Pseudo-Boolean Reasoning and Compilation

PhD Defense

Romain Wallon
December 14th, 2020
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
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

$$(a \lor b \lor \neg c) \land (a \lor \neg b \lor c) \land (\neg a \lor \neg b \lor c)$$
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

$$(a \lor b \lor \neg c) \land (a \lor \neg b \lor c) \land (\neg a \lor \neg b \lor c)$$
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

\[
(a \lor b \lor \neg c) \land (a \lor \neg b \lor c) \land (\neg a \lor \neg b \lor c)
\]

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

\[
(a \lor b \lor \neg c) \land (a \lor \neg b \lor c) \land (\neg a \lor \neg b \lor c)
\]

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 can now deal with problems containing millions of variables and clauses*
Despite their practical efficiency, some instances remain completely out of reach for modern SAT solvers, due to the weakness of the resolution proof system.
SAT Solver Limitations

Despite their practical efficiency, some instances remain completely out of reach for modern SAT solvers, due to the weakness of the resolution proof system. 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.”
Despite their practical efficiency, some instances remain completely out of reach for modern SAT solvers, due to the weakness of the resolution proof system.

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”

On such instances, pseudo-Boolean reasoning and cutting planes based inference can offer better performance.
Pseudo-Boolean Reasoning
Pseudo-Boolean (PB) Constraints

PB solvers are generalizations of SAT solvers that allow to consider

- normalized PB constraints $\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta$
- cardinality constraints $\sum_{i=1}^{n} \ell_i \geq \delta$
- clauses $\sum_{i=1}^{n} \ell_i \geq 1$

in which

- the coefficients $\alpha_i$ are non-negative integers
- $\ell_i$ are literals, i.e., a variable $v$ or its negation $\overline{v} = 1 - v$
- the degree $\delta$ is a non-negative integer
Pseudo-Boolean (PB) Constraints

PB solvers are generalizations of SAT solvers that allow to consider

- **normalized PB constraints** \( \sum_{i=1}^{n} \alpha_i\ell_i \geq \delta \)
- **cardinality constraints** \( \sum_{i=1}^{n} \ell_i \geq \delta \)
- **clauses** \( \sum_{i=1}^{n} \ell_i \geq 1 \)

in which

- the coefficients \( \alpha_i \) are non-negative integers
- \( \ell_i \) are literals, i.e., a variable \( v \) or its negation \( \bar{v} = 1 - v \)
- the degree \( \delta \) is a non-negative integer

\[ 2a + 2b + \bar{c} \geq 3 \]
Pseudo-Boolean (PB) Constraints

PB solvers are generalizations of SAT solvers that allow to consider

- normalized PB constraints $\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta$
- cardinality constraints $\sum_{i=1}^{n} \ell_i \geq \delta$
- clauses $\sum_{i=1}^{n} \ell_i \geq 1$

in which

- the coefficients $\alpha_i$ are non-negative integers
- $\ell_i$ are literals, i.e., a variable $v$ or its negation $\bar{v} = 1 - v$
- the degree $\delta$ is a non-negative integer

$$a + b + \bar{c} \geq 2$$
Pseudo-Boolean (PB) Constraints

PB solvers are generalizations of SAT solvers that allow to consider

- normalized PB constraints \( \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \)
- cardinality constraints \( \sum_{i=1}^{n} \ell_i \geq \delta \)
- clauses \( \sum_{i=1}^{n} \ell_i \geq 1 \)

in which

- the coefficients \( \alpha_i \) are non-negative integers
- \( \ell_i \) are literals, i.e., a variable \( v \) or its negation \( \bar{v} = 1 - v \)
- the degree \( \delta \) is a non-negative integer

\[ a + b + \bar{c} \geq 1 \]
Pseudo-Boolean (PB) Constraints

PB solvers are generalizations of SAT solvers that allow to consider

- normalized PB constraints \( \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \)
- cardinality constraints \( \sum_{i=1}^{n} \ell_i \geq \delta \)
- clauses \( \sum_{i=1}^{n} \ell_i \geq 1 \equiv \bigvee_{i=1}^{n} \ell_i \)

in which

- the coefficients \( \alpha_i \) are non-negative integers
- \( \ell_i \) are literals, i.e., a variable \( \nu \) or its negation \( \bar{\nu} = 1 - \nu \)
- the degree \( \delta \) is a non-negative integer

