Deep Dive into CDCL Pseudo-Boolean Solvers

Romain Wallon
May 20th, 2021

Laboratoire d’Informatique de l’X (LIX), École Polytechnique
Context
The satisfiability problem (SAT) is the first problem proven to be NP-complete (Cook, 1971).

Given a CNF formula $\Sigma$, this problem is determining whether there exists an assignment of the (Boolean) variables of $\Sigma$ such that this formula evaluates to true.
The **satisfiability problem** (SAT) is the first problem proven to be **NP-complete** (Cook, 1971)

Given a **CNF formula** $\Sigma$, this problem is determining whether there exists an assignment of the (Boolean) variables of $\Sigma$ such that this formula evaluates to true

$$(a \lor b \lor \overline{c}) \land (a \lor \overline{b} \lor c) \land (\overline{a} \lor \overline{b} \lor c)$$
Boolean Satisfiability

The satisfiability problem (SAT) is the first problem proven to be NP-complete (Cook, 1971)

Given a CNF formula $\Sigma$, this problem is determining whether there exists an assignment of the (Boolean) variables of $\Sigma$ such that this formula evaluates to true

$$(a \lor b \lor \bar{c}) \land (a \lor \bar{b} \lor c) \land (\bar{a} \lor \bar{b} \lor c)$$
The satisfiability problem (SAT) is the first problem proven to be NP-complete (Cook, 1971)

Given a CNF formula $\Sigma$, this problem is determining whether there exists an assignment of the (Boolean) variables of $\Sigma$ such that this formula evaluates to true

$$(a \lor b \lor \bar{c}) \land (a \lor \bar{b} \lor c) \land (\bar{a} \lor b \lor c)$$

In the early 2000s, a revolution in the architecture of SAT solvers happened, with the wide adoption of the CDCL approach (GRASP) and the use of efficient heuristics and data structures (Chaff, MiniSat)
Boolean Satisfiability

The satisfiability problem (SAT) is the first problem proven to be NP-complete (Cook, 1971)

Given a CNF formula \( \Sigma \), this problem is determining whether there exists an assignment of the (Boolean) variables of \( \Sigma \) such that this formula evaluates to true

\[
(a \lor b \lor \bar{c}) \land (a \lor \bar{b} \lor c) \land (\bar{a} \lor \bar{b} \lor c)
\]

In the early 2000s, a revolution in the architecture of SAT solvers happened, with the wide adoption of the CDCL approach (GRASP) and the use of efficient heuristics and data structures (Chaff, MiniSat)

Modern SAT solvers (Glucose, CaDiCaL, Kissat) can now deal with problems containing millions of variables and clauses
So-called “modern” SAT solvers are very efficient in practice, but some instances remain **completely out of reach** for these solvers, due to the weakness of the **resolution proof system** they use internally.
So-called “modern” SAT solvers are very efficient in practice, but some instances remain completely out of reach for these solvers, due to the weakness of the resolution proof system they use internally.

This is particularly for instances requiring the ability to count, such as pigeonhole-principle formulae, stating that “n pigeons do not fit in n – 1 holes.”
So-called “modern” SAT solvers are very efficient in practice, but some instances remain completely out of reach for these solvers, due to the weakness of the resolution proof system they use internally.

This is particularly for instances requiring the ability to count, such as pigeonhole-principle formulae, stating that “n pigeons do not fit in n – 1 holes”.

While modern SAT solvers perform poorly on such instances for \( n > 20 \), PB solvers based on cutting-planes may solve them in linear time.
We consider conjunctions of linear equations or inequations over Boolean variables of the form:

\[ \sum_{i=1}^{n} \alpha_i \ell_i \triangleq \delta \]

in which

- the **coefficients** \( \alpha_i \) are integers
- \( \ell_i \) are **literals**, i.e., a variable \( v \) or its negation \( \overline{v} = 1 - v \)
- \( \triangleq \) is a **relational operator** among \( \{<, \leq, =, \geq, >\} \)
- the **degree** \( \delta \) is an integer
Pseudo-Boolean (PB) Constraints

We consider conjunctions of linear equations or inequations over Boolean variables of the form:

$$\sum_{i=1}^{n} \alpha_i \ell_i \triangle \delta$$

in which

- the coefficients $\alpha_i$ are integers
- $\ell_i$ are literals, i.e., a variable $v$ or its negation $\bar{v} = 1 - v$
- $\triangle$ is a relational operator among $\{<, \leq, =, \geq, >\}$
- the degree $\delta$ is an integer

For example:

$$-3a + 4b - 7c + d \leq -5$$
Without loss of generality, we consider conjunctions of normalized PB constraints of the form:

\[ \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \]

in which

- the coefficients \(\alpha_i\) are non-negative integers
- \(\ell_i\) are literals
- the degree \(\delta\) is a non-negative integer
Without loss of generality, we consider conjunctions of normalized PB constraints of the form:

\[ \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \]

in which

- the coefficients \( \alpha_i \) are non-negative integers
- \( \ell_i \) are literals
- the degree \( \delta \) is a non-negative integer

For example:

\[-3a + 4b - 7c + d \leq -5 \equiv 3a + 4\bar{b} + 7c + \bar{d} \geq 10\]
The CDCL Architecture

Figure 1: Overview of the CDCL algorithm
Figure 2: Use of the proof system in the CDCL algorithm
Fitting Cutting-Planes into the CDCL Architecture
The constraint $3a + 4\bar{b} + 7c + \bar{d} \geq 10$ propagates $c$ to true
The constraint $3a + 4\overline{b} + 7c + \overline{d} \geq 10$ propagates $c$ to true.

- A PB constraint can propagate truth values \textit{without any assignment}.
The constraint $3a + 4\bar{b} + 7c + \bar{d} \geq 10$ propagates $c$ to true

- A PB constraint can propagate truth values *without any assignment*

- A PB constraint can propagate *multiple* truth values at *different* decision levels
Propagations in PB Constraints

The constraint $3a + 4\overline{b} + 7c + \overline{d} \geq 10$ propagates $c$ to true

- A PB constraint can propagate truth values without any assignment
- A PB constraint can propagate multiple truth values at different decision levels

*The constraint above can be rewritten as $c \land 3a + 4\overline{b} + \overline{d} \geq 3$
*but also as $c \land (a \lor \overline{b})$*
Many PB solvers have been designed based on the Generalized Resolution (Hooker, 1988).

\[
\alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta_1 \quad \beta \bar{l} + \sum_{i=1}^{n} \beta_i l_i \geq \delta_2 \quad \text{(cancellation)}
\]

\[
\sum_{i=1}^{n} (\beta \alpha_i + \alpha \beta_i) l_i \geq \beta \delta_1 + \alpha \delta_2 - \alpha \beta
\]

\[
\sum_{i=1}^{n} \alpha_i l_i \geq \delta \quad \sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta \quad \text{(saturation)}
\]
Many PB solvers have been designed based on the Generalized Resolution (Hooker, 1988).

