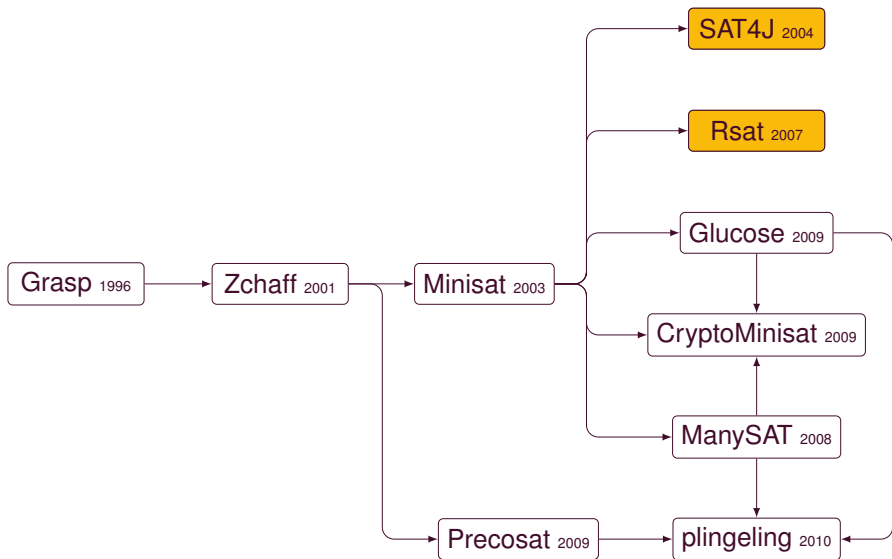
Quelques démonstrateurs SAT



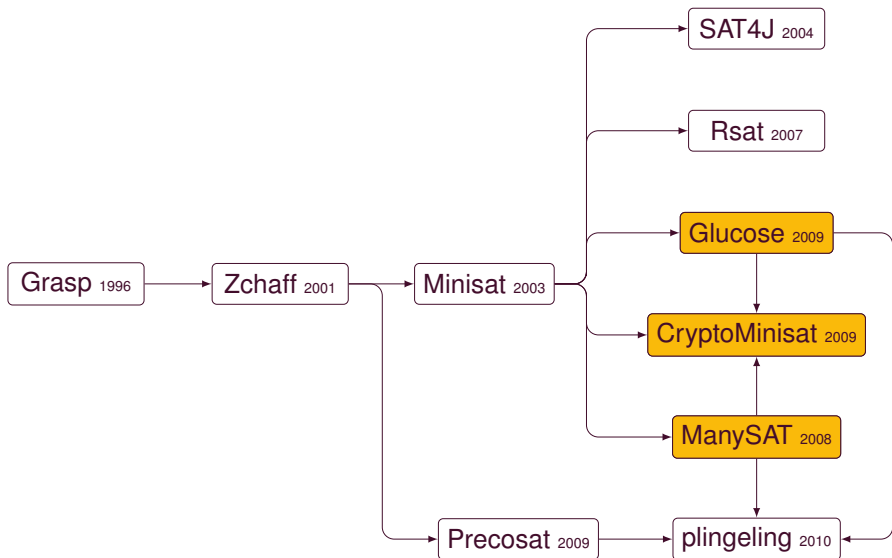
Quelques démonstrateurs SAT



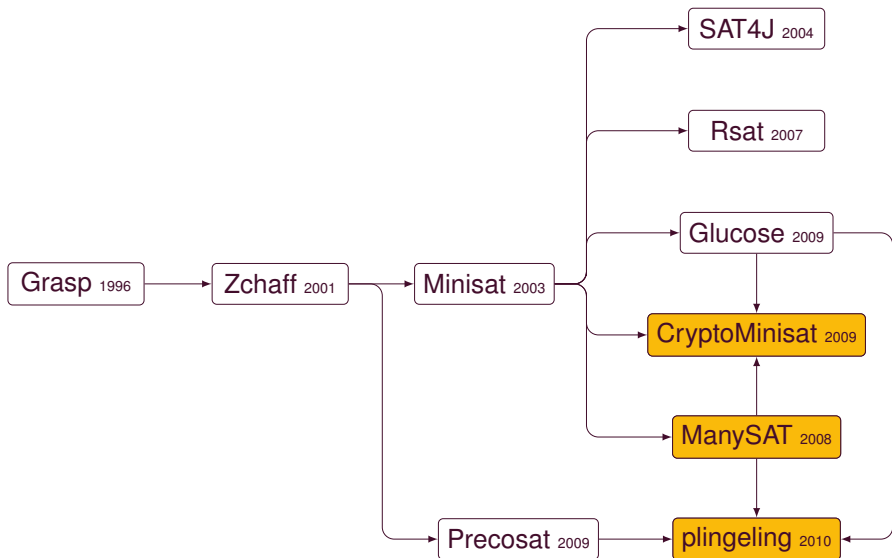
Quelques démonstrateurs SAT



Quelques démonstrateurs SAT

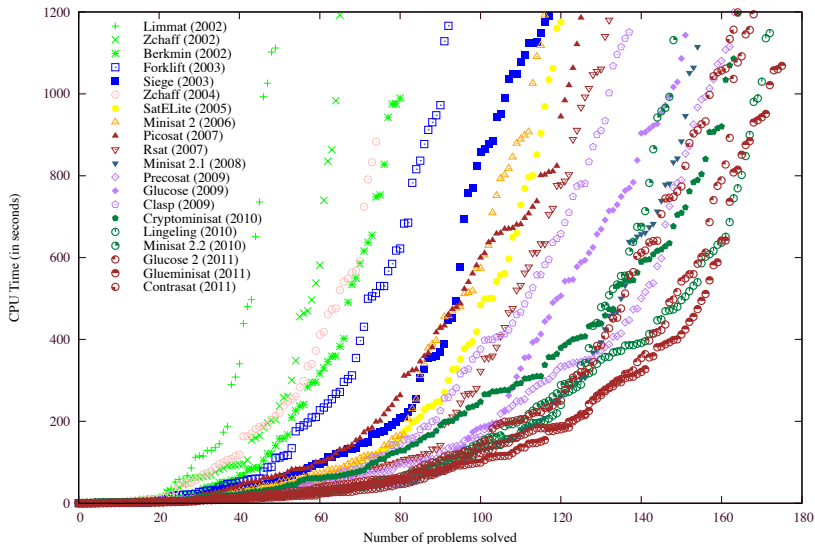


Quelques démonstrateurs SAT



Evolution des démonstrateurs

Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout



Un bref aperçu des démonstrateurs CDCL

Séquence de décision, Propagation

$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$

$$C_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$

$$C_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

Un bref aperçu des démonstrateurs CDCL

Séquence de décision, Propagation



$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$

$$C_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$

$$C_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

Un bref aperçu des démonstrateurs CDCL

Séquence de décision, Propagation



$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$

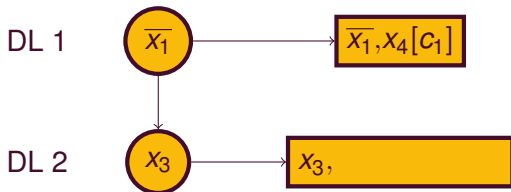
$$C_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$

$$C_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

Un bref aperçu des démonstrateurs CDCL

Séquence de décision, Propagation

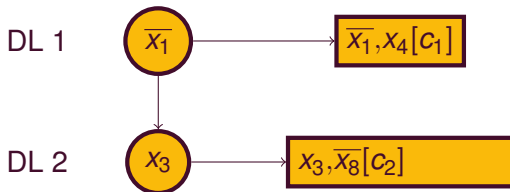
$$\begin{aligned}
 C_1 &= x_1 \vee x_4 \\
 C_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
 C_3 &= x_1 \vee x_8 \vee x_{12} \\
 C_4 &= x_2 \vee x_{11} \\
 C_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