\[ a + b + \bar{c} \geq 1 \]
Pseudo-Boolean (PB) Constraints

PB solvers are generalizations of SAT solvers that allow to consider

- normalized PB constraints \( \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \)
- cardinality constraints \( \sum_{i=1}^{n} \ell_i \geq \delta \)
- clauses \( \sum_{i=1}^{n} \ell_i \geq 1 \equiv \bigvee_{i=1}^{n} \ell_i \)

in which

- the coefficients \( \alpha_i \) are non-negative integers
- \( \ell_i \) are literals, i.e., a variable \( v \) or its negation \( \bar{v} = 1 - v \)
- the degree \( \delta \) is a non-negative integer

\[ a + b + \bar{c} \geq 1 \equiv a \lor b \lor \neg c \]
Pseudo-Boolean (PB) Constraints

PB solvers are generalizations of SAT solvers that allow to consider

- normalized PB constraints \( \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \)
- cardinality constraints \( \sum_{i=1}^{n} \ell_i \geq \delta \)
- clauses \( \sum_{i=1}^{n} \ell_i \geq 1 \equiv \bigvee_{i=1}^{n} \ell_i \)

in which

- the coefficients \( \alpha_i \) are non-negative integers
- \( \ell_i \) are literals, i.e., a variable \( \nu \) or its negation \( \bar{\nu} = 1 - \nu \)
- the degree \( \delta \) is a non-negative integer

\[
a + b + \bar{c} \geq 1 \equiv a \lor b \lor \neg c
\]

We use the cutting-planes proof system to reason with such constraints
PB solvers often use a subset of cutting planes rules known as **Generalized Resolution** [Hooker, 1988], which uses the following rules

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

\[
\frac{\sum_{i=1}^{n} \alpha_i l_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta} \quad \text{(saturation)}
\]
PB solvers often use a subset of cutting planes rules known as Generalized Resolution [Hooker, 1988], which uses the following rules:

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

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

\[
\sum_{i=1}^{n} \alpha_i l_i \geq \delta \quad \quad \sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta 
\]  
(saturation)
PB solvers often use a subset of cutting planes rules known as **Generalized Resolution** [Hooker, 1988], which uses the following rules:

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

\[
\frac{\sum_{i=1}^{n} \alpha_i l_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta} \quad \text{(saturation)}
\]
PB solvers often use a subset of cutting planes rules known as \textbf{Generalized Resolution} [Hooker, 1988], which uses the following rules:

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

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

\[
\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta) \ell_i \geq \delta} \quad \text{(saturation)}
\]
Cutting Planes and Generalized Resolution

PB solvers often use a subset of cutting planes rules known as Generalized Resolution [Hooker, 1988], which uses the following rules:

\[
\alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta_1
\]

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

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

\[
\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta
\] (saturation)
PB solvers often use a subset of cutting planes rules known as Generalized Resolution [Hooker, 1988], which uses the following rules:

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

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

\[
\sum_{i=1}^{n} \alpha_i l_i \geq \delta \quad \frac{\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta) l_i \geq \delta} \tag{saturation}
\]

As with the resolution rule in classical SAT solvers, these two rules can be used to learn new constraints during conflict analysis.
Another useful rule is that of the division

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

\[
\sum_{i=1}^{n} \left\lceil \frac{\alpha_i}{\rho} \right\rceil \ell_i \geq \left\lceil \frac{\delta}{\rho} \right\rceil
\]
Another useful rule is that of the division

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \left\lceil \frac{\alpha_i}{\rho} \right\rceil \ell_i \geq \left\lceil \frac{\delta}{\rho} \right\rceil} \quad \rho \in \mathbb{N}^*$$
Another useful rule is that of the division

\[
\frac{\sum_{i=1}^{n} \alpha_i l_i \geq \delta}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil l_i \geq \lceil \frac{\delta}{\rho} \rceil} \quad (\text{division})
\]
Another useful rule is that of the division

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

This rule may allow to strengthen a constraint over the reals and can be used to replace the saturation rule.
A $\rightarrow B$ if $A$ is strictly stronger than $B$ († on polynomial size coefficients)

$A \rightarrow B$ if any proof of $B$ can be translated in polynomial time into a proof of $A$
Proof System Strength (on PB Inputs) [Vinyals et al., 2018]

\[ A \rightarrow B \text{ if } A \text{ is strictly stronger than } B \text{ († on polynomial size coefficients)} \]

\[ A \rightarrow B \text{ if any proof of } B \text{ can be translated in polynomial time into a proof of } A \]
Proof System Strength (on PB Inputs) [Vinyals et al., 2018]

\[ A \rightarrow B \text{ if } A \text{ is strictly stronger than } B \text{ († on polynomial size coefficients)} \]

\[ A \rightarrowrightarrow B \text{ if any proof of } B \text{ can be translated in polynomial time into a proof of } A \]
Proof System Strength (on PB Inputs) [Vinyals et al., 2018]