\[
\begin{align*}
\alpha l + \sum_{i=1}^{n} \alpha_i l_i & \geq \delta_1 \\
\beta \bar{\ell} + \sum_{i=1}^{n} \beta_i l_i & \geq \delta_2 \\
\sum_{i=1}^{n} (\beta \alpha_i + \alpha \beta_i) l_i & \geq \beta \delta_1 + \alpha \delta_2 - \alpha \beta \\
\sum_{i=1}^{n} \alpha_i l_i & \geq \delta \\
\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i & \geq \delta
\end{align*}
\]

(cancellation) (saturation)
Many PB solvers have been designed based on the Generalized Resolution (Hooker, 1988).

\[
\alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta_1 \\
\beta \bar{l} + \sum_{i=1}^{n} \beta_i l_i \geq \delta_2 \\
\sum_{i=1}^{n} (\beta \alpha_i + \alpha \beta_i) l_i \geq \beta \delta_1 + \alpha \delta_2 - \alpha \beta \\
\sum_{i=1}^{n} \alpha_i l_i \geq \delta \\
\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta
\]  

(cancellation)

(saturation)
Many PB solvers have been designed based on the Generalized Resolution (Hooker, 1988).

\[
\alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta_1 \quad \frac{\beta \bar{l} + \sum_{i=1}^{n} \beta_i l_i \geq \delta_2}{\sum_{i=1}^{n} (\beta \alpha_i + \alpha \beta_i) l_i \geq \beta \delta_1 + \alpha \delta_2 - \alpha \beta} \quad \text{(cancellation)}
\]

\[
\sum_{i=1}^{n} \alpha_i l_i \geq \delta \quad \frac{\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta} \quad \text{(saturation)}
\]
Many PB solvers have been designed based on the Generalized Resolution (Hooker, 1988).

\[
\alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta_1 \quad \beta \bar{l} + \sum_{i=1}^{n} \beta_i l_i \geq \delta_2 \quad \text{(cancellation)}
\]

\[
\sum_{i=1}^{n} (\beta \alpha_i + \alpha \beta_i) l_i \geq \beta \delta_1 + \alpha \delta_2 - \alpha \beta
\]

\[
\sum_{i=1}^{n} \alpha_i l_i \geq \delta \quad \sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta \quad \text{(saturation)}
\]
Many PB solvers have been designed based on the **Generalized Resolution** (Hooker, 1988).

\[
\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \quad \beta \bar{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2
\]

(cancellation)

\[
\sum_{i=1}^{n} (\beta \alpha_i + \alpha \beta_i) \ell_i \geq \beta \delta_1 + \alpha \delta_2 - \alpha \beta
\]

As with the resolution rule in classical SAT solvers, these two rules can be used to **learn new constraints** during conflict analysis.
Suppose that we have the following constraints:

\[6\bar{b} + 6c + 4e + f + g + h \geq 7\]

This conflict is analyzed by applying the cancellation rule as follows:

\[6\bar{b} + 6c + 4e + f + g + h \geq 7\]

The constraint we obtain here is no longer conflicting!
Suppose that we have the following constraints:

\[6\bar{b} + 6c + 4e + f + g + h \geq 7\]

\[5a + 4b + c + d \geq 6\]
Analyzing Conflicts

Suppose that we have the following constraints:

\[ 6\bar{b} + 6c + 4e + f + g + h \geq 7 \]

\[ 5a + 4b + c + d \geq 6 \]
Suppose that we have the following constraints:

\[ 6\bar{b} + 6c + 4e + f + g + h \geq 7 \]

\[ 5a + 4b + c + d \geq 6 \]
Suppose that we have the following constraints:

\[ 6\bar{b} + 6c + 4e + f + g + h \geq 7 \]
\[ 5a + 4b + c + d \geq 6 \]

(reason for \( \bar{b} \))
Analyzing Conflicts

Suppose that we have the following constraints:

\[ 6\bar{b} + 6c + 4e + f + g + h \geq 7 \]

(reason for \( \bar{b} \))

\[ 5a + 4b + c + d \geq 6 \]

(conflict)

This conflict is analyzed by applying the cancellation rule as follows:

\[ 6\bar{b} + 6c + 4e + f + g + h \geq 7 \]

\[ 5a + 4b + c + d \geq 6 \]

The constraint we obtain here is no longer conflicting!
Analyzing Conflicts

Suppose that we have the following constraints:

\[6\overline{b} + 6c + 4e + f + g + h \geq 7\quad (\text{reason for } \overline{b})\]

\[5a + 4b + c + d \geq 6\quad (\text{conflict})\]

This conflict is analyzed by applying the cancellation rule as follows:

\[
\begin{align*}
6\overline{b} + 6c + 4e + f + g + h & \geq 7 \\
15a + 15c + 8e + 3d + 2f + 2g + 2h & \geq 20
\end{align*}
\]
Suppose that we have the following constraints:

\[ 6\bar{b} + 6c + 4e + f + g + h \geq 7 \]
\[ 5a + 4b + c + d \geq 6 \]

(reason for \(\bar{b}\)) (conflict)

This conflict is analyzed by applying the cancellation rule as follows:

\[ \frac{6\bar{b} + 6c + 4e + f + g + h \geq 7}{15a + 15c + 8e + 3d + 2f + 2g + 2h \geq 20} \]
\[ 5a + 4b + c + d \geq 6 \]

The constraint we obtain here is no longer conflicting!
To preserve the conflict, the \textit{weakening} rule must be used:

$$\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha} \quad \text{(weakening)}$$

$$\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad \text{for} \quad k \in \mathbb{N} \quad 0 < k \leq \alpha \quad \text{(partial weakening)}$$
To preserve the conflict, the *weakening* rule must be used:

\[
\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \\
\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha
\]

(weakening)

\[
\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d \\
(\alpha - k)\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k
\]

(partial weakening)
To preserve the conflict, the weakening rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i}{\sum_{i=1}^{n} \alpha_i \ell_i} \geq \delta - \alpha \quad \text{(weakening)}
\]

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i} \geq \delta - k \quad k \in \mathbb{N} \quad 0 < k \leq \alpha \quad \text{(partial weakening)}
\]
To preserve the conflict, the *weakening* rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha}
\] (weakening)

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k)\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k}
\] (partial weakening)

\[5a + 5b + 3c \geq 8\]
Weakening

To preserve the conflict, the weakening rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha} \quad (\text{weakening})
\]

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad (\text{partial weakening})
\]

\[5a + 5b + 3c \geq 8\]
Weakening

To preserve the conflict, the weakening rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha} \quad \text{(weakening)}
\]

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad \text{(partial weakening)}
\]

\[
5a + 5b \geq 8 - 3
\]
Weakening

To preserve the conflict, the weakening rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha}
\]

