Modeling and solving problems with SAT

Jean-Marie Lagniez

14th December 2023

GDR ROD & RADIA

Outline

Introduction

- SAT solvers are generally efficient when it comes to tackle NP-hard problems and beyond NP problems
- To do so, we have to write our lovely problems into a set of clauses
- But sometimes that does not work as expected :'(
- Two different case studies where such a situation occurs will be presented and discussed:
    - MUS extraction
    - Team Formation

The SAT problem

$$\begin{aligned}\Sigma = & (\neg a \vee \neg b \vee \neg c) \\ & \wedge (a \vee c) \\ & \wedge (a \vee b) \\ & \wedge (\neg b \vee \neg c)\end{aligned}$$

- Propositional variables: a, b, c
- Literals: $a, \neg a$
- Clauses: $a \vee \neg b$ (the constraints)
- CNF formula: Σ
- SAT problem: can we find an interpretation $\mathcal{I}$ of the variables that satisfies the formula?

The SAT problem

$$\Sigma = (\neg a \vee \neg b \vee \neg c) \\ \wedge (a \vee c) \\ \wedge (a \vee b)$$

a	b	c
$\perp$	$\perp$	$\perp$

- Propositional variables: a, b, c
- Literals: $a, \neg a$
- Clauses: $a \vee \neg b$ (the constraints)
- CNF formula: Σ
- SAT problem: can we find an interpretation $\mathcal{I}$ of the variables that satisfies the formula?

The SAT problem

$$\Sigma = (\neg a \vee \neg b \vee \neg c) \wedge (a \vee c) \wedge (a \vee b)$$

a	b	c
$\top$	$\bot$	$\bot$

- Propositional variables: a, b, c
- Literals: $a, \neg a$
- Clauses: $a \vee \neg b$ (the constraints)
- CNF formula: Σ
- SAT problem: can we find an interpretation $\mathcal{I}$ of the variables that satisfies the formula?

The SAT problem

$$\Sigma = (\neg a \vee \neg b \vee \neg c) \wedge (a \vee c) \wedge (a \vee b)$$

a	b	c
$\top$	$\bot$	$\bot$

- Propositional variables: a, b, c
- Literals: $a, \neg a$
- Clauses: $a \vee \neg b$ (the constraints)
- CNF formula: Σ
- SAT problem: can we find an interpretation $\mathcal{I}$ of the variables that satisfies the formula?
- Try all the possibility: illusory!

Number of instructions	Time needed
$2^3 = 8$	immediate
$2^{37} = 80 \times 10^9$	1 second
$2^{56} = 8 \times 10^{16}$	$\approx$ 277 hours
$2^{60} = 10^{18}$	166 days
$2^{128} = 340 \times 10^{38}$	$\geq$ 3 billion of years

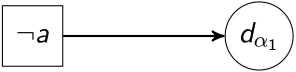
What is a CDCL SAT solver?

- **Extend DPLL SAT** solver with:
  - Clause learning and non-chronological backtracking
    - Exploit UIPs
    - Minimize learned clauses
    - Opportunistically delete clauses
  - Can **restart** the current search
  - **Lazy data structures**
    - Watched literals
  - **Conflict-guiding** branching
    - Lightweight branching heuristics
    - Phase saving

CDCL SAT solver ingredients

- Affectation, unit propagation
  - heuristic to choose the next variable to assign
  - heuristic to choose its polarity
  - unit propagation

$$\Sigma = \{\alpha_1 : a \vee d\}$$



- Conflict analysis and learning
  - **implication graph**
  - **learning**
  - **back-jumping**

Constructing and analyzing the implication graph

Conflict graph construction

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation

Conflict graph construction

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation

$\boxed{\neg a^1}$

Conflict graph construction

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation

$\boxed{\neg a^1}$

Conflict graph construction

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation

$$\boxed{\neg a^1} \longrightarrow d^1_{\alpha_1}$$

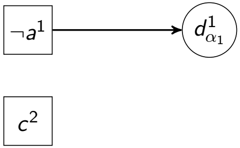
Conflict graph construction

$\alpha_1 : a \vee d \qquad \alpha_2 : a \vee \neg c \vee \neg f \qquad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h \qquad \alpha_5 : \neg c \vee \neg e \vee i \qquad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k \qquad \alpha_8 : e \vee \neg h \vee k \qquad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



Conflict graph construction

$$
\begin{array}{lll}
\alpha_1 : a \vee d & \alpha_2 : a \vee \neg c \vee \neg f & \alpha_3 : \neg d \vee j \vee f \\
\alpha_4 : b \vee h & \alpha_5 : \neg c \vee \neg e \vee i & \alpha_6 : \neg i \vee \neg j \vee \neg g \\
\alpha_7 : e \vee \neg k & \alpha_8 : e \vee \neg h \vee k & \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g
\end{array}
$$

Affectation, Propagation

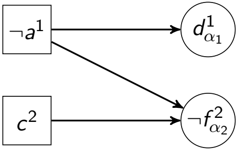
$$\boxed{\neg a^1} \longrightarrow d^1_{\alpha_1}$$

$$\boxed{c^2}$$

Conflict graph construction

$$\begin{array}{lll}
\alpha_1 : a \vee d & \alpha_2 : a \vee \neg c \vee \neg f & \alpha_3 : \neg d \vee j \vee f \\
\alpha_4 : b \vee h & \alpha_5 : \neg c \vee \neg e \vee i & \alpha_6 : \neg i \vee \neg j \vee \neg g \\
\alpha_7 : e \vee \neg k & \alpha_8 : e \vee \neg h \vee k & \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g
\end{array}$$

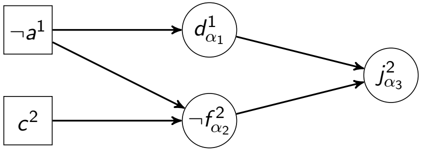
Affectation, Propagation



Conflict graph construction

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



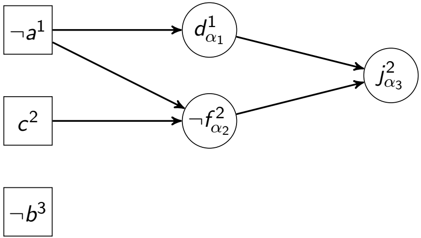
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



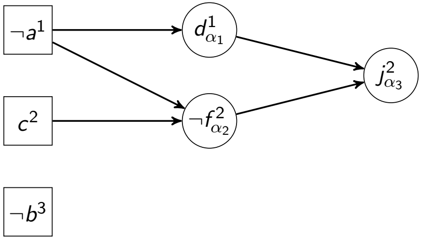
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