We consider PB solvers that handle PB constraints natively to benefit from the strength of the cutting-planes proof system.
A $\rightarrow B$ if $A$ is strictly stronger than $B$ († on polynomial size coefficients)

$A \longrightarrow B$ if any proof of $B$ can be translated in polynomial time into a proof of $A$

We consider PB solvers that handle PB constraints natively to benefit from the strength of the cutting-planes proof system
Theory

The rules of the proof system are applied based on the *structure* and *semantics* of the constraints.
A Gap Between Theory and Practice

Theory
The rules of the proof system are applied based on the structure and semantics of the constraints.

Practice
The solver has no information about the semantics of the constraints, the global structure of the problem: it reasons locally.
A Gap Between Theory and Practice

Theory
The rules of the proof system are applied based on the structure and semantics of the constraints

Practice
The solver has no information about the semantics of the constraints, the global structure of the problem: it reasons locally

The application of the cancellation rule in PB solvers is guided by the propagations that lead to a conflict
Properties of PB Constraints
PB formulae can be grouped into different languages, depending on the kind of constraints they contain:

- **CNF** formulae are conjunctions of clauses
- **CARD** formulae are conjunctions of cardinality constraints
- **PBC** formulae are conjunctions of any normalized PB constraints
PB formulae can be grouped into different languages, depending on the kind of constraints they contain:

- CNF formulae are conjunctions of clauses
- CARD formulae are conjunctions of cardinality constraints
- PBC formulae are conjunctions of any normalized PB constraints

It is often convenient to use clauses to represent pieces of knowledge, even though a PB constraint is more expressive than a clause.
PB formulae can be grouped into different languages, depending on the kind of constraints they contain:

- **CNF** formulae are conjunctions of clauses
- **CARD** formulae are conjunctions of cardinality constraints
- **PBC** formulae are conjunctions of any normalized PB constraints

It is often convenient to use clauses to represent pieces of knowledge, even though a PB constraint is more expressive than a clause.

*Let us study the pros and cons of using PB constraints from a knowledge representation perspective.*
## Queries [IJCAI’18]

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>VA</th>
<th>CE</th>
<th>IM</th>
<th>EQ</th>
<th>SE</th>
<th>CT</th>
<th>ME</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNF</td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARD</td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBC</td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- √ polynomial-time
- ○ NP-hard

**CO (COnsistency)**  Is a formula consistent?

**VA (VAlidity)**    Is a formula valid?

**CE (Clausal Entailment)**  Is a given clause implied by a formula?

**IM (IMplication)**  Is a formula implied by a given cube/term?

**EQ (EQuivalence)**  Are two formulas equivalent?

**SE (Sentential Entailment)**  Is a formula entailed by an other one?

**CT (CounTing)**  How many models does a formula have?
Transformations [IJCAI’18] (with more recent results)

<table>
<thead>
<tr>
<th></th>
<th>CD</th>
<th>FO</th>
<th>SFO</th>
<th>∧C</th>
<th>∧BC</th>
<th>∨C</th>
<th>∨BC</th>
<th>¬C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNF</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CARD</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PBC</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NNF</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓ polynomial-time  ○ NP-hard  ■ exponential-size

**CD (ConDitioning)** Compute $\phi|\tau$ where $\tau$ is a consistent cube/term

**SFO (Singleton FOrgetting)** Compute $\exists x\phi \equiv (\phi|x) \lor (\phi|x)$

**FO (FOrgetting)** Compute $\exists X\phi$ where $X$ is a set of variables

**∧C (Closure under ∧)** Compute $\land_{i=1}^n \phi_i$

**∧BC (Bounded Closure under ∧)** Compute $\land_{i=1}^n \phi_i$, where $n \leq N$

**∨C (Closure under ∨)** Compute $\lor_{i=1}^n \phi_i$

**∨BC (Bounded Closure under ∨)** Compute $\lor_{i=1}^n \phi_i$, where $n \leq N$

**¬C (Closure under ¬)** Compute $\neg \phi$
Succinctness captures the ability of a language to represent information using little space.
Succinctness captures the ability of a language to represent information using little space.