(weakening)

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k}
\]

(partial weakening)

\[
5a + 5b \geq 5
\]
To preserve the conflict, the weakening rule must be used:

\[
\frac{\alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta}{\sum_{i=1}^{n} \alpha_i l_i \geq \delta - \alpha} \quad \text{(weakening)}
\]

\[
\frac{\alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq d}{(\alpha - k)l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta - k} \quad \text{(partial weakening)}
\]
Weakening

To preserve the conflict, the \textit{weakening} rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha} \quad \text{(weakening)}
\]

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k)\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad \text{(partial weakening)}
\]
Weakening

To preserve the conflict, the *weakening* rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha}
\]  
(weakening)

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k}
\]  
(partial weakening)

\[
5a + 5b + 3c \geq 8
\]
Weakening

To preserve the conflict, the \textit{weakening} rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha} \quad \text{(weakening)}
\]

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k)\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad \text{(partial weakening)}
\]

\[
5a + 5b + 3c \geq 8
\]
To preserve the conflict, the \textit{weakening} rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i}{\sum_{i=1}^{n} \alpha_i \ell_i} \geq \delta \quad \text{(weakening)}
\]

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i}{\sum_{i=1}^{n} \alpha_i \ell_i} \geq \delta - \alpha
\]

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i}{\sum_{i=1}^{n} \alpha_i \ell_i} \geq \delta - k
\]

\[
5a + (5 - 2)b + 3c \geq 8 - 2
\]
To preserve the conflict, the **weakening** rule must be used:

\[
\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \quad \text{(weakening)}
\]

\[
\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha
\]

\[
\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d \quad k \in \mathbb{N}, \quad 0 < k \leq \alpha 
\]

\[
(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k \quad \text{(partial weakening)}
\]

\[
5a + 3b + 3c \geq 6
\]
To preserve the conflict, the **weakening** rule must be used:

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - \alpha} \quad \text{(weakening)}
\]

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad \text{(partial weakening)}
\]

\[5a + 3b + 3c \geq 6\]

*Weakening can be applied in many different ways (Le Berre et al., 2020b)*
Different Weakening Strategies

The original approach (Dixon and Ginsberg, 2002; Chai and Kuehlmann, 2003) successively weakens away literals from the reason, until the saturation rule guarantees to derive a conflicting constraint.
Different Weakening Strategies

The original approach (Dixon and Ginsberg, 2002; Chai and Kuehlmann, 2003) successively weakens away literals from the reason, until the saturation rule guarantees to derive a conflicting constraint.

As the operation must be repeated multiple times, its cost is not negligible.
Different Weakening Strategies

The original approach (Dixon and Ginsberg, 2002; Chai and Kuehlmann, 2003) successively weakens away literals from the reason, until the saturation rule guarantees to derive a conflicting constraint.

As the operation must be repeated multiple times, its cost is not negligible.

Another solution is to take advantage of the following property:

As soon as the coefficient of the literal to cancel is equal to 1 in at least one of the constraints, the derived constraint is guaranteed to be conflicting (Dixon, 2004).
Weakening Ineffective Literals

During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants.
Weakening Ineffective Literals

During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[
\begin{align*}
3\bar{a} + 3\bar{b} + c + d + e & \geq 6 \\
3\bar{b} + c & \geq 6 - 3 - 1 - 1 = 1 \\
\bar{b} + c & \geq 1
\end{align*}
\]
During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[
3\bar{a} + 3\bar{b} + c + d + e \geq 6
\]

\[
\frac{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}{\bar{b} + c \geq 1}
\]
During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[ 3\bar{a} + 3\bar{b} + c + d + e \geq 6 \]

\[ 3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1 \]

\[ \bar{b} + c \geq 1 \]
Weakening Ineffective Literals

During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[ 3\bar{a} + 3\bar{b} + c + \bar{d} + e \geq 6 \]

\[ 3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1 \]

\[ \bar{b} + c \geq 1 \]
During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[
\begin{align*}
3\bar{a} + 3\bar{b} + c + d + e & \geq 6 \\
3\bar{b} + c & \geq 6 - 3 - 1 - 1 = 1 \\
\bar{b} + c & \geq 1
\end{align*}
\]
Weakening Ineffective Literals

During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[
\begin{align*}
3\bar{a} + 3\bar{b} + c + d + e &\geq 6 \\
3\bar{b} + c &\geq 6 - 3 - 1 - 1 = 1 \\
\bar{b} + c &\geq 1
\end{align*}
\]

\[
\begin{align*}
2a + b + c + f &\geq 2 \\
2a + b + f &\geq 2 - 1 = 1 \\
a + b + f &\geq 1
\end{align*}
\]
During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[
\begin{align*}
3\bar{a} + 3\bar{b} + c + d + e & \geq 6 \\
\bar{b} + c & \geq 6 - 3 - 1 - 1 = 1 \\
\bar{b} + c & \geq 1
\end{align*}
\]

\[
\begin{align*}
2a + b + c + f & \geq 2 \\
2a + b + f & \geq 2 - 1 = 1 \\
a + b + f & \geq 1
\end{align*}
\]
Weakening Ineffective Literals

During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[
\begin{align*}
\boxed{3a} + 3\overline{b} + c + \boxed{d} + \boxed{e} & \geq 6 \\
3\overline{b} + c & \geq 6 - 3 - 1 - 1 = 1 \\
\overline{b} + c & \geq 1
\end{align*}
\]

\[
\begin{align*}
2a + b + \boxed{c} + f & \geq 2 \\
2a + b + f & \geq 2 - 1 = 1 \\
a + b + f & \geq 1
\end{align*}
\]
Weakening Ineffective Literals

During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[
\begin{align*}
3\bar{a} + 3\bar{b} + c + d + e & \geq 6 \\
3\bar{b} + c & \geq 6 - 3 - 1 - 1 = 1 \\
\bar{b} + c & \geq 1
\end{align*}
\]

\[
\begin{align*}
2a + b + c + f & \geq 2 \\
2a + b + f & \geq 2 - 1 = 1 \\
a + b + f & \geq 1
\end{align*}
\]
Weakening Ineffective Literals

During conflict analysis, some literals may not play a role in the conflict being analyzed: it is thus possible to weaken them away while preserving invariants

\[
\begin{align*}
3\overline{a} + 3\overline{b} + c + d + e & \geq 6 \\
3\overline{b} + c & \geq 6 - 3 - 1 - 1 = 1 \\
\overline{b} + c & \geq 1
\end{align*}
\]

\[
\begin{align*}
2a + b + c + f & \geq 2 \\
2a + b + f & \geq 2 - 1 = 1 \\
a + b + f & \geq 1
\end{align*}
\]

This strategy is equivalent to that used by solvers such as SATIRE (Whittemore and Sakallah, 2001) or Sat4j-Resolution to lazily infer clauses to use resolution based reasoning.
Weakening and Division