 C_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
 C_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
 \end{aligned}$$



Un bref aperçu des démonstrateurs CDCL

Séquence de décision, Propagation

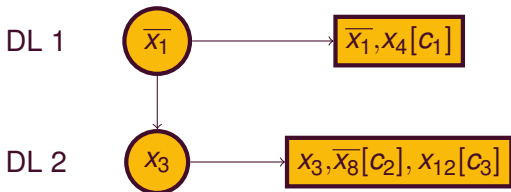
$$\begin{aligned}
 C_1 &= x_1 \vee x_4 \\
 C_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
 C_3 &= x_1 \vee x_8 \vee x_{12} \\
 C_4 &= x_2 \vee x_{11} \\
 C_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
 C_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
 C_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
 \end{aligned}$$



Un bref aperçu des démonstrateurs CDCL

Séquence de décision, Propagation

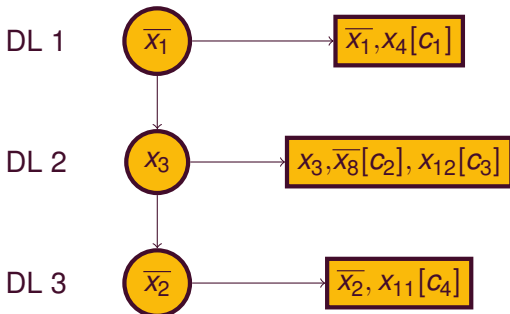
$$\begin{aligned}
 C_1 &= x_1 \vee x_4 \\
 C_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
 C_3 &= x_1 \vee x_8 \vee x_{12} \\
 C_4 &= x_2 \vee x_{11} \\
 C_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
 C_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
 C_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
 \end{aligned}$$



Un bref aperçu des démonstrateurs CDCL

Séquence de décision, Propagation

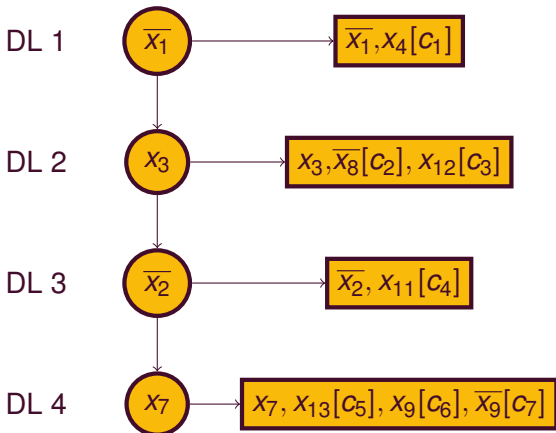
$$\begin{aligned}
 C_1 &= x_1 \vee x_4 \\
 C_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
 C_3 &= x_1 \vee x_8 \vee x_{12} \\
 C_4 &= x_2 \vee x_{11} \\
 C_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
 C_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
 C_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
 \end{aligned}$$