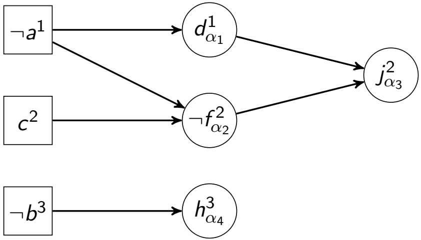
Affectation, Propagation



Conflict graph construction

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



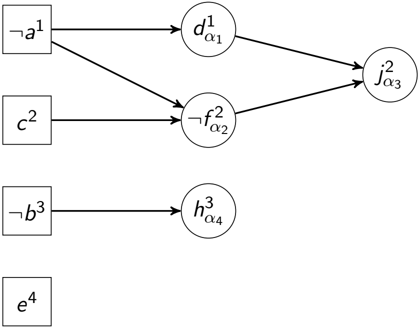
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



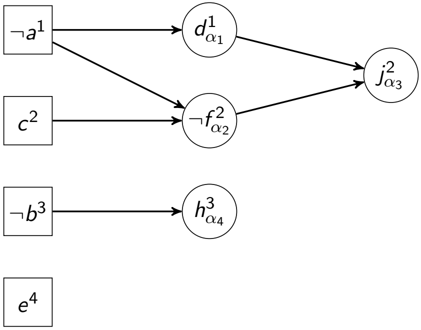
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

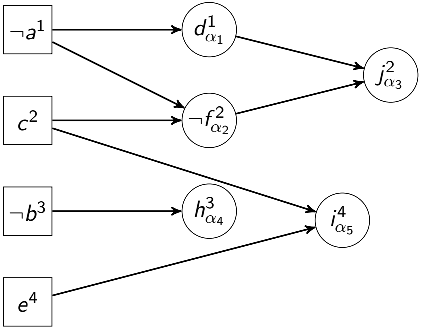
Affectation, Propagation



Conflict graph construction

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



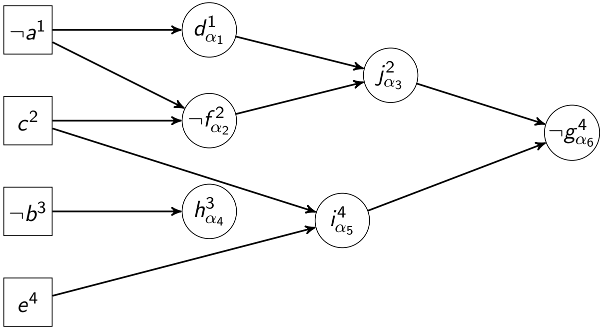
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad\alpha_2 : a \vee \neg c \vee \neg f$ $\quad\alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad\alpha_5 : \neg c \vee \neg e \vee i$ $\quad\alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad\alpha_8 : e \vee \neg h \vee k$ $\quad\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



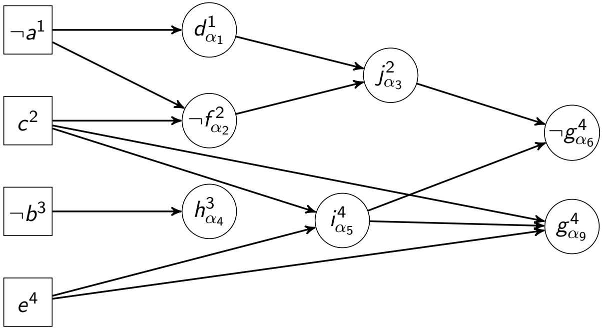
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad$ $\alpha_2 : a \vee \neg c \vee \neg f$ $\quad$ $\alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad$ $\alpha_5 : \neg c \vee \neg e \vee i$ $\quad$ $\alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad$ $\alpha_8 : e \vee \neg h \vee k$ $\quad$ $\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



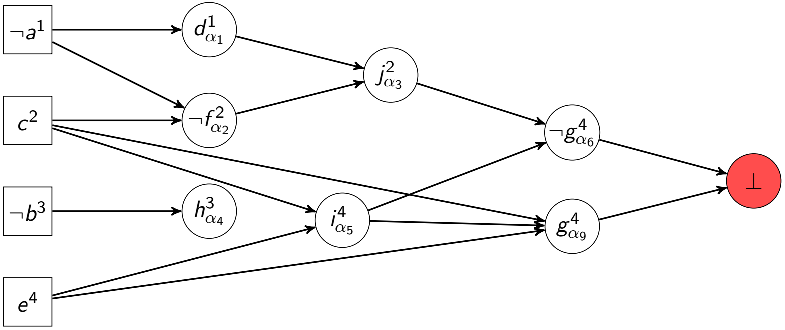
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



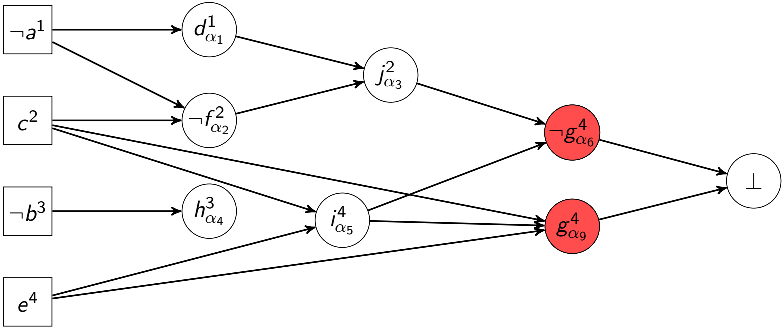
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