In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the **division rule**, applied after having weakened away some unfalsified literals

\[
\sum_{i=1}^{n} \alpha_i l_i \geq \delta \quad \rho \in \mathbb{N}^* \quad \text{(division)}
\]
In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the **division rule**, applied after having weakened away some unfalsified literals:

\[
\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \left\lceil \frac{\alpha_i}{\rho} \right\rceil \ell_i \geq \left\lceil \frac{\delta}{\rho} \right\rceil} \quad \rho \in \mathbb{N}^* \quad \text{(division)}
\]
Weakening and Division

In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the division rule, applied after having weakened away some unfalsified literals

\[
\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \quad \rho \in \mathbb{N}^* \\
\sum_{i=1}^{n} \left\lceil \frac{\alpha_i}{\rho} \right\rceil \ell_i \geq \left\lceil \frac{\delta}{\rho} \right\rceil
\]

(division)
Weakening and Division

In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the *division rule*, applied after having weakened away some unfalsified literals

\[
\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \quad \rho \in \mathbb{N}^* \quad \text{(division)}
\]

\[
\begin{align*}
8a + 7b + 7c + 2d + 2e + f & \geq 11 \\
7b + 7c + 2d + 2e & \geq 2 \\
b + c + d + e & \geq 1
\end{align*}
\]
Weakening and Division

In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the division rule, applied after having weakened away some unfalsified literals

\[
\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil \ell_i \geq \lceil \frac{\delta}{\rho} \rceil} \quad (\text{division})
\]

\[
8a + 7b + 7c + 2d + 2e + f \geq 11
\]

\[
\frac{7b + 7c + 2d + 2e \geq 2}{b + c + d + e \geq 1}
\]
In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the **division rule**, applied after having weakened away some unfalsified literals

\[
\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \quad \rho \in \mathbb{N}^* \quad \text{(division)}
\]

\[
\frac{8a + 7b + 7c + 2d + 2e + f \geq 11}{7b + 7c + 2d + 2e \geq 2}
\]

\[
b + c + d + e \geq 1
\]
Weakening and Division

In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the **division rule**, applied after having weakened away some unfalsified literals

\[
\sum_{i=1}^{n} \alpha_i l_i \geq \delta \quad \rho \in \mathbb{N}^* \quad \text{(division)}
\]

\[
\begin{align*}
8a + 7b + 7c + 2d + 2e + f & \geq 11 \\
7b + 7c + 2d + 2e & \geq 2 \\
b + c + d + e & \geq 1
\end{align*}
\]
Weakening and Division

In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the **division rule**, applied after having weakened away some unfalsified literals

\[
\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \quad \rho \in \mathbb{N}^* \\
\left( \text{division} \right)
\]

\[
8a + 7b + 7c + 2d + 2e + f \geq 11 \\
7b + 7c + 2d + 2e \geq 2 \\
b + c + d + e \geq 1
\]

It is also possible to apply **partial weakening** before division to infer stronger constraints
Many Different Strategies

![Graph showing the performance of various strategies against the number of non-easy instances and time in seconds. The graph compares strategies like VBS, Sat4j-PartialRoundingSat (both), Sat4j-RoundingSat (both), Weaken Ineffective (both), Weaken Ineffective (conflict), Sat4j-PartialRoundingSat (conflict), Sat4j-RoundingSat (conflict), Weaken Ineffective (reason), Sat4j-RoundingSat (reason), Sat4j-PartialRoundingSat (reason), Multiply and Weaken, and Sat4j-GeneralizedResolution. Each strategy is represented by a different line color or style on the graph.]
An Achilles Heel in the Cutting Planes Proof System
Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

\[
3d + a + b + c \geq 3 \quad \quad 3\bar{d} + 2a + 2b \geq 3
\]

\[
3a + 3b + c \geq 3
\]
Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

\[
\begin{align*}
3d + a + b + c & \geq 3 \\
3\bar{d} + 2a + 2b & \geq 3 \\
3a + 3b + c & \geq 3
\end{align*}
\]
Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

\[
\begin{align*}
3d + a + b + c &\geq 3 \\
\frac{3d + 2a + 2b}{3} &\geq 3 \\
3a + 3b + c &\geq 3
\end{align*}
\]
Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

\[
\begin{align*}
3d + a + b + c & \geq 3 \\
3\bar{d} + 2a + 2b & \geq 3 \\
3a + 3b + c & \geq 3
\end{align*}
\]

A literal is said to be irrelevant in a PB constraint when its truth value does not impact the truth value of the constraint: irrelevant literals can thus be removed.
Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

\[
\begin{align*}
3d + a + b + c & \geq 3 \\
3d + 2a + 2b & \geq 3 \\
3a + 3b + x & \geq 3
\end{align*}
\]

A literal is said to be irrelevant in a PB constraint when its truth value does not impact the truth value of the constraint: irrelevant literals can thus be removed.
**Figure 3:** Statistics about the production of irrelevant literals in *Sat4j-GeneralizedResolution* for each family of benchmarks (logarithmic scale)
Irrelevant literals may become *artificially relevant*, in which case they may impact the strength of the derived constraints.

\[
\begin{align*}
3a + 3b + c & \geq 3 \\
3\bar{a} + 3d + 2c & \geq 3 \\
3b + 3c + 3d & \geq 3 \\
b + c + d & \geq 1
\end{align*}
\]
Artificially Relevant Literals

Irrelevant literals may become *artificially relevant*, in which case they may impact the strength of the derived constraints

\[
\begin{align*}
3a + 3b + c & \geq 3 \\
\frac{3\bar{a} + 3d + 2c}{3b + 3c + 3d} & \geq 3 \\
\frac{b + c + d}{3} & \geq 1
\end{align*}
\]
Irrelevant literals may become *artificially relevant*, in which case they may impact the strength of the derived constraints.

\[
\begin{align*}
3a + 3b + c & \geq 3 \\
3\overline{a} + 3d + 2c & \geq 3 \\
3b + 3c + 3d & \geq 3 \\
b + c + d & \geq 1
\end{align*}
\]
Irrelevant literals may become *artificially relevant*, in which case they may impact the strength of the derived constraints.

\[
\begin{align*}
3a + 3b + \chi & \geq 3 \\
3\bar{a} + 3d + 2c & \geq 3 \\
3b + 3c + 3d & \geq 3 \\
b + c + d & \geq 1
\end{align*}
\]
Irrelevant literals may become **artificially relevant**, in which case they may impact the strength of the derived constraints

\[
\begin{align*}
3a + 3b + \chi & \geq 3 \\
3\bar{a} + 3d + 2\bar{c} & \geq 3 \\
3b + 3c + 3d & \geq 3 \\
b + c + d & \geq 1
\end{align*}
\]
Irrelevant literals may become *artificially relevant*, in which case they may impact the strength of the derived constraints.