The main advantage of PB constraints is their succinctness w.r.t. clauses, and the reasoning power brought by the cutting planes proof system.
An Achilles Heel in the Cutting Planes Proof System
Cutting planes rules may introduce irrelevant literals

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

\[
\begin{align*}
3a + 3b + c & \geq 3
\end{align*}
\]
Irrelevant Literals [IJCAI’20]

Cutting planes rules may introduce irrelevant literals

\[
\begin{align*}
3d + a + b + c & \geq 3 \\
\bar{d} + 2a + 2b & \geq 3 \\
3a + 3b + c & \geq 3
\end{align*}
\]
Cutting planes rules may introduce irrelevant literals

\[
\begin{align*}
3d + a + b + c & \geq 3 \\
\underline{3\bar{d} + 2a + 2b \geq 3} \\
3a + 3b + c & \geq 3
\end{align*}
\]
Cutting planes rules may introduce irrelevant literals

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

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.
Cutting planes rules may introduce *irrelevant literals*

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

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*
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

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

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

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

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

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

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

\[
\begin{align*}
3a + 3b + \bar{c} & \geq 3 \\
3\bar{a} + 3d + \bar{z} & \geq 3 \\
3b + \bar{z} + 3d & \geq 3 \\
b + \bar{z} + d & \geq 1
\end{align*}
\]

Detecting irrelevant literals is \textbf{NP-hard}, we thus introduce an incomplete algorithm for removing them
Irrelevant literals can be detected thanks to this reduction to subset-sum

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

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

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

\[ \iff \sum_{i=1}^{n} \alpha_i \ell_i = \delta - k \text{ has no solution for } k \in \{1, \ldots, \alpha\} \]

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

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

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

\[ \iff \sum_{i=1}^{n} \alpha_i \ell_i = \delta - k \text{ has no solution for } k \in \{1, \ldots, \alpha\} \]

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 dynamic programming algorithm can decide whether any of the equalities has a solution in 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.
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

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

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

\[3a + 3b \geq 3 - 2 = 1\]
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 based on the slack to decide which strategy to apply, as none of them is better than the other.
Impact of the Removal of Irrelevant Literals on the Proof

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)
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**
Impact of the Removal of Irrelevant Literals on the Runtime

Comparison of the runtime of Sat4j-GeneralizedResolution with and without the removal of irrelevant literals on all benchmarks (logarithmic scale)
Leveraging Weakening

The weakening rules are defined as follows:

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

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d \quad k \in \mathbb{N} \quad 0 < k \leq \alpha}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad \text{(partial weakening)}
\]
The weakening rules are defined as follows:

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

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i l_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i l_i \geq \delta - k}
\] (partial weakening)
The weakening rules are defined as follows:

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

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

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

(weakening)

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

(partial weakening)

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

The weakening rules are defined as follows:

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

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

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

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

\[ 5a + 5b + 3c \geq 8 \] (partial weakening)
The weakening rules are defined as follows:

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

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

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

The weakening rules are defined as follows:

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

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

\[5a + 5b \geq 5\]
The weakening rules are defined as follows:

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

(weakening)

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

\(k \in \mathbb{N} \quad 0 < k \leq \alpha\)

(partial weakening)
The weakening rules are defined as follows:

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

\[
\frac{\alpha l + \sum_{i=1}^{n} \alpha_i l_i \geq d}{(\alpha - k)l + \sum_{i=1}^{n} \alpha_i l_i \geq \delta - k}\quad k \in \mathbb{N} \quad 0 < k \leq \alpha\quad \text{(partial weakening)}
\]
Leveraging Weakening

The weakening rules are defined as follows:

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

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

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

The weakening rules are defined as follows:

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

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k}
\] \( k \in \mathbb{N} \) \( 0 < k \leq \alpha \) (partial weakening)

\[5a + 5b + 3c \geq 8\]
The weakening rules are defined as follows:

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

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

\[ 5a + (5 - 2)b + 3c \geq 8 - 2 \]
Leveraging Weakening

The weakening rules are defined as follows:

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

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d \quad k \in \mathbb{N} \quad 0 < k \leq \alpha}{(\alpha - k)\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad \text{(partial weakening)}
\]

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

The **weakening** rules are defined as follows:

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

\[
\frac{\alpha \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq d}{(\alpha - k) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta - k} \quad k \in \mathbb{N} \quad 0 < k \leq \alpha \quad \text{(partial weakening)}
\]

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

*These rules are already used by PB solvers to maintain invariants during conflict analysis*
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