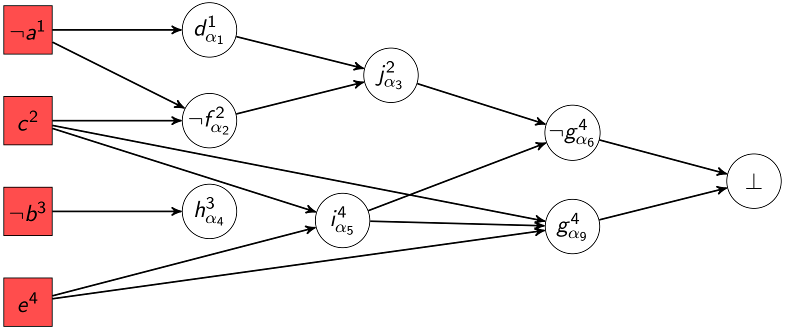
Affectation, Propagation



Conflict graph construction

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation



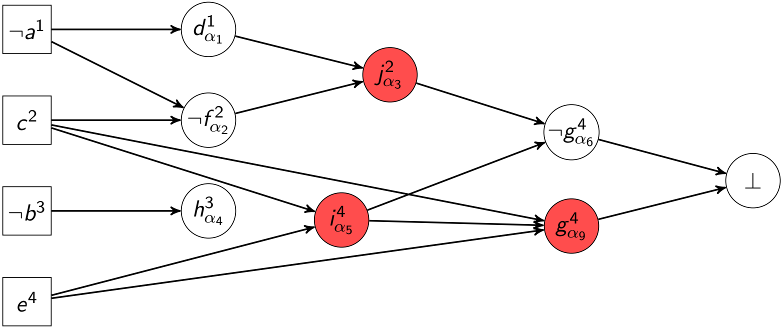
Conflict graph construction

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

Affectation, Propagation

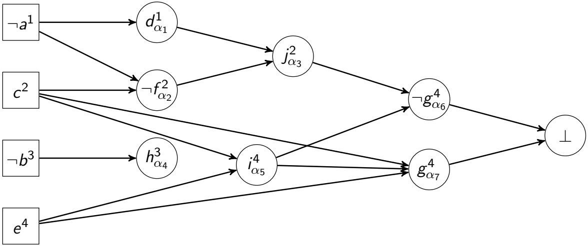


Conflict graph analysis

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

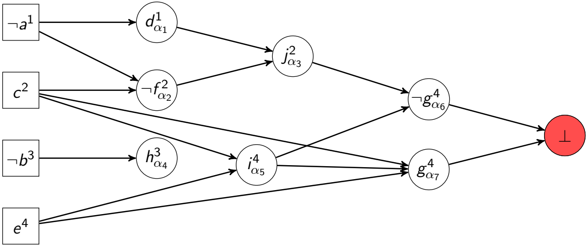


Conflict graph analysis

$\alpha_1 : a \lor d$ $\qquad \alpha_2 : a \lor \neg c \lor \neg f$ $\qquad \alpha_3 : \neg d \lor j \lor f$

$\alpha_4 : b \lor h$ $\qquad \alpha_5 : \neg c \lor \neg e \lor i$ $\qquad \alpha_6 : \neg i \lor \neg j \lor \neg g$

$\alpha_7 : e \lor \neg k$ $\qquad \alpha_8 : e \lor \neg h \lor k$ $\qquad \alpha_9 : \neg c \lor \neg e \lor \neg i \lor g$

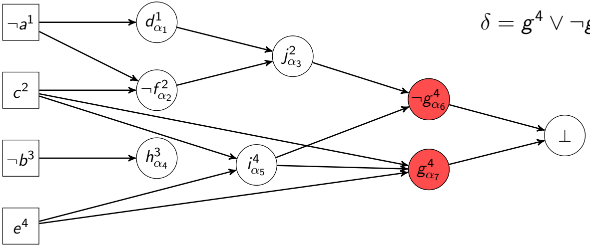


Conflict graph analysis

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

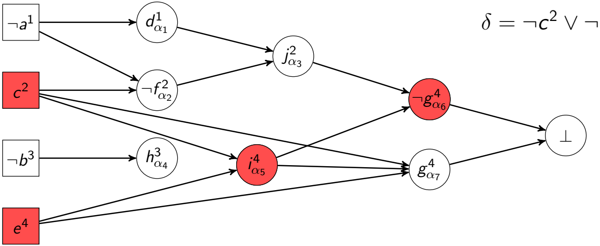
$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



$\delta = g^4 \vee \neg g^4$

Conflict graph analysis

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



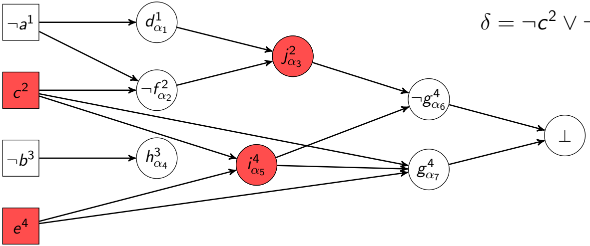
$$\delta = \neg c^2 \vee \neg e^4 \vee \neg i^4 \vee g^4$$

Conflict graph analysis

$\alpha_1 : a \vee d \qquad \alpha_2 : a \vee \neg c \vee \neg f \qquad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h \qquad \alpha_5 : \neg c \vee \neg e \vee i \qquad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k \qquad \alpha_8 : e \vee \neg h \vee k \qquad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



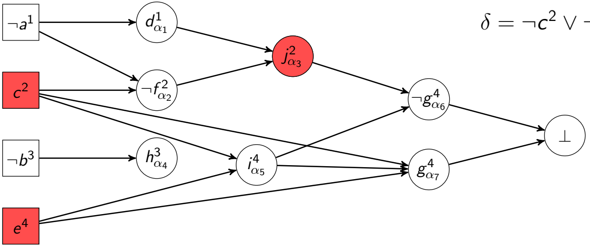
$\delta = \neg c^2 \vee \neg j^2 \vee \neg e^4 \vee \neg i^4$

Conflict graph analysis

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

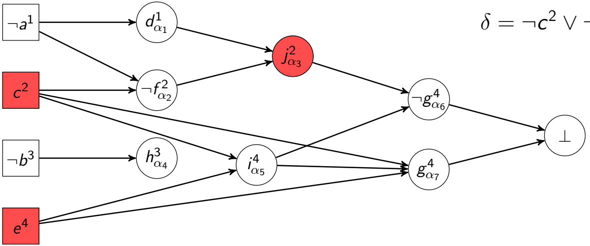
$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



$\delta = \neg c^2 \vee \neg j^2 \vee \neg e^4$

Conflict graph analysis

$\alpha_1 : a \vee d$	$\alpha_2 : a \vee \neg c \vee \neg f$	$\alpha_3 : \neg d \vee j \vee f$
$\alpha_4 : b \vee h$	$\alpha_5 : \neg c \vee \neg e \vee i$	$\alpha_6 : \neg i \vee \neg j \vee \neg g$
$\alpha_7 : e \vee \neg k$	$\alpha_8 : e \vee \neg h \vee k$	$\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



$\delta = \neg c^2 \vee \neg j^2 \vee \neg e^4$

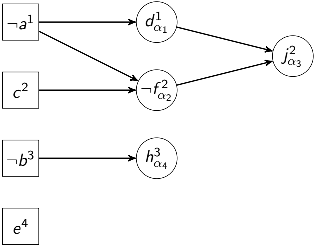
- Stops as soon as the resolvant has a **unique literal from the last decision level** (FUIP)
- δ is added to the clauses databases (ensure a systematic search)

Back-jumping

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



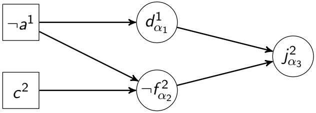
$$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^4$$

Back-jumping

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



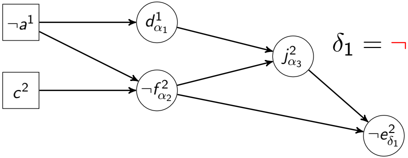
$$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e$$