\[
\begin{align*}
3a + 3b + \chi & \geq 3 \\
3\bar{a} + 3d + 2\chi & \geq 3 \\
3b + \chi + 3d & \geq 3 \\
b + c + d & \geq 1
\end{align*}
\]
Irrelevant literals may become artificially relevant, in which case they may impact the strength of the derived constraints

\[ 3a + 3b + \chi \geq 3 \quad 3\bar{a} + 3d + 2\zeta \geq 3 \]

\[ 3b + \chi + 3d \geq 3 \]

\[ b + \chi + d \geq 1 \]
Irrelevant literals may become *artificially relevant*, in which case they may impact the strength of the derived constraints

\[
\begin{align*}
3a + 3b + \chi & \geq 3 \\
3\bar{a} + 3d + 2\xi & \geq 3 \\
3b + \bar{\chi} + 3d & \geq 3 \\
b + \chi + d & \geq 1
\end{align*}
\]

Detecting irrelevant literals is *NP-hard*, we thus introduce an *incomplete* algorithm for removing them
Detecting Irrelevant Literals (1)

Irrelevant literals can be detected thanks to this reduction to subset-sum:

\[ l \text{ is irrelevant in } \alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta \]

\[ \iff \sum_{i=1}^{n} \alpha_i l_i = \delta - k \text{ has no solution for } k \in \{1, \ldots, \alpha\} \]
Irrelevant literals can be detected thanks to this reduction to \textit{subset-sum}

\[
\ell \text{ is irrelevant in } \alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \\
\Leftrightarrow \sum_{i=1}^{n} \alpha_i \ell_i = \delta - k \text{ has no solution for } k \in \{1, \ldots, \alpha\}
\]

For instance, \(c\) is irrelevant in \(3a + 3b + 2c \geq 3\) because there is no solution for neither of the \textit{equalities}

\[
3a + 3b = 1 \text{ and } 3a + 3b = 2
\]
Irrelevant literals can be detected thanks to this reduction to \textit{subset-sum}

\[
\ell \text{ is irrelevant in } \alpha \ell + \sum_{i=1}^{n} \alpha_i\ell_i \geq \delta \\
\Leftrightarrow \sum_{i=1}^{n} \alpha_i\ell_i = \delta - k \text{ has no solution for } k \in \{1, \ldots, \alpha\}
\]

For instance, \(c\) is irrelevant in \(3a + 3b + 2c \geq 3\) because there is no solution for neither of the equalities

\[
3a + 3b = 1 \text{ and } 3a + 3b = 2
\]

A \textit{dynamic programming} algorithm can decide whether any of the equalities has a solution in \textit{pseudo-polynomial time with a single run}
As coefficients and degrees may be very big in the derived PB constraints, solving subset-sum on the corresponding instances would be very inefficient.
As coefficients and degrees may be very big in the derived PB constraints, solving subset-sum on the corresponding instances would be very inefficient.

We thus consider an incomplete approach for solving these instances.
As coefficients and degrees may be very big in the derived PB constraints, solving subset-sum on the corresponding instances would be very inefficient.

We thus consider an incomplete approach for solving these instances.

In our case, we want our algorithm to be exact when it detects that the instance has no solution, since the literal is irrelevant in this case (said differently, we accept to miss irrelevant literals, but not the contrary).
As coefficients and degrees may be very big in the derived PB constraints, solving subset-sum on the corresponding instances would be very inefficient.

We thus consider an incomplete approach for solving these instances.

In our case, we want our algorithm to be exact when it detects that the instance has no solution, since the literal is irrelevant in this case (said differently, we accept to miss irrelevant literals, but not the contrary).

Our algorithm solves subset-sum modulo a fixed number, or even several numbers.
Removing Irrelevant Literals

We can remove any irrelevant literal while preserving equivalence, by taking advantage that their truth value does not affect the constraint
We can remove any irrelevant literal while preserving equivalence, by taking advantage that their truth value does not affect the constraint.

\[ 3a + 3b + 2c \geq 3 \]
Removing Irrelevant Literals

We can remove any irrelevant literal while preserving equivalence, by taking advantage that their truth value does not affect the constraint

\[ 3a + 3b + 2c \geq 3 \]

First, we can locally assign the literal to 0, and simply remove it:

\[ 3a + 3b \geq 3 \]
Removing Irrelevant Literals

We can remove any irrelevant literal while preserving equivalence, by taking advantage that their truth value does not affect the constraint:

\[ 3a + 3b + 2c \geq 3 \]

First, we can locally assign the literal to 0, and simply remove it:

\[ 3a + 3b \geq 3 \]

Or, we can locally assign it to 1, and simplify the constraint accordingly:

\[ 3a + 3b \geq 3 - 2 = 1 \]
Removing Irrelevant Literals

We can remove any irrelevant literal while preserving equivalence, by taking advantage that their truth value does not affect the constraint

\[ 3a + 3b + 2c \geq 3 \]

First, we can locally assign the literal to 0, and simply remove it:

\[ 3a + 3b \geq 3 \]

Or, we can locally assign it to 1, and simplify the constraint accordingly:

\[ 3a + 3b \geq 3 - 2 = 1 \]

In practice, we use a heuristic to decide which strategy to apply, as none of them is better than the other.
Figure 4: Comparison of the size of the proofs (number of cancellations) built by *Sat4j-GeneralizedResolution* with and without the removal of irrelevant literals on all benchmarks (logarithmic scale).
Figure 5: Comparison of the size of the proofs (number of cancellations) built by *Sat4j-GeneralizedResolution* with and without the removal of irrelevant literals on vertex-cover instances (logarithmic scale)
When given an instance of this family, the first constraint learned by Sat4j has the form

\[ nx + x_1 + \ldots + x_{n-1} \geq n \]
When given an instance of this family, the first constraint learned by Sat4j has the form

\[ nx + x_1 + \ldots + x_{n-1} \geq n \]

All the literals \( x_1, \ldots, x_{n-1} \) are irrelevant, and this constraint is actually equivalent to the unit clause \( x \).
When given an instance of this family, the first constraint learned by Sat4j has the form

\[ nx + x_1 + \ldots + x_{n-1} \geq n \]

All the literals \( x_1, \ldots, x_{n-1} \) are irrelevant, and this constraint is actually equivalent to the unit clause \( x \)

No other irrelevant literals are detected in the other constraints derived by Sat4j
When given an instance of this family, the first constraint learned by Sat4j has the form

\[ nx + x_1 + \ldots + x_{n-1} \geq n \]

All the literals \(x_1, \ldots, x_{n-1}\) are irrelevant, and this constraint is actually equivalent to the unit clause \(x\).

No other irrelevant literals are detected in the other constraints derived by Sat4j.