Un bref aperçu des démonstrateurs CDCL

Séquence de décision, Propagation

$$\begin{aligned}
 C_1 &= x_1 \vee x_4 \\
 C_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
 C_3 &= x_1 \vee x_8 \vee x_{12} \\
 C_4 &= x_2 \vee x_{11} \\
 C_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
 C_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
 C_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
 \end{aligned}$$



Un bref aperçu des démonstrateurs CDCL

Analyse de conflits

$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$

$$C_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$

$$C_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

DL 4



Un bref aperçu des démonstrateurs CDCL

Analyse de conflits

$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \bar{x}_3 \vee \bar{x}_8$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \bar{x}_3 \vee \bar{x}_7 \vee x_{13}$$

$$C_6 = \bar{x}_3 \vee \bar{x}_7 \vee \bar{x}_{13} \vee x_9$$

$$C_7 = x_8 \vee \bar{x}_7 \vee \bar{x}_9$$

DL 4



$$d^* = c_7 \otimes_{x_9} c_6 = \bar{x}_3 \vee x_8 \vee \bar{x}_7 \vee \bar{x}_{13}$$

Un bref aperçu des démonstrateurs CDCL

Analyse de conflits

$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \bar{x}_3 \vee \bar{x}_8$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \bar{x}_3 \vee \bar{x}_7 \vee x_{13}$$

$$C_6 = \bar{x}_3 \vee \bar{x}_7 \vee \bar{x}_{13} \vee x_9$$

$$C_7 = x_8 \vee \bar{x}_7 \vee \bar{x}_9$$

DL 4



$$d^* = c_7 \otimes_{x_9} c_6 = \bar{x}_3 \vee x_8 \vee \bar{x}_7 \vee \bar{x}_{13}$$

$$d_1 = d_1 \otimes_{x_{13}} c_5 = \bar{x}_3 \vee x_8 \vee \bar{x}_7$$

Un bref aperçu des démonstrateurs CDCL

Analyse de conflits

$$\begin{aligned}
 C_1 &= x_1 \vee x_4 \\
 C_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
 C_3 &= x_1 \vee x_8 \vee x_{12} \\
 C_4 &= x_2 \vee x_{11} \\
 C_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
 C_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
 C_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
 \end{aligned}$$

DL 4



$$d^* = c_7 \otimes_{x_9} c_6 = \overline{x_3} \vee x_8 \vee \overline{x_7} \vee \overline{x_{13}}$$

$$d_1 = d_1 \otimes_{x_{13}} c_5 = \overline{x_3} \vee x_8 \vee \overline{x_7}$$

- Première résolvente qui ne contient qu'un littéral du niveau de décision courant
- Schéma d'apprentissage appelé le premier UIP
- d_1 est ajoutée à la base des clauses apprises, et...

Un bref aperçu des démonstrateurs CDCL

Backjumping

$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

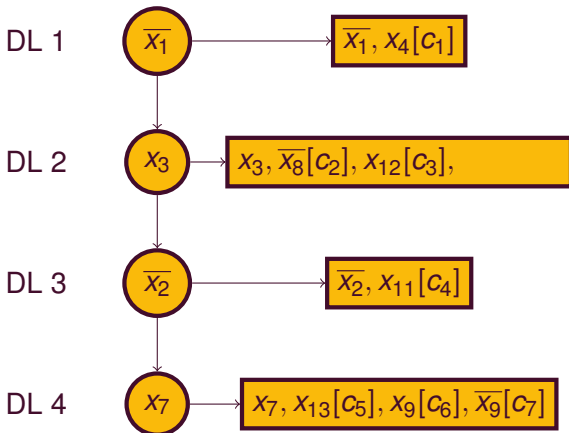
$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$

$$C_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$

$$C_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$d_1 = \overline{x_3} \vee x_8 \vee \overline{x_7}$$



Un bref aperçu des démonstrateurs CDCL

Backjumping

$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \bar{x}_3 \vee \bar{x}_8$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

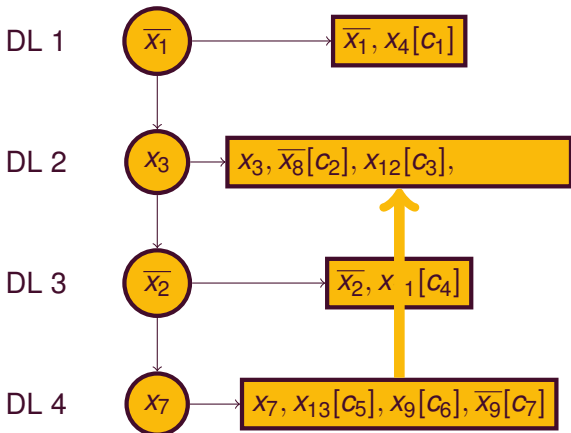
$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \bar{x}_3 \vee \bar{x}_7 \vee x_{13}$$

$$C_6 = \bar{x}_3 \vee \bar{x}_7 \vee \bar{x}_{13} \vee x_9$$

$$C_7 = x_8 \vee \bar{x}_7 \vee \bar{x}_9$$

$$d_1 = \bar{x}_3 \vee x_8 \vee \bar{x}_7$$



Un bref aperçu des démonstrateurs CDCL

Backjumping

$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

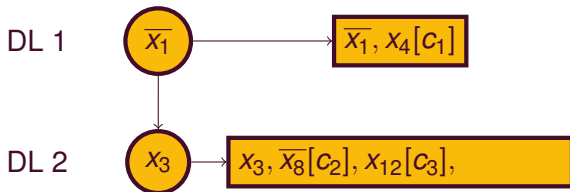
$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$

$$C_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$

$$C_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$d_1 = \overline{x_3} \vee x_8 \vee \overline{x_7}$$