Back-jumping

$\alpha_1 : a \vee d \quad \alpha_2 : a \vee \neg c \vee \neg f \quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h \quad \alpha_5 : \neg c \vee \neg e \vee i \quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k \quad \alpha_8 : e \vee \neg h \vee k \quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



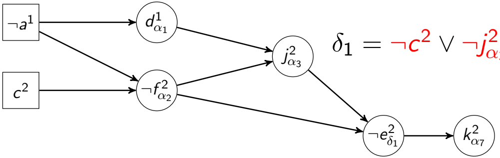
$$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^2$$

Back-jumping

$\alpha_1 : a \vee d \qquad \alpha_2 : a \vee \neg c \vee \neg f \qquad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h \qquad \alpha_5 : \neg c \vee \neg e \vee i \qquad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k \qquad \alpha_8 : e \vee \neg h \vee k \qquad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



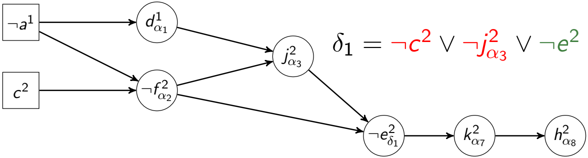
$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^2$

Back-jumping

$\alpha_1 : a \vee d \quad \alpha_2 : a \vee \neg c \vee \neg f \quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h \quad \alpha_5 : \neg c \vee \neg e \vee i \quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k \quad \alpha_8 : e \vee \neg h \vee k \quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



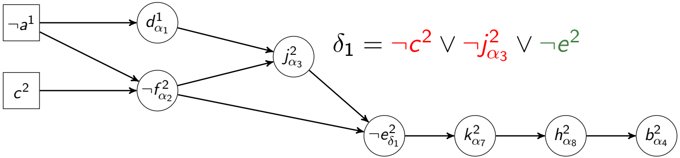
$$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^2$$

Back-jumping

$\alpha_1 : a \vee d \qquad \alpha_2 : a \vee \neg c \vee \neg f \qquad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h \qquad \alpha_5 : \neg c \vee \neg e \vee i \qquad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k \qquad \alpha_8 : e \vee \neg h \vee k \qquad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



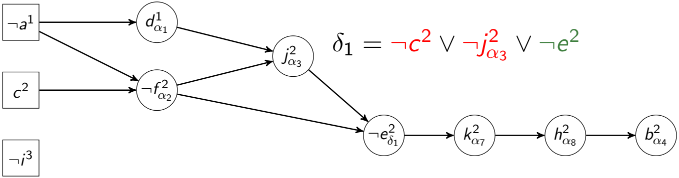
$$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^2$$

Back-jumping

$\alpha_1 : a \vee d$ $\quad$ $\alpha_2 : a \vee \neg c \vee \neg f$ $\quad$ $\alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad$ $\alpha_5 : \neg c \vee \neg e \vee i$ $\quad$ $\alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad$ $\alpha_8 : e \vee \neg h \vee k$ $\quad$ $\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



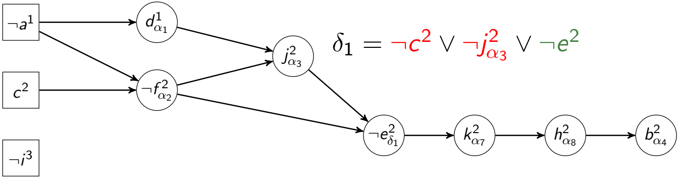
$$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^2$$

Back-jumping

$\alpha_1 : a \vee d$ $\alpha_2 : a \vee \neg c \vee \neg f$ $\alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\alpha_5 : \neg c \vee \neg e \vee i$ $\alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\alpha_8 : e \vee \neg h \vee k$ $\alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



SATISFIABILITY PROVED

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \quad \alpha_2 : \neg a \vee \neg c \vee \neg b \quad \alpha_3 : \neg a \vee c \vee \neg b$$

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \quad \alpha_2 : \neg a \vee \neg c \vee \neg b \quad \alpha_3 : \neg a \vee c \vee \neg b$$

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \quad \alpha_2 : \neg a \vee \neg c \vee \neg b \quad \alpha_3 : \neg a \vee c \vee \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$$\begin{array}{ccc} a : \{\} & b : \{\alpha_1\} & c : \{\alpha_3\} \\ \neg a : \{\alpha_1, \alpha_3\} & \neg b : \{\alpha_2\} & \neg c : \{\alpha_2\} \end{array}$$

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \quad \alpha_2 : \neg a \vee \neg c \vee \neg b \quad \alpha_3 : \neg a \vee c \vee \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$$a : \{\} \qquad b : \{\alpha_1\} \qquad c : \{\alpha_3\}$$
$$\neg a : \{\alpha_1, \alpha_3\} \qquad \neg b : \{\alpha_2\} \qquad \neg c : \{\alpha_2\}$$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \qquad \alpha_2 : \neg a \vee \neg c \vee \neg b \qquad \alpha_3 : \neg a \vee c \vee \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$$\begin{array}{ccc} a : \{\} & b : \{\alpha_1\} & c : \{\alpha_3\} \\ \neg a : \{\alpha_1, \alpha_3\} & \neg b : \{\alpha_2\} & \neg c : \{\alpha_2\} \end{array}$$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \lor b \lor c \quad \alpha_2 : \neg a \lor \neg c \lor \neg b \quad \alpha_3 : \neg a \lor c \lor \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$$\begin{array}{ccc} a : \{\} & b : \{\alpha_1\} & c : \{\alpha_3\} \\ \neg a : \{\alpha_1, \alpha_3\} & \neg b : \{\alpha_2\} & \neg c : \{\alpha_2\} \end{array}$$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \quad \alpha_2 : \neg a \vee \neg c \vee \neg b \quad \alpha_3 : \neg a \vee c \vee \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$a : \{\}$	$b : \{\alpha_1\}$	$c : \{\alpha_3\}$
$\neg a : \{\alpha_1, \alpha_3\}$	$\neg b : \{\alpha_2\}$	$\neg c : \{\alpha_2\}$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \qquad \alpha_2 : \neg a \vee \neg c \vee \neg b \qquad \alpha_3 : \neg a \vee c \vee \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$$a : \{\} \qquad b : \{\alpha_1\} \qquad c : \{\alpha_3, \alpha_1\}$$
$$\neg a : \{\alpha_1, \alpha_3\} \qquad \neg b : \{\alpha_2\} \qquad \neg c : \{\alpha_2\}$$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \qquad \alpha_2 : \neg a \vee \neg c \vee \neg b \qquad \alpha_3 : \neg a \vee c \vee \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$$a : \{\} \qquad b : \{\alpha_1\} \qquad c : \{\alpha_3, \alpha_1\}$$
$$\neg a : \{\alpha_1, \alpha_3\} \qquad \neg b : \{\alpha_2\} \qquad \neg c : \{\alpha_2\}$$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \qquad \alpha_2 : \neg a \vee \neg c \vee \neg b \qquad \alpha_3 : \neg a \vee c \vee \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$$a : \{\} \qquad b : \{\alpha_1\} \qquad c : \{\alpha_3, \alpha_1\}$$
$$\neg a : \{\alpha_1, \alpha_3\} \qquad \neg b : \{\alpha_2\} \qquad \neg c : \{\alpha_2\}$$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \lor b \lor c \qquad \alpha_2 : \neg a \lor \neg c \lor \neg b \qquad \alpha_3 : \neg a \lor c \lor \neg b$$