Even few irrelevant literals can lead to the production of an exponentially larger proof.
Figure 6: Comparison of the runtime of *Sat4j-GeneralizedResolution* with and without the removal of irrelevant literals on all benchmarks (logarithmic scale)
Recall that, during conflict analysis, some literals may be ineffective.
Recall that, during conflict analysis, some literals may be ineffective

\[3\bar{a} + 3\bar{b} + c + \Box d + \Box e \geq 6\]

\[2a + b + \Box c + f \geq 2\]
Weakening Ineffective Literals

Recall that, during conflict analysis, some literals may be ineffective:

\[ 3\overline{a} + 3\overline{b} + c + d + e \geq 6 \]

\[ 2a + b + c + f \geq 2 \]

Ineffective literals can be seen as *locally* irrelevant, as opposed to the *globally* irrelevant literals presented before.
Weakening Ineffective Literals

Recall that, during conflict analysis, some literals may be ineffective

\[3\overline{a} + 3\overline{b} + c + \overline{d} + e \geq 6\]

\[2a + b + c + f \geq 2\]

Ineffective literals can be seen as locally irrelevant, as opposed to the globally irrelevant literals presented before.

In the context of the current partial assignment, it is easy to detect ineffective literals, but they can only be weakened away (as ineffective literals may be relevant)
Adapting further PB Solvers to CDCL
Figure 7: Overview of the CDCL Algorithm
Figure 8: Use of other strategies in the CDCL Algorithm
It is well known that, in addition to conflict analysis, several features of SAT solvers are crucial for solving problems efficiently, such as:

- branching heuristic
- learned constraint deletion strategy
- restart policy
Motivation

It is well known that, in addition to conflict analysis, several features of SAT solvers are crucial for solving problems efficiently, such as:

- branching heuristic
- learned constraint deletion strategy
- restart policy

*These features are mostly reused as is by current PB solvers, without taking into account the particular properties of PB constraints*
Motivation

It is well known that, in addition to conflict analysis, several features of SAT solvers are crucial for solving problems efficiently, such as:

- branching heuristic
- learned constraint deletion strategy
- restart policy

These features are mostly reused as is by current PB solvers, without taking into account the particular properties of PB constraints.

Our main finding (Le Berre and Wallon, 2021) is that considering the size of the coefficients and the current partial assignment allows to significantly improve the solver.
Comparison of different variants (decision)

Figure 9: Cactus plot of the best configurations of different solvers
Figure 10: Cactus plot of the best configurations of different solvers
Conclusion and Perspectives
Implementations of the cutting planes proof system in PB solvers are not fully satisfactory, as its strength is not fully exploited.

Irrelevant literals may be produced during conflict analysis, and lead to the inference of weaker constraints.

Applying the weakening rule on ineffective literals is a possible (aggressive) counter-measure.

Applying partial weakening and division gives better performance.
Implementations of the cutting planes proof system in PB solvers are not fully satisfactory, as its strength is not fully exploited. Irrelevant literals may be produced during conflict analysis, and lead to the inference of weaker constraints. Applying the weakening rule on ineffective literals is a possible (aggressive) counter-measure. Applying partial weakening and division gives better performance.

Complementary heuristics implemented in CDCL PB solvers can be adapted to take into account properties of PB constraints and to improve the performance of Sat4j.
Perspectives

- Find other strategies for applying cutting planes rules so as to exploit more power of this proof system
- Design such strategies so as to prevent the production of irrelevant literals instead of removing them
Find other strategies for applying cutting planes rules so as to exploit more power of this proof system
• Design such strategies so as to prevent the production of irrelevant literals instead of removing them

• Combine the strategies to exploit their complementarity (e.g., using DAC)
• Identify possible interactions between the new heuristics
Perspectives

- Find other strategies for applying cutting planes rules so as to exploit more power of this proof system.
- Design such strategies so as to prevent the production of irrelevant literals instead of removing them.

- Combine the strategies to exploit their complementarity (e.g., using DAC).
- Identify possible interactions between the new heuristics.

- Improve the detection of the optimal backjump level during conflict analysis.
Perspectives

- Find other strategies for applying cutting planes rules so as to exploit more power of this proof system
- Design such strategies so as to prevent the production of irrelevant literals instead of removing them

- **Combine** the strategies to exploit their complementarity (e.g., using DAC)
- Identify possible interactions between the new heuristics

- Improve the detection of the optimal backjump level during conflict analysis

- Improve the detection of conflicts to deal with the conflictual reasons encountered during conflict analysis
Deep Dive into CDCL Pseudo-Boolean Solvers

Romain Wallon
May 20th, 2021

Laboratoire d’Informatique de l’X (LIX), École Polytechnique
Adapting further PB Solvers
Let us consider again a conflict analysis

\[ 3\bar{a}(?@?) + 3\bar{f}(?@?) + d(?@?) + e(?@?) \geq 5 \]

\[ 6a(?@?) + 3b(?@?) + 3c(?@?) + 3d(?@?) + 3f(?@?) \geq 9 \]
Let us consider again a conflict analysis

\[3\bar{a}(\oplus) + 3\bar{f}(\oplus) + d(\oplus) + e(\oplus) \geq 5\]
\[6a(\oplus) + 3b(1\oplus) + 3c(\oplus) + 3d(\oplus) + 3f(\oplus) \geq 9\]
Let us consider again a conflict analysis

\[ 3\overline{a}(?@?) + 3\overline{f}(?@?) + d(?@?) + e(?@?) \geq 5 \]
\[ 6a(?@?) + 3b(1@1) + 3c(0@2) + 3d(?@?) + 3f(?@?) \geq 9 \]
Let us consider again a conflict analysis

\[ 3\bar{a}(\oplus?) + 3\bar{f}(\oplus?) + d(0\oplus3) + e(\oplus?) \geq 5 \]
\[ 6a(\oplus?) + 3b(1\oplus1) + 3c(0\oplus2) + 3d(\oplus?) + 3f(\oplus?) \geq 9 \]
Let us consider again a conflict analysis

\[
3\bar{a}(1@3) + 3\bar{f}(1@3) + d(0@3) + e(?@?) \geq 5
\]

\[
6a(0@3) + 3b(1@1) + 3c(0@2) + 3d(?@?) + 3f(0@3) \geq 9
\]

The PB constraints involved in this conflict analysis have very different properties compared to clauses!
Let us consider again a conflict analysis:

\[ 3\bar{a}(1\@3) + 3\bar{f}(1\@3) + d(0\@3) + e(?@?) \geq 5 \]
\[ 6a(0\@3) + 3b(1\@1) + 3c(0\@2) + 3d(?@?) + 3f(0\@3) \geq 9 \]

We now apply the cancellation rule between these two constraints:

\[
\begin{align*}
3\bar{a} + 3\bar{f} + d + e & \geq 5 \\
6a + 3b + 3c + 3d + 3f & \geq 9 \\
3a(0\@3) + 3b(1\@1) + 3c(0\@2) + 2\bar{d}(1\@3) + e(?@?) & \geq 7
\end{align*}
\]
Let us consider again a conflict analysis

\[
3\overline{a}(1@3) + 3\overline{f}(1@3) + d(0@3) + e(?@?) \geq 5
\]
\[
6a(0@3) + 3b(1@1) + 3c(0@2) + 3d(?@?) + 3f(0@3) \geq 9
\]