Un bref aperçu des démonstrateurs CDCL

Backjumping

$$C_1 = x_1 \vee x_4$$

$$C_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$

$$C_3 = x_1 \vee x_8 \vee x_{12}$$

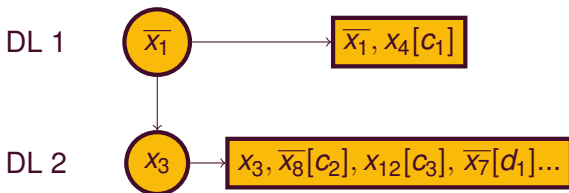
$$C_4 = x_2 \vee x_{11}$$

$$C_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$

$$C_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$

$$C_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

$$d_1 = \overline{x_3} \vee x_8 \vee \overline{x_7}$$

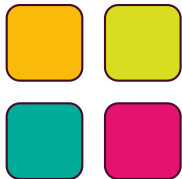


Autres composants essentiels

- Heuristique de choix de variables
 - ▶ Dynamiques : récompensent les variables les plus récemment utilisées dans l'analyse de conflits
 - ▶ Enregistrement de la phase (ne pas oublier le passé)

- Redémarrages : statiques ou dynamiques

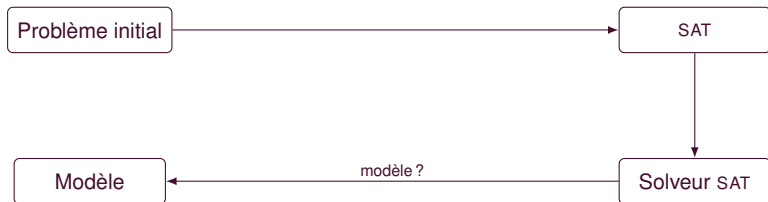
- Le nettoyage de la base de clauses
 - ▶ Pour éviter une explosion de la mémoire utilisée, il est nécessaire de supprimer certaines clauses apprises



Modélisation

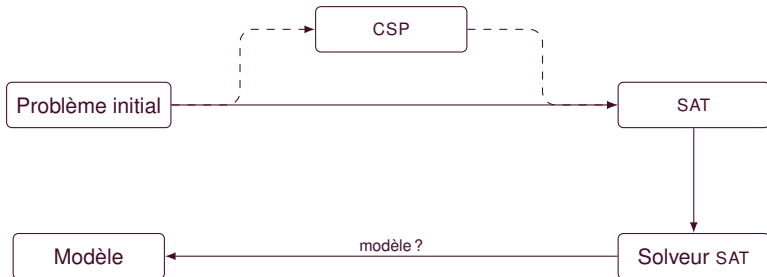
Introduction

- Modélisation de problèmes en SAT
- Si un modèle est découvert, il faudra le transformer dans la syntaxe du problème initial



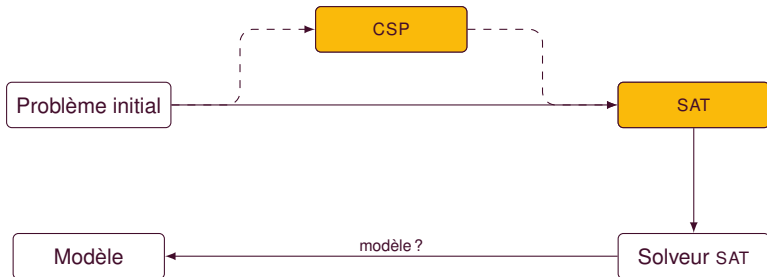
Introduction

- Modélisation de problèmes en SAT
- Si un modèle est découvert, il faudra le transformer dans la syntaxe du problème initial
- On peut également partir du problème CSP et le traduire en SAT
 - ▶ Etape supplémentaire
 - ▶ Peut sembler inutile, mais le codage en CSP paraît plus naturel



Introduction

- Modélisation de problèmes en SAT
- Si un modèle est découvert, il faudra le transformer dans la syntaxe du problème initial
- On peut également partir du problème CSP et le traduire en SAT
 - ▶ Etape supplémentaire
 - ▶ Peut sembler inutile, mais le codage en CSP paraît plus naturel



Codage des variables

- Comment coder une variable CSP x_i et son domaine $\{a, b, c\}$

Codage des variables

- Comment coder une variable CSP x_i et son domaine $\{a, b, c\}$
- Une première solution
 - ▶ Une variable booléenne représentant $x_i = k$ x_a^i, x_b^i, x_c^i

Codage des variables

- Comment coder une variable CSP x_i et son domaine $\{a, b, c\}$
- Une première solution
 - ▶ Une variable booléenne représentant $x_i = k$ x_a^i, x_b^i, x_c^i
 - ▶ Une clause (*at least*) nous disant que x_i doit avoir une valeur $x_a^i \vee x_b^i \vee x_c^i$