- **Mapping** between sentinel **literals** and the **clauses** they watch

$a : \{\}$	$b : \{\alpha_1\}$	$c : \{\alpha_3, \alpha_1\}$
$\neg a : \{\alpha_1, \alpha_3\}$	$\neg b : \{\alpha_2, \alpha_3\}$	$\neg c : \{\alpha_2\}$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Watched Literals

- Unit propagation **fires** when **all but one literal** is assigned **false**
- Idea: If two variables are either unassigned or one is assigned true, no need to do anything
- So just find two variables which satisfy this condition

$$\alpha_1 : \neg a \vee b \vee c \qquad \alpha_2 : \neg a \vee \neg c \vee \neg b \qquad \alpha_3 : \neg a \vee c \vee \neg b$$

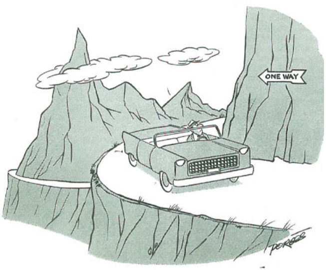
- **Mapping** between sentinel **literals** and the **clauses** they watch

$$\begin{array}{ccc} a : \{\} & b : \{\alpha_1\} & c : \{\alpha_3, \alpha_1\} \\ \neg a : \{\} & \neg b : \{\alpha_2, \alpha_3\} & \neg c : \{\alpha_2\} \end{array}$$

- When a literal x **is propagated** to true it is enough to consider the **clauses observed by** $\neg x$ and search another watched literal

Let us suppose that a is assigned to true

Heavy-Tailed Phenomena



- Depth-first search procedures often exhibit a remarkable **variability in the time** required to solve any problem instance
- Heavy-tailed behavior arises from the fact that **wrong branching decisions** may lead to explore an **exponentially large subtree** that contains no solutions
- Restarts provide good mechanisms to avoid such an issue

Restarts

- Often it a good strategy to abandon what you do and restart
  - for satisfiable instances the solver may get stuck in the unsatisfiable part
  - for unsatisfiable instances focusing on one part might miss short proofs
  - $\Rightarrow$ restart after the number of conflicts reached a restart limit
- Avoid to run into the same dead end
  - by randomization (either on the decision variable or its phase)
  - and/or just keep all the learned clauses
- For completeness dynamically increase restart limit
  - arithmetically, geometrically, Luby, Inner/Outer, Glucose restart

Reducing learned clauses

- CDCL SAT solvers learn clauses at each conflict
- Keeping all these clauses can slow down the unit propagation process

Reducing learned clauses

- CDCL SAT solvers learn clauses at each conflict
- Keeping all these clauses can slow down the unit propagation process

- "Useless" learned clauses are periodically deleted $(t_0, t_1 \ldots t_k, \ldots)$

α_1	α_2	α_3	α_4	α_5	$\ldots$ $\ldots$	α_k	α_n

Reducing learned clauses

- CDCL SAT solvers learn clauses at each conflict
- Keeping all these clauses can slow down the unit propagation process

- "Useless" learned clauses are periodically deleted $(t_0, t_1 \ldots t_k, \ldots)$

α_k	α_5	α_2	α_1	α_n	$\ldots \qquad \ldots$	α_3	α_4

Reducing learned clauses

- CDCL SAT solvers learn clauses at each conflict
- Keeping all these clauses can slow down the unit propagation process

- "Useless" learned clauses are periodically deleted $(t_0, t_1 \dots t_k, \dots)$

α_k	α_5	α_2	α_1	α_n	$\dots$	$\dots$	α_3	α_4

Reducing learned clauses

- CDCL SAT solvers learn clauses at each conflict
- Keeping all these clauses can slow down the unit propagation process
- "Useless" learned clauses are periodically deleted ($t_0, t_1 \ldots t_k, \ldots$)

α_k	α_5	α_2	α_1	α_n

- Deleting too much clauses make the learning process useless

Reducing learned clauses

- CDCL SAT solvers learn clauses at each conflict
- Keeping all these clauses can slow down the unit propagation process

- "Useless" learned clauses are periodically deleted $(t_0, t_1 \ldots t_k, \ldots)$

$$\boxed{\alpha_k}\boxed{\alpha_5}\boxed{\alpha_2}\boxed{\alpha_1}\boxed{\alpha_n}$$

- Deleting too much clauses make the learning process useless

- However, identify if a clause will be useful in the future is a hard task!

Estimate the clauses' utility

- The VSIDS measure
  - Keeping clauses that are often – and recently – used in the conflict analysis process
  - Dynamic measure
  - A clause useful in the past will be useful again in the future!
- The LBD measure
  - Represent the number of decision-levels in the learned clause
  - Static measure
  - Keeping clauses with a small LBD
- The PSM measure
  - Represent the number of literals assigned to false by *Progress Saving* interpretation
  - Static measure
  - Keeping clauses with a small PSM

CDCL algorithm

```
Input: a CNF formula Σ
Output: SAT or UNSAT
Δ = ∅ // learnt clauses database
while (true) do
    if (!propagate()) then
        if ((c = analyzeConflict()) == ∅) then returnUNSAT ;
        Δ = Δ ∪ {c};
        if (timeToRestart() then backtrack to level 0;
        else
            backtrack to the assertion level of c;
    else
        ℓ = decide();
        if (ℓ == null) then return SAT ;
        assert ℓ in a new decision level;
        if (timeToReduce()) then clean(Δ);
```

CDCL algorithm

```
Input: a CNF formula Σ
Output: SAT or UNSAT
Δ = ∅ // learnt clauses database
while (true) do
    if (!propagate()) then
        if ((c = analyzeConflict()) == ∅) then  returnUNSAT ;
        Δ = Δ ∪ {c};
        if (timeToRestart() then  backtrack to level 0;
        else
            backtrack to the assertion level of c;
    else
        ℓ = decide();
        if (ℓ == null) then  return SAT ;
        assert ℓ in a new decision level;
        if (timeToReduce()) then clean(Δ);
```

Outline

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

UNSAT

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - → constructive
  - → **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - → constructive
  - → **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - → constructive
  - → **destructive**

Minimal Unsatisfiable Set (MUS)

	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

SAT

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - → constructive
  - → **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

UNSAT

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - → constructive
  - → **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		
$\neg x \vee \neg y$	$\neg x \vee \neg z$	

MUS!