We now apply the cancellation rule between these two constraints:

\[
\frac{3\overline{a} + 3\overline{f} + d + e \geq 5}{3a(?@?) + 3b(1@1) + 3c(0@2) + 2\overline{d}(?@?) + e(?@?) \geq 7}
\]
\[
6a + 3b + 3c + 3d + 3f \geq 9
\]
Let us consider again a conflict analysis

\[ 3\bar{a}(1\@3) + 3\bar{f}(1\@3) + d(0\@3) + e(?@?) \geq 5 \]
\[ 6a(0\@3) + 3b(1\@1) + 3c(0\@2) + 3d(?@?) + 3f(0\@3) \geq 9 \]

We now apply the cancellation rule between these two constraints:

\[
\begin{align*}
3\bar{a} + 3\bar{f} + d + e & \geq 5 \\
6a + 3b + 3c + 3d + 3f & \geq 9 \\
3a(?@?) + 3b(1\@1) + 3c(0\@2) + 2d(?@?) + e(?@?) & \geq 7
\end{align*}
\]

The PB constraints involved in this conflict analysis have very different properties compared to clauses!
All variables encountered during conflict analysis are bumped.
All variables encountered during conflict analysis are **bumped**

This is the case for all the variables appearing in the previous **reason**:

\[ 3\bar{a} + 3\bar{f} + d + e \geq 5 \]
All variables encountered during conflict analysis are bumped. This is the case for all the variables appearing in the previous reason:

\[3\bar{a} + 3\bar{f} + d + e \geq 5\]

This means that the scores of the variables a, f, d and e are incremented.
A first approach for adapting VSIDS to PB constraints has been proposed in (Dixon, 2004), but it only takes into account the original cardinality constraints, and thus not the reason we have here:

\[ 3\bar{a} + 3\bar{f} + d + e \geq 5 \]
A first approach for adapting VSIDS to PB constraints has been proposed in (Dixon, 2004), but it only takes into account the original cardinality constraints, and thus not the reason we have here:

\[ 3\overline{a} + 3\overline{f} + d + e \geq 5 \]

We propose to take these coefficients into account with 3 new strategies:
A first approach for adapting VSIDS to PB constraints has been proposed in (Dixon, 2004), but it only takes into account the original cardinality constraints, and thus not the reason we have here:

\[ 3\bar{a} + 3\bar{f} + d + e \geq 5 \]

We propose to take these coefficients into account with 3 new strategies:

- **bump-degree**: the score of each variable is incremented by the degree of the constraint (5 for all variables)
A first approach for adapting VSIDS to PB constraints has been proposed in (Dixon, 2004), but it only takes into account the original cardinality constraints, and thus not the reason we have here:

\[ 3\bar{a} + 3\bar{f} + d + e \geq 5 \]

We propose to take these coefficients into account with 3 new strategies:

- **bump-degree**: the score of each variable is incremented by the **degree** of the constraint (5 for all variables)

- **bump-coefficient**: the score of each variable is incremented by their **coefficients** in the constraint (3 for \( a \) and \( f \))
A first approach for adapting VSIDS to PB constraints has been proposed in (Dixon, 2004), but it only takes into account the original cardinality constraints, and thus not the reason we have here:

\[ 3\overline{a} + 3\overline{f} + d + e \geq 5 \]

We propose to take these coefficients into account with 3 new strategies:

- **bump-degree**: the score of each variable is incremented by the degree of the constraint (5 for all variables)

- **bump-coefficient**: the score of each variable is incremented by their coefficients in the constraint (3 for \( a \) and \( f \))

- **bump-ratio**: the score of each variable is incremented by the ratio of the degree by their coefficient in the constraint (\( \frac{5}{3} \) for \( a \) and \( f \))
(E) VSIDS for Making Decisions: Experiments

![Graph showing time (s) vs. number of non-easy instances for different strategies.]

- VBS
- bump-ratio-coefficient-degree
- bump-default
- bump-coefficient
- bump-degree
- bump-ratio-degree-coefficient
Observe also that some literals are *unassigned* in the reason:

\[3\bar{a} + 3\bar{f} + d + e \geq 5\]
Observe also that some literals are unassigned in the reason:

\[ 3\overline{a} + 3\overline{f} + d + e \geq 5 \]

In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation.
Observe also that some literals are unassigned in the reason:

\[ 3\overline{a} + 3\overline{f} + d + e \geq 5 \]

*In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation*

We can take the current assignment into account with 3 new strategies:
Observe also that some literals are unassigned in the reason:

\[ 3\bar{a} + 3\bar{f} + d + e \geq 5 \]

*In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation*

We can take the current assignment into account with 3 new strategies:

- **bump-assigned**: the score of each assigned variable is incremented (\(a, f\) and \(d\))
Observe also that some literals are unassigned in the reason:

$$3\overline{a} + 3\overline{f} + d + e \geq 5$$

*In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation*

We can take the current assignement into account with 3 new strategies:

- **bump-assigned**: the score of each assigned variable is incremented ($a$, $f$ and $d$)
- **bump-falsified**: the score of each falsified variable is incremented ($f$ and $d$)
Observe also that some literals are unassigned in the reason:

$$3\bar{a} + 3\bar{f} + d + e \geq 5$$

In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation.

We can take the current assignment into account with 3 new strategies:

- **bump-assigned**: the score of each assigned variable is incremented ($a$, $f$ and $d$)
- **bump-falsified**: the score of each falsified variable is incremented ($f$ and $d$)
- **bump-effective**: the score of each effective variable is incremented ($f$ and $d$)
(E)VSIDS for Making Decisions: Experiments

![Graph showing the number of non-easy instances vs time for different strategies: VBS, bump-effective, bump-falsified, bump-effective-propagated, bump-falsified-propagated, bump-assigned, and bump-default. The graph indicates the performance of these strategies over varying numbers of instances.](image-url)
In SAT solvers, evaluating the quality of learned constraints is used to choose which constraints should be deleted and to decide when a restart should be triggered.
In SAT solvers, evaluating the quality of learned constraints is used to choose which constraints should be deleted and to decide when a restart should be triggered.

*The quality measures used by SAT solvers do not take into account the properties of PB constraints.*
Quality of Learned Constraints: Size and Coefficients

In SAT solvers, the size of a clause is a naive measure of its quality: the longer the clause, the lower its strength.
In SAT solvers, the size of a clause is a naive measure of its quality: the longer the clause, the lower its strength.

In the PB case, the length of a constraint does not reflect its strength.
In SAT solvers, the size of a clause is a naive measure of its quality: the longer the clause, the lower its strength.

In the PB case, the length of a constraint does not reflect its strength.