Codage des variables

- Comment coder une variable CSP x_i et son domaine $\{a, b, c\}$
- Une première solution
 - ▶ Une variable booléenne représentant $x_i = k \quad x_a^i, x_b^i, x_c^i$
 - ▶ Une clause (*at least*) nous disant que x_i doit avoir une valeur $x_a^i \vee x_b^i \vee x_c^i$
 - ▶ Des clauses (*at most*) nous disant que x_i ne peut pas avoir plus d'une valeur : $(\overline{x_a^i} \vee \overline{x_b^i}), (\overline{x_a^i} \vee \overline{x_c^i}) \dots$

Codage des variables

- Comment coder une variable CSP x_i et son domaine $\{a, b, c\}$
- Une première solution
 - ▶ Une variable booléenne représentant $x_i = k \quad x_a^i, x_b^i, x_c^i$
 - ▶ Une clause (*at least*) nous disant que x_i doit avoir une valeur $x_a^i \vee x_b^i \vee x_c^i$
 - ▶ Des clauses (*at most*) nous disant que x_i ne peut pas avoir plus d'une valeur : $(\overline{x_a^i} \vee \overline{x_b^i}), (\overline{x_a^i} \vee \overline{x_c^i}) \dots$
- Pour chaque variable x_i de domaine de taille n , il faut donc
 - ▶ n variables booléennes
 - ▶ $1 + \frac{n \times (n-1)}{2}$ clauses

Codage des contraintes

- Les contraintes peuvent être codées de diverses manières
 - ▶ Codage direct
 - ▶ Codage support
- Attention : cela nécessite de connaître les tuples interdits ou les tuples autorisés d'une contrainte
- On ne peut pas traiter directement les contraintes en intention ou les contraintes globales (il faut expliciter les tuples)

Codage direct

- Etant donnée une contrainte C de scope $scp(C) = (x_i, x_j, x_k)$
- Le codage direct [Walsh 00] va ajouter une clause pour chaque tuple interdit de la contrainte C
- Pour chaque tuple interdit, (a, a, b) , on ajoute la clause $\overline{x}_a^i \vee \overline{x}_a^j \vee \overline{x}_b^k$
- Les tuples interdits falsifient ainsi les clauses associées !

Exemple

- Soit $x, y \in \{2, 3, 4, 5, 6\}$
- Représenter les variables et les clauses nécessaires à la contrainte :
 $x + y \leq 7$

Exemple

- Soit $x, y \in \{2, 3, 4, 5, 6\}$
- Représenter les variables et les clauses nécessaires à la contrainte : $x + y \leq 7$

$x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \qquad y_2 \quad y_3 \quad y_4 \quad y_5 \quad y_6$

$(x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6)$

$(\overline{x_2} \vee \overline{x_3}) \quad (\overline{x_2} \vee \overline{x_4}) \quad (\overline{x_2} \vee \overline{x_5}) \quad (\overline{x_2} \vee \overline{x_6}) \quad (\overline{x_3} \vee \overline{x_4}) \quad (\overline{x_3} \vee \overline{x_5}) \quad (\overline{x_3} \vee \overline{x_6}) \quad (\overline{x_4} \vee \overline{x_5})$
 $(\overline{x_4} \vee \overline{x_6}) \quad (\overline{x_5} \vee \overline{x_6})$

$(y_2 \vee y_3 \vee y_4 \vee y_5 \vee y_6)$

$(\overline{y_2} \vee \overline{y_3}) \quad (\overline{y_2} \vee \overline{y_4}) \quad (\overline{y_2} \vee \overline{y_5}) \quad (\overline{y_2} \vee \overline{y_6}) \quad (\overline{y_3} \vee \overline{y_4}) \quad (\overline{y_3} \vee \overline{y_5}) \quad (\overline{y_3} \vee \overline{y_6}) \quad (\overline{y_4} \vee \overline{y_5})$
 $(\overline{y_4} \vee \overline{y_6}) \quad (\overline{y_5} \vee \overline{y_6})$

$(\overline{x_2} \vee \overline{y_6}) \quad (\overline{x_3} \vee \overline{y_5}) \quad (\overline{x_3} \vee \overline{y_6}) \quad (\overline{x_4} \vee \overline{y_4}) \quad (\overline{x_4} \vee \overline{y_5}) \quad (\overline{x_4} \vee \overline{y_6}) \quad (\overline{x_5} \vee \overline{y_3}) \quad (\overline{x_5} \vee \overline{y_4})$
 $(\overline{x_5} \vee \overline{y_5}) \quad (\overline{x_5} \vee \overline{y_6}) \quad (\overline{x_6} \vee \overline{y_2}) \quad (\overline{x_6} \vee \overline{y_3}) \quad (\overline{x_6} \vee \overline{y_4}) \quad (\overline{x_6} \vee \overline{y_5}) \quad (\overline{x_6} \vee \overline{y_6})$

Codage support

- Le codage direct ne permet pas d'inférer (propager) autant que l'arc consistance !
- C'est pour cela qu'a été proposé le codage support [Gent 02]
- Le codage support code non seulement la structure du réseau mais aussi l'algorithme (AC) utilisé pour le résoudre
- Etant donné une contrainte C de scope $scp(C) = (x^i, x^j)$
- si $x^i = a$ possède comme support $\{a, b, c\} \subseteq D(x^j)$
- On ajoute alors la clause : $(\overline{x_a^i} \vee x_a^j \vee x_b^j \vee x_c^j)$
- $x^i = a$ est possible tant qu'il lui reste un support dans la contrainte !