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		
$\neg x \vee \neg y$	$\neg x \vee \neg z$	

MUS!

- The formula is **unsatisfiable**: why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

SAT Incremental

From SAT to Incremental SAT

Solving the SAT problem

- Modern SAT solvers are based on the CDCL paradigm
- Dynamic heuristics:
  - $\rightarrow$ VSIDS, polarity, cleaning learned clauses and restart

Solving incrementally SAT

- Successive calls of a SAT solver
- Keeping a lot of information between the different runs
  - $\rightarrow$ VSIDS, polarity, cleaning learned clauses and restart

From SAT to Incremental SAT

Solving the SAT problem

- Modern SAT solvers are based on the CDCL paradigm
- Dynamic heuristics:
  - → VSIDS, polarity, cleaning learned clauses and restart

Solving incrementally SAT

- Successive calls of a SAT solver
- Keeping a lot of information between the different runs
  - → VSIDS, polarity, cleaning learned clauses and restart
  - → **learned clauses**

From SAT to Incremental SAT

Solving the SAT problem

- Modern SAT solvers are based on the CDCL paradigm
- Dynamic heuristics:
  - $\rightarrow$ VSIDS, polarity, cleaning learned clauses and restart

Solving incrementally SAT

- Successive calls of a SAT solver
- Keeping a lot of information between the different runs
  - $\rightarrow$ VSIDS, polarity, cleaning learned clauses and restart
  - $\rightarrow$ **learned clauses**

Adding selectors

Selectors

$\neg s_1 \vee x \vee y \vee z$ $\neg s_2 \vee x \vee \neg y$ $\neg s_3 \vee x \vee \neg z$

$\neg s_4 \vee \neg x \vee y \vee z$ $\neg s_5 \vee x \vee w$ $\neg s_6 \vee w \vee z \vee \neg y$

$\neg s_7 \vee \neg x \vee \neg y$ $\neg s_8 \vee \neg x \vee \neg z$ $\neg s_9 \vee w \vee \neg x \vee \neg z$

- To activate/deactivate the i^{th} clause :
  - $\rightarrow$ assign a_i to **false** to **activate** the clause
  - $\rightarrow$ assign a_i to **true** to **deactivate** the clause
- Used to know which initial clauses are participating to the creation of each learned clause

$$\begin{array}{ccc} \neg s_1 \vee x \vee y \vee z & & \neg s_2 \vee x \vee \neg y \\ & \searrow \quad \swarrow & \\ & \neg s_1 \vee \neg s_2 \vee x \vee z & \end{array}$$

Selectors

$\neg s_1 \vee x \vee y \vee z$ $\neg s_2 \vee x \vee \neg y$ $\neg s_3 \vee x \vee \neg z$

$\neg s_4 \vee \neg x \vee y \vee z$ $\neg s_5 \vee x \vee w$ $\neg s_6 \vee w \vee z \vee \neg y$

$\neg s_7 \vee \neg x \vee \neg y$ $\neg s_8 \vee \neg x \vee \neg z$ $\neg s_9 \vee w \vee \neg x \vee \neg z$

- To activate/deactivate the i^{th} clause :
  - $\rightarrow$ assign a_i to **false** to **activate** the clause
  - $\rightarrow$ assign a_i to **true** to **deactivate** the clause
- Used to know which initial clauses are participating to the creation of each learned clause

$\neg s_1 \vee x \vee y \vee z$ $\neg s_2 \vee x \vee \neg y$

$\neg s_1 \vee \neg s_2 \vee x \vee z$

Selectors impact on the size of the clauses

Let's test

- 300 instances from the MUS competition 2011
- timeout = 2400 secondes
- memout = 7800 Mo

- MUSer as MUS extractor

  → defaults options
- SAT solvers: **GLUCOSE** versus **MINISAT**

- Intel XEON X5550 Quad-Core 2.66 GHz with 32Go of RAM

GLUCOSE VS. MINISAT

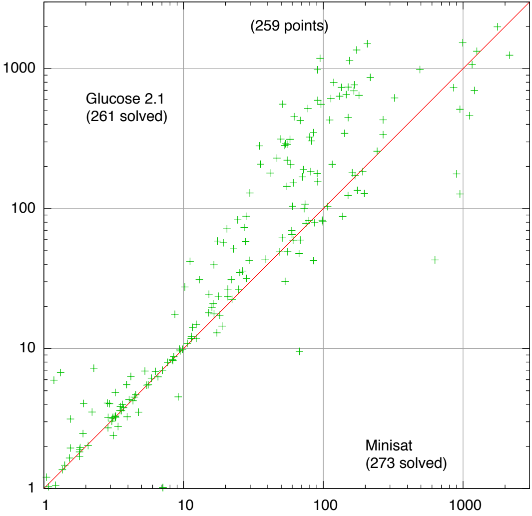


Figure: Solving time

GLUCOSE VS. MINISAT

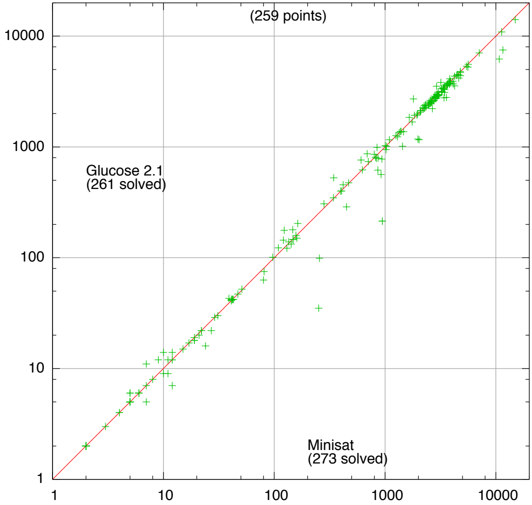


Figure: Number of SAT calls

Why GLUCOSE is so inefficient?