However, the size of a PB constraint also takes into account its coefficients.
In SAT solvers, the size of a clause is a naive measure of its quality: the longer the clause, the lower its strength.

In the PB case, the length of a constraint does not reflect its strength.

However, the size of a PB constraint also takes into account its coefficients.

Consider the constraint we derived in the previous conflict analysis:

\[3a + 3b + 3c + 2d + e \geq 7\]
In SAT solvers, the size of a clause is a naive measure of its quality: the longer the clause, the lower its strength.

In the PB case, the length of a constraint does not reflect its strength.

However, the size of a PB constraint also takes into account its coefficients.

Consider the constraint we derived in the previous conflict analysis:

\[3a + 3b + 3c + 2d + e \geq 7\]

In practice, the coefficients may become very big, which requires the use of arbitrary precision encodings and slows down arithmetic operations.
Quality of Learned Constraints: Size and Coefficients

In SAT solvers, the size of a clause is a naive measure of its quality: the longer the clause, the lower its strength.

In the PB case, the length of a constraint does not reflect its strength.

However, the size of a PB constraint also takes into account its coefficients.

Consider the constraint we derived in the previous conflict analysis:

\[ 3a + 3b + 3c + 2\bar{d} + e \geq 7 \]

In practice, the coefficients may become very big, which requires the use of arbitrary precision encodings and slows down arithmetic operations.

We consider quality measures based on the value and size of the degree of the constraints: the lower the degree, the better the constraint.
In SAT solvers, the size of a clause is a naive measure of its quality: the longer the clause, the lower its strength.

In the PB case, the length of a constraint does not reflect its strength.

However, the size of a PB constraint also takes into account its coefficients.

Consider the constraint we derived in the previous conflict analysis:

$$3a + 3b + 3c + 2\bar{d} + e \geq 7$$

In practice, the coefficients may become very big, which requires the use of arbitrary precision encodings and slows down arithmetic operations.

We consider quality measures based on the value and size of the degree of the constraints: the lower the degree, the better the constraint.
Quality of Learned Constraints: Assignments (LBD)

In SAT solvers, the **Literal Block Distance** (Audemard and Simon, 2009) measures the quality of clauses by the number of decision levels appearing in this clause.

\[
\text{lbd-a} = a_0 + 3a_1 + 3a_2 + \bar{d}_1 + \bar{d}_3 \\
\text{lbd-s} = a_0 + 3a_1 + 3a_2 + \bar{d}_1 + \bar{d}_3 + \Sigma \text{unassigned literals}
\]

- **lbd-a**: the LBD is computed over **assigned** literals only
- **lbd-s**: the LBD is computed over **assigned** literals, and unassigned literals are considered assigned at the same (dummy) decision level
- **lbd-d**: the LBD is computed over **assigned** literals, and unassigned literals are considered assigned at different (dummy) decision levels
- **lbd-f**: the LBD is computed over **falsified** literals only
- **lbd-e**: the LBD is computed over **effective** literals only
In SAT solvers, the **Literal Block Distance** (Audemard and Simon, 2009) measures the quality of clauses by the number of decision levels appearing in this clause:

\[
3a(0@3) + 3b(1@1) + 3c(0@2) + 2\bar{d}(1@3) + e(?@?) \geq 7
\]
Quality of Learned Constraints: Assignments (LBD)

In SAT solvers, the **Literal Block Distance** (Audemard and Simon, 2009) measures the quality of clauses by the number of *decision levels* appearing in this clause

\[
3a(0@3) + 3b(1@1) + 3c(0@2) + 2\bar{d}(1@3) + e(?@?) \geq 7
\]

*There are satisfied and unassigned literals in this constraint!*
In SAT solvers, the **Literal Block Distance** (Audemard and Simon, 2009) measures the quality of clauses by the number of decision levels appearing in this clause

\[ 3a(0@3) + 3b(1@1) + 3c(0@2) + 2\bar{d}(1@3) + e(?@?) \geq 7 \]

*There are satisfied and unassigned literals in this constraint!*

We thus introduce 5 new definitions of LBD:

- **lbd-a**: the LBD is computed over assigned literals only
- **lbd-s**: the LBD is computed over assigned literals, and unassigned literals are considered assigned at the same (dummy) decision level
- **lbd-d**: the LBD is computed over assigned literals, and unassigned literals are considered assigned at different (dummy) decision levels
- **lbd-f**: the LBD is computed over falsified literals only
- **lbd-e**: the LBD is computed over effective literals only
Deleting constraints is required by SAT solvers to limit the memory usage and to prevent unit propagation from slowing down.
Deleting constraints is required by SAT solvers to limit the memory usage and to prevent unit propagation from slowing down.

This is also true, but to a lesser extent, for PB solvers.
Deleting constraints is required by SAT solvers to limit the memory usage and to prevent unit propagation from slowing down.

This is also true, but to a lesser extent, for PB solvers.

The constraints to delete are those having a bad score w.r.t. the quality measure used in the solver.
Deleting constraints is required by SAT solvers to limit the memory usage and to prevent unit propagation from slowing down. This is also true, but to a lesser extent, for PB solvers. The constraints to delete are those having a bad score w.r.t. the quality measure used in the solver.

We thus introduce the following deletion strategies, based on the different quality measures we presented:

- delete-degree
- delete-degree-size
- delete-lbd-a
- delete-lbd-s
- delete-lbd-d
- delete-lbd-f
- delete-lbd-e
Quality of Learned Constraints: Restarts

Restarting allows to forget all decisions made by the solver, so as to avoid being stuck in a subpart of the search space.
Restarting allows to forget all decisions made by the solver, so as to avoid being stuck in a subpart of the search space.

Following *Glucose*’s approach (Audemard and Simon, 2012), we consider adaptive restarts based on the quality of recently learned constraints.
Quality of Learned Constraints: Restarts

**Restarting** allows to forget all decisions made by the solver, so as to avoid being stuck in a subpart of the search space.

Following *Glucose’s* approach (Audemard and Simon, 2012), we consider **adaptive restarts** based on the quality of recently learned constraints.

Whenever the *most recent constraints are of poor quality compared to all the others*, a restart is performed.
Restarting allows to forget all decisions made by the solver, so as to avoid being stuck in a subpart of the search space.

Following *Glucose*’s approach (Audemard and Simon, 2012), we consider adaptive restarts based on the quality of recently learned constraints. Whenever the most recent constraints are of poor quality compared to all the others, a restart is performed.

We thus introduce the following restart strategies, based on the different quality measures we presented:

- restart-degree
- restart-degree-size
- restart-lbd-a
- restart-lbd-s
- restart-lbd-d
- restart-lbd-f
- restart-lbd-e
Comparison of different variants (decision)

Figure 11: Cactus plot of the best configurations of different solvers
Comparison of different variants (optimization)

Figure 12: Cactus plot of the best configurations of different solvers


Bibliography