Contrainte d'arité supérieure

- Ce codage ne fonctionne que pour les contraintes binaires !
- On peut l'étendre aux contraintes n-aires [Bessiere 03]
- Etant donnée une contrainte C de scope $scp(C) = (x^i, x^j, x^k)$
- si $x^i = a$ possède comme support $\{(a, b), (a, e)\} \subseteq D(x^j) \times D(x^k)$
- On ajoute alors les clauses
 - ▶ $s_1 \leftrightarrow (x_a^j \wedge x_b^k)$
 - ▶ $s_2 \leftrightarrow (x_a^j \wedge x_e^k)$
 - ▶ $\overline{x_a^i} \vee s_1 \vee s_2$
- Nécessite des variables additionnelles

Exemple

- Soit $x, y \in \{2, 3, 4, 5, 6\}$
- Représenter les variables et les clauses nécessaires à la contrainte :
 $x + y \leq 7$

Exemple

- Soit $x, y \in \{2, 3, 4, 5, 6\}$
- Représenter les variables et les clauses nécessaires à la contrainte : $x + y \leq 7$

$x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \qquad y_2 \quad y_3 \quad y_4 \quad y_5 \quad y_6$

$(x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6)$

$(\overline{x_2} \vee \overline{x_3}) \quad (\overline{x_2} \vee \overline{x_4}) \quad (\overline{x_2} \vee \overline{x_5}) \quad (\overline{x_2} \vee \overline{x_6}) \quad (\overline{x_3} \vee \overline{x_4}) \quad (\overline{x_3} \vee \overline{x_5}) \quad (\overline{x_3} \vee \overline{x_6}) \quad (\overline{x_4} \vee \overline{x_5})$
 $(\overline{x_4} \vee \overline{x_6}) \quad (\overline{x_5} \vee \overline{x_6})$

$(y_2 \vee y_3 \vee y_4 \vee y_5 \vee y_6)$

$(\overline{y_2} \vee \overline{y_3}) \quad (\overline{y_2} \vee \overline{y_4}) \quad (\overline{y_2} \vee \overline{y_5}) \quad (\overline{y_2} \vee \overline{y_6}) \quad (\overline{y_3} \vee \overline{y_4}) \quad (\overline{y_3} \vee \overline{y_5}) \quad (\overline{y_3} \vee \overline{y_6}) \quad (\overline{y_4} \vee \overline{y_5})$
 $(\overline{y_4} \vee \overline{y_6}) \quad (\overline{y_5} \vee \overline{y_6})$

$(\overline{x_2} \vee y_2 \vee y_3 \vee y_4 \vee y_5) \quad (\overline{x_3} \vee y_2 \vee y_3 \vee y_4) \quad (\overline{x_4} \vee y_2 \vee y_3) \quad (\overline{x_5} \vee y_2) \quad (\overline{x_6})$
 $(\overline{y_2} \vee x_2 \vee x_3 \vee x_4 \vee x_5) \quad (\overline{y_3} \vee x_2 \vee x_3 \vee x_4) \quad (\overline{y_4} \vee x_2 \vee x_3) \quad (\overline{y_5} \vee x_2) \quad (\overline{y_6})$

Codage Mix

- Dès lors que les domaines des variables sont importants, le choix entre l'utilisation des tuples interdits ou autorisés n'est plus de mise !
- Il faut prendre le plus concis des 2
 - ▶ Etant donné une contrainte C de scope $scp(C) = (x^i, x^j, x^k)$
 - ▶ Taille des domaines : 50
 - ▶ La contrainte à 20 couples interdits
 - ▶ Et donc : $50^3 - 20 = 124\,980$ couples autorisés
- En même temps, plus une contrainte possède de couples autorisés, moins AC est efficace

Un autre codage : le codage ordonné

- Proposé pour résoudre des problèmes d'ordonnement [Tamura 09]
- On se place dans le cas où les variables prennent leurs valeurs dans $\{l_x, 2, 3 \dots u_x\}$.
- l_x et u_x sont les bornes inférieures et supérieures que peut prendre x
- On va utiliser $(u_x - l_x + 1)$ variables booléennes
 - ▶ x_k représente $x \leq k$ (pour $l_x - 1 \leq k \leq u_x$)
- Et les contraintes suivantes
 - ▶ $\overline{x_{l_x-1}}$
 - ▶ x_{u_x}
 - ▶ $\overline{x_{k-1}} \vee x_k$ pour $l_x \leq k \leq u_x$
- Une fois cela fait, on va coder les clauses représentant des régions entières

Codage ordonné : les contraintes

- Ce codage est particulièrement adapté aux problèmes contenant de l'arithmétique : Ordonnancement, sac à dos,...
- Soit $x, y \in \{2, 3, 4, 5, 6\}$
- Représenter les variables et les clauses nécessaires à la contrainte :
 $x + y \leq 7$

Codage ordonné : les contraintes

- Ce codage est particulièrement adapté aux problèmes contenant de l'arithmétique : Ordonnancement, sac à dos,...
- Soit $x, y \in \{2, 3, 4, 5, 6\}$
- Représenter les variables et les clauses nécessaires à la contrainte : $x + y \leq 7$

$x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \qquad y_1 \quad y_2 \quad y_3 \quad y_4 \quad y_5 \quad y_6$

$\bar{x}_1 \quad x_6 \quad (\bar{x}_1 \vee x_2) \quad (\bar{x}_2 \vee x_3) \quad (\bar{x}_3 \vee x_4) \quad (\bar{x}_4 \vee x_5) \quad (\bar{x}_5 \vee x_6)$

$\bar{y}_1 \quad y_6 \quad (\bar{y}_1 \vee y_2) \quad (\bar{y}_2 \vee y_3) \quad (\bar{y}_3 \vee y_4) \quad (\bar{y}_4 \vee y_5) \quad (\bar{y}_5 \vee y_6)$

$(x_1 \vee y_5) \quad (x_2 \vee y_4) \quad (x_3 \vee y_3) \quad (x_4 \vee y_2) \quad (x_5 \vee y_1)$

Codage ordonné : résultats

- `Sugar` est le démonstrateur implantant ce codage
- Autorise des contraintes en intention
- Bons résultats aux évaluations CSP
 - ▶ Résolution de problèmes ouverts jusque là
 - ▶ Y compris sur certaines contraintes globales !!