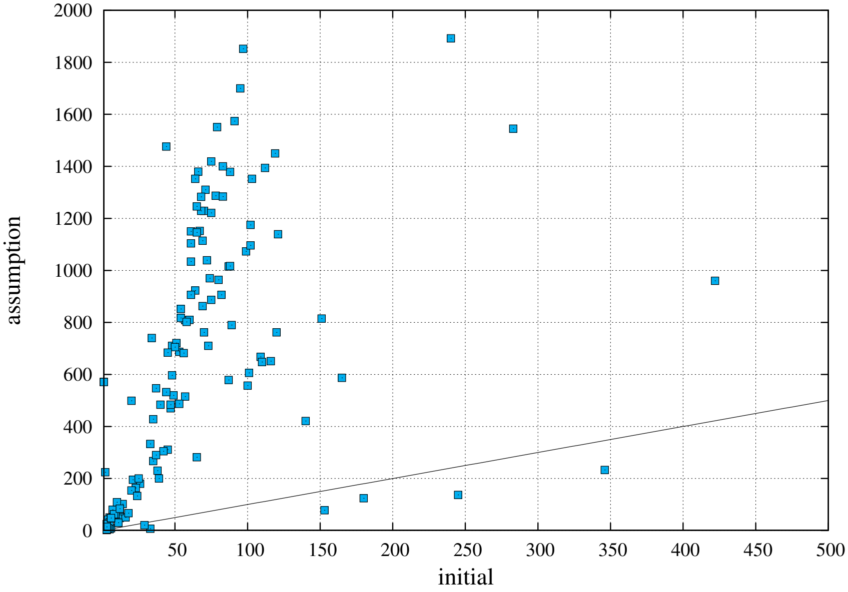
- **Main difference** between GLUCOSE et MINISAT

  → restart and cleaning strategy

- GLUCOSE is entirely **based on the LBD notion**

- Each selector has its own decision level

Why GLUCOSE is so inefficient?



Why GLUCOSE is so inefficient?

- **Main difference** between GLUCOSE et MINISAT
  → restart and cleaning strategy
- GLUCOSE is entirely **based on the LBD notion**
- Each selector has its own decision level
- Then the LBD reflects the number of selectors of a learned clause
- We have to redefine the notion of LBD: **improved LBD**

Why GLUCOSE is so inefficient?

- **Main difference** between GLUCOSE et MINISAT
  → restart and cleaning strategy
- GLUCOSE is entirely **based on the LBD notion**
- Each selector has its own decision level
- Then the LBD reflects the number of selectors of a learned clause
- We have to redefine the notion of LBD: **improved LBD**

Not considering selectors when computing the LBD

Updated LBD

		taille		LBD		Imp. LBD	
Instance	#C	moy.	max	moy.	max	moy.	max
fdmus_b21_96	8541	1145	5980	1095	5945	8	71
longmult6	8853	694	3104	672	3013	11	61
dump_vc950	360419	522	36309	498	35873	8	307
g7n	15110	1098	16338	1049	16268	27	160

- The new LBD looks more appropriate
- It really measures now how a clause is useful!

GLUCOSE Improved LBD vs. GLUCOSE de base

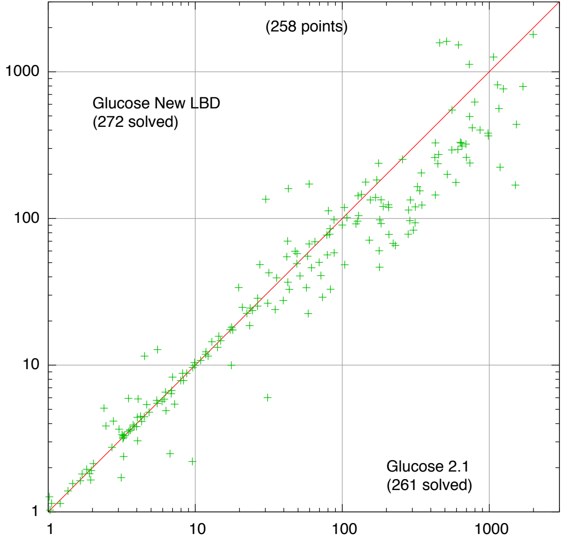


Figure: Solving time

GLUCOSE Improved LBD vs. MINISAT

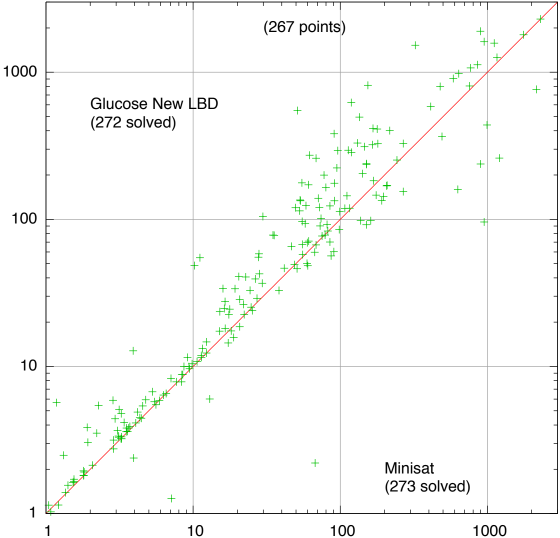


Figure: Solving time

The clauses are too long

The main procedures depend on the size of the clauses

- Conflict analysis
  - → put the initial literals at the front of the clause
- Unit propagation
  - → search for a initial literal or a literal satisfied
  - → push the selector to the end of the clause
- Simplification procedure
  - → only check for the watched literals
- LBD recalculation
  - → save the number of selectors
  - → stop once we are sure that only selectors remain