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

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

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

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

\[
\overline{b} + c \geq 1
\]
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

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

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

\[
\bar{b} + c \geq 1
\]
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

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

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

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

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

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

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

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

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

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

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

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

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

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)*
Cactus plot of the different removal strategies of irrelevant literals
Considering a similar idea to that of RoundingSat, we propose to use partial weakening instead of weakening.
Considering a similar idea to that of *RoundingSat*, we propose to use partial weakening instead of weakening

\[
\begin{align*}
8a + 7b + 7c + 2d + 2e + f & \geq 11 \\
7a + 7b + 7c + 2d + 2e & \geq 9 \\
a + b + c + d + e & \geq 2
\end{align*}
\]
Considering a similar idea to that of *RoundingSat*, we propose to use **partial weakening** instead of weakening

\[
\begin{align*}
8a + 7b + 7c + 2d + 2e + f &\geq 11 \\
7a + 7b + 7c + 2d + 2e &\geq 9 \\
a + b + c + d + e &\geq 2
\end{align*}
\]
Considering a similar idea to that of *RoundingSat*, we propose to use partial weakening instead of weakening

\[
\begin{align*}
8a + 7b + 7c + 2d + 2e + f & \geq 11 \\
7a + 7b + 7c + 2d + 2e & \geq 9 \\
a + b + c + d + e & \geq 2
\end{align*}
\]
Considering a similar idea to that of *RoundingSat*, we propose to use **partial weakening** instead of weakening

\[
\begin{align*}
8a + 7b + 7c + 2d + 2e + f &\geq 11 \\
7a + 7b + 7c + 2d + 2e &\geq 9 \\
a + b + c + d + e &\geq 2
\end{align*}
\]
Considering a similar idea to that of *RoundingSat*, we propose to use **partial weakening** instead of weakening

\[
\begin{align*}
8a + 7b + 7c + 2d + 2e + f & \geq 11 \\
7a + 7b + 7c + 2d + 2e & \geq 9 \\
a + b + c + d + e & \geq 2
\end{align*}
\]

Observe that the constraint obtained here is **stronger** than the clause \(b + c + d + e \geq 1\) derived by *RoundingSat*
Considering a similar idea to that of *RoundingSat*, we propose to use partial weakening instead of weakening

\[
\begin{align*}
8a + 7b + 7c + 2d + 2e + f & \geq 11 \\
7a + 7b + 7c + 2d + 2e & \geq 9 \\
a + b + c + d + e & \geq 2
\end{align*}
\]

Observe that the constraint obtained here is stronger than the clause \(b + c + d + e \geq 1\) derived by *RoundingSat*

*This operation may be applied on either one or both sides of the cancellation*
Comparison of the runtime of different weakening strategies
Fine Tuning of PB Solvers
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
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 is that considering the size of the coefficients and the current partial assignment allows to significantly improve the solver.
Runtime of Sat4j with Different Configurations

![Graph showing runtime of Sat4j with different configurations.](image-url)

- **VBS**
- **best-combination (Sat4j-PartialRoundingSat)**
- **best-combination (Sat4j-RoundingSat)**
- **best-combination (Sat4j-GeneralizedResolution)**
- **default (Sat4j-PartialRoundingSat)**
- **default (Sat4j-RoundingSat)**
- **default (Sat4j-GeneralizedResolution)**

All configurations are improved by the combination of the new strategies.
All configurations are improved by the combination of the new strategies.
Comparison of Sat4j with RoundingSat

![Graph comparing Sat4j and RoundingSat](image)

- VBS
- RoundingSat
- Sat4j-GeneralizedResolution-Both (sober)
- Sat4j-GeneralizedResolution-Both (default)
- Sat4j-RoundingSat-Both (best-combination, watched literals)
- Sat4j-PartialRoundingSat-Both (best-combination, watched literals)
- Sat4j-GeneralizedResolution-Both (best-combination, watched literals)
- RoundingSat2 (no gmp)
- Sat4j-Resolution (default)
- RoundingSat2 (gmp)
- Sat4j-RoundingSat (best-combination, watched literals)
- Sat4j-PartialRoundingSat (best-combination, watched literals)
- Sat4j-GeneralizedResolution (best-combination, watched literals)
- Sat4j-PartialRoundingSat (default)
- Sat4j-RoundingSat (default)
- Sat4j-GeneralizedResolution (default)
Conclusion and Perspectives
The main advantage of using PB constraints from a knowledge representation perspective is their succinctness.