Codage dangereux

- Supposons un problème modélisé difficile à résoudre
- Cela vient il
 - ▶ Du problème en lui même ?
 - ▶ De la modélisation ?

Codage dangereux

- Supposons un problème modélisé difficile à résoudre
- Cela vient il
 - ▶ Du problème en lui même ?
 - ▶ De la modélisation ?
- Dans [Hertel07], les auteurs proposent deux codages d'un même problème et montrent une séparation exponentielle entre les deux

Codage dangereux

- Supposons un problème modélisé difficile à résoudre
- Cela vient il
 - ▶ Du problème en lui même ?
 - ▶ De la modélisation ?
- Dans [Hertel07], les auteurs proposent deux codages d'un même problème et montrent une séparation exponentielle entre les deux
- Dans les années 90, un challenge important :
 - ▶ Résoudre les instances `par32`
 - ▶ Des démonstrateurs spécialisés ont été développés pour atteindre ce but (eqSat z, [Li 00])
 - ▶ Dans [Bailleux 03], les auteurs ont montré que ce n'est pas le problème qui était difficile mais son codage en SAT.

Un exemple : le problème des pigeons

- Un autre exemple : le problème des pigeons
- Déjà vu dans la première partie du cours
- On doit mettre n pigeons dans $(n - 1)$ pigeonniers : un seul pigeon par pigeonnier
- Evidemment insatisfiable
- Et pourtant

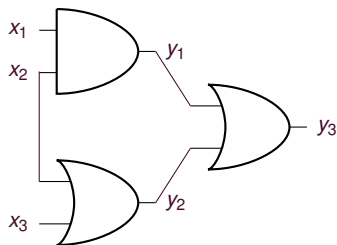
Le codage des pigeons

- Les variables p_h^p : Le pigeon p est dans le pigeonnier h
- Les clauses
 - ▶ Le pigeon i est un dans un pigeonniers : $p_1^i \vee p_2^i \dots p_{(n-1)}^i$
 - ▶ Les pigeons i et j ne peuvent pas être dans le même pigeonnier : $\overline{p_i^p} \vee \overline{p_j^p}$
- Testons cela : comportement exponentiel
- Ajoutons quelques contraintes :
 - ▶ Les pigeons sont tous les mêmes, les pigeonniers aussi : **Symétries**
 - ▶ En codant ces symétries, on rend le problème extrêmement simple

Un autre exemple : codage de circuits

- SAT est le langage parfait pour coder des circuits
- Très utilisé : EDA (Electronic Design Automation)
- De la même manière : cryptanalyse, beaucoup de `xor`
- Des démonstrateurs dédiés : `CryptoMinisat`

Codage des circuits



- 6 variables booléennes :
 $x_1, x_2, x_3, y_1, y_2, y_3$
- Variables d'entrées x_i
- Variables auxiliaires y_i

$$(y_1 \vee \bar{x}_1 \vee \bar{x}_2)$$

$$(\bar{y}_1 \vee x_1)$$

$$(\bar{y}_1 \vee x_2)$$

$$(\bar{y}_2 \vee x_2 \vee x_3)$$

$$(y_2 \vee \bar{x}_2)$$

$$(y_2 \vee \bar{x}_3)$$

$$(\bar{y}_3 \vee y_1 \vee y_2)$$

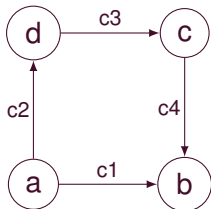
$$(y_3 \vee \bar{y}_1)$$

$$(y_3 \vee \bar{y}_2)$$

$$y_3$$

BMC

- Premier domaine où l'utilisation de SAT s'est montrée performante
- Model Checking ou vérification formelle
- On dispose d'un automate à états et on souhaite vérifier qu'une propriété donnée est vraie
- Utilisée pour la vérification des circuits intégrés ou la vérification de logiciels



Propriété : Est on sur d'arriver à l'état a ?