GLUCOSEInc vs. GLUCOSE de base

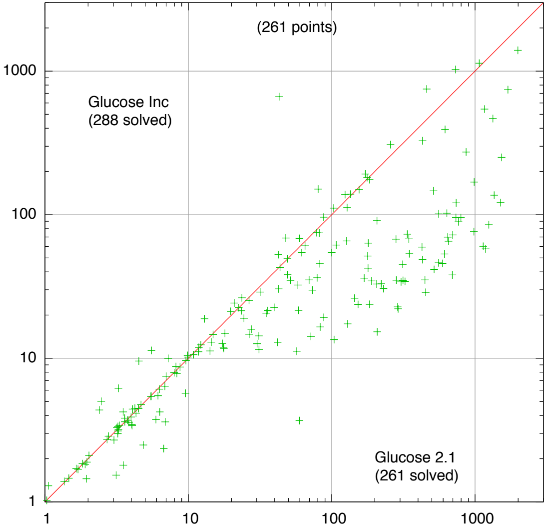


Figure: Solving time

GLUCOSEInc VS. MINISAT

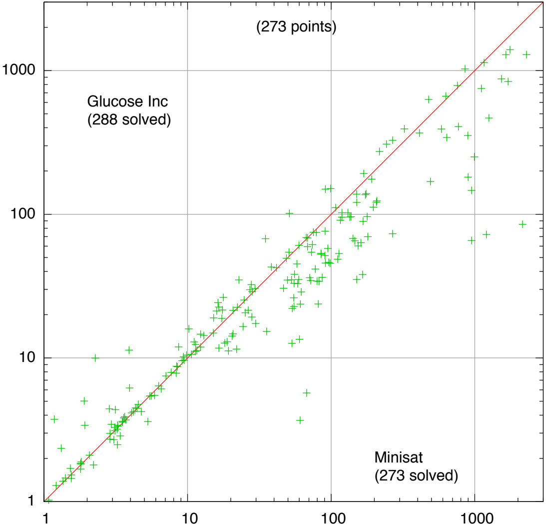


Figure: Solving time

Comparisons

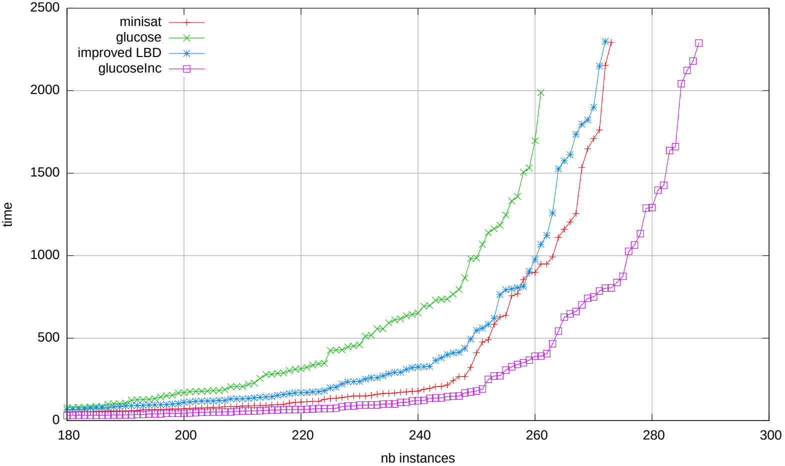


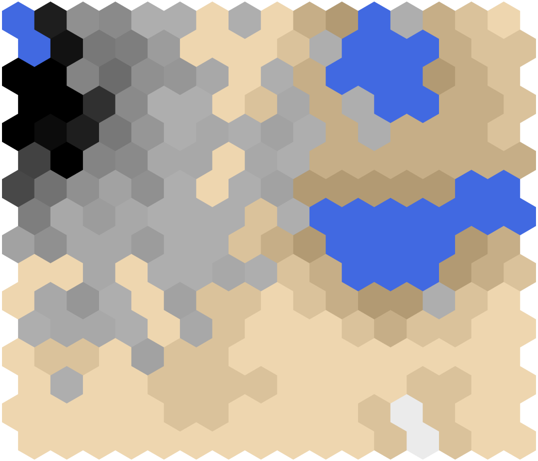
Figure: Number of instances solved regarding the time

Outline

Team Formation Problem

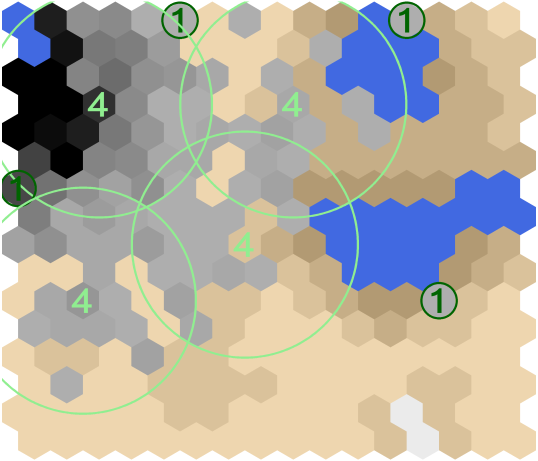
- Team formation (TF) is the problem of deploying the least expensive team of agents while covering a set of skills
- A *TF problem description* is a tuple $\langle A, S, f, \alpha \rangle$
    - where $A = \{a_1, \ldots, a_n\}$ is a set of agents,
    - $S = \{s_1, \ldots, s_m\}$ is a set of skills,
    - $f : A \mapsto \mathbb{N}$ is a deployment cost function,
    - and $\alpha : A \mapsto 2^S$ is an agent-to-skill function.
- A **team** is a subset of agents $T \subseteq A$
- One extends the cost function f to teams T as $f(T) = \sum_{a_i \in T} f(a_i)$
- The agent-to-skill function α is extended to teams T as $\alpha(T) = \bigcup_{a_i \in T} \alpha(a_i)$
- A team $T \subseteq A$ is **efficient** if all skills from S are covered by T, i.e., when $\alpha(T) = S$ (it is almost equivalent to the Set Cover problem)
- An optimal team is an efficient team minimizing the cost function
- The corresponding decision problem asks, given a bound $b \in \mathbb{N}$ as input, whether there exists an efficient team T such that $f(T) \leq b$

Example



- An agent_i has a deployment cost equal to i and a cover range equal to $i - 1$

Example



- An $agent_i$ has a deployment cost equal to i and a cover range equal to $i - 1$

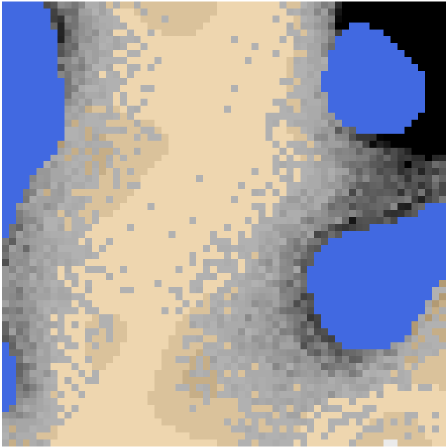
SAT encoding

- We associate a boolean variable p_i with each agent in $a_i \in A$, where p_i is true if and only if the agent a_i is present in the deployed team.
- Given a TF instance $\langle A, S, f, \alpha \rangle$ and $b \in \mathbb{N}$, a minimal efficient team T exists if and only if:

$$\bigwedge_{s_j \in S} \bigvee_{a_i \in A | s_j \in \alpha(i)} p_i \tag{1}$$

$$\sum_{a_i \in A} f(i) \times p_i \leq b \tag{2}$$

And in practice ...



- 6 types of agent by cell with cover ranging from 1 to 6
- $b = 200$

And in practice ...

- CaDiCaL:

```
...
c total process time since initialization:      1799.17    seconds
c total real time since initialization:         1800.08    seconds
c maximum resident set size of process:         2577.50    MB
c
c raising signal 2 (SIGINT)
```

And in practice ...

- CaDiCaL:

```
...
c total process time since initialization:     1799.17    seconds
c total real time since initialization:        1800.08    seconds
c maximum resident set size of process:        2577.50    MB
c
c raising signal 2 (SIGINT)
```

- LMHS:

```
...
o 230
s OPTIMUM FOUND
...
c CPU time: 373144 ms
c Real time: 373256 ms
```

Outline

Take away message

- It is better to know how SAT solvers work to avoid some pitfalls
- Sometimes the problem is the size of the encoding, then try to encode your problem incrementally
- SAT solvers are not always efficient when it comes to use cardinality and/or pseudo boolean constraints