Fonctionnement

- En BMC, plutôt que de prouver que la propriété est vraie, on va essayer de trouver un contre exemple en bornant à k le nombre d'étapes
 - ▶ A l'étape i on se trouve dans l'état s_i
- Formule SAT : $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \dots \wedge T(s_{k-1}, s_k) \wedge \overline{p(s_k)}$
- Dans le cas où l'instance est satisfiable : la propriété est fausse (bug)
- Dans le cas où l'instance est insatisfiable : on ne sait rien !
- Il faut augmenter la borne max

Optimisation

- Dans le même esprit que précédemment, comment faire pour trouver une solution optimale ?
- En effet, le problème SAT est un problème de décision, réponse oui ou non
- Il faut essayer de trouver une solution d'un cout k donné
- Dans le cas où le problème est satisfiable, il faut augmenter (ou réduire) la borne k
- Malgré tout, les démonstrateurs SAT s'avèrent quelque fois très performants sur ce type de problème (cf. `sugar`)

Avantages et inconvénients de SAT

■ Avantages

- ▶ Langage très simple
- ▶ L'apprentissage : les nogoods sont codés directement dans le langage de départ (clauses)

Avantages et inconvénients de SAT

■ Avantages

- ▶ Langage très simple
- ▶ L'apprentissage : les nogoods sont codés directement dans le langage de départ (clauses)

■ Inconvénients

- ▶ Un niveau d'abstraction élevé
- ▶ Pas d'optimisation
- ▶ Des problèmes inaccessibles (trop gros)

Avantages et inconvénients de CSP

■ Avantages

- ▶ Expressivité : contraintes en intention, contraintes globales
- ▶ Optimisation

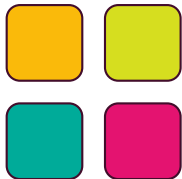
Avantages et inconvénients de CSP

■ Avantages

- ▶ Expressivité : contraintes en intention, contraintes globales
- ▶ Optimisation

■ Inconvénients

- ▶ Des démonstrateurs moins efficaces ?



Extensions

Introduction

- Pour palier les différents inconvénients listés ci-dessus, des extensions ont été proposées
 - ▶ MAXSAT : Problème d'optimisation associé à SAT
 - ▶ Pseudo Booléens : expressivité plus importante et optimisation
 - ▶ SMT : SAT modulo théorie : expressivité plus importante

Pseudo booléens

- Formalisme plus expressif
- Les clauses : $a_1 \times x_1 + a_2 \times x_2 \dots a_n \times x_n \quad op \quad a$
- $a_j, a \in N$
- $op \in \{<, >, =, \leq, \geq\}$
- Les variables sont toujours booléennes, mais on a maintenant des expressions mathématiques
- On peut également introduire une fonction d'optimisation (de la même forme)

Codage des pigeons

- Les variables p_h^p : Le pigeon p est dans le pigeonnier h
- Un pigeonnier contient un pigeon : $p_h^1 + p_h^2 + \dots + p_h^n = 1$
- Un pigeon est dans un pigeonnier : $p_1^p + p_2^p + \dots + p_{n-1}^p = 1$

MaxSAT

- Problème d'optimisation associé à SAT
- Satisfaire le plus de clauses possibles
- Une version plus adaptée aux problèmes réels : MAXSAT pondéré
- Les clauses ont chacune un poids (appartenant à $[0, \infty]$)
- Maximiser le poids des clauses satisfaites
- Clauses Dures (poids ∞) :
 - ▶ Pas deux cours dans la même salle, pas d'enseignant dans deux salles différentes
- Clauses souples :
 - ▶ Je veux commencer à 10h et terminer à 16h....

Avantages et inconvénients

- MAXSAT et pseudo sont des problèmes d'optimisation
- Pseudo est une formalisme plus expressif
- Pour autant, beaucoup de démonstrateurs passent par un solveur SAT pour résoudre ces problèmes
- Ceci vient de l'efficacité des solveurs SAT
- N'est pas viable à long terme ??

Bibliographie

■ Base sur SAT

- ▶ [sat 08] Lakhdar Saïs. Problème SAT : Progrès et défis. Hermes editions Lavoisier, 2008.
- ▶ [Audemard 10] : Gilles Audemard. Habilitation à diriger les recherches, 2010.

■ Démonstrateurs SAT

- ▶ [Silva 96] Joao Marques Silva, Karem Sakallah. GRASP - a new search algorithm for satisfiability. ICCAD pp 220-227, 1996
- ▶ [Moskewicz 01] Matthew Moskewicz, Conor Madigan, Ying Zhao, Lintao Zhang, Sharad Malik : Chaff : Engineering an Efficient SAT Solver. DAC, pp 530-535, 2001

■ Différents codages

- ▶ [Walsh 00] Toby Walsh. SAT v CSP. CP, pp 441-456, 2000.
- ▶ [Gent 02] Ian Gent. Arc consistencies in SAT. ECAI, pp 121-125, 2002.
- ▶ [Bessiere 03] Christian Bessiere, Emmanuel Hebrard, Toby Walsh. Local consistencies in SAT. SAT, pp 299-314, 2003.
- ▶ [Tamura 09] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, Mutsunori Banbara. Compiling fininte linear CSP into SAT. Constraints, pp 254-272, 2009.

Bibliographie

■ Codages dangereux

- ▶ [Bailleux 03] Olivier Bailleux, Yacine Boufkhad : Efficient CNF Encoding of Boolean Cardinality Constraints. CP, pages 108-122, 2003.
- ▶ [Hertel 07] Alexander Hertel, Philipp Hertel, Alasdair Urquhart. Formalizing Dangerous SAT encodings. SAT, pp 159-172, 2007.
- ▶ [LI 00] Chu Min LI. Integrating Equivalency reasoning into Davis-Putnam procedure. AAAI, pp 291-296, 2000.

■ MAXSAT

- ▶ [Li 08] Chu Min Li. Chapitre dans [sat 08]

■ Pseudo booleés

- ▶ [Roussel 09] Olivier Roussel et Vasco Manquinho. Handbook of Satisfiability, vol. 2, chapitre 22, pp. 695-733, 2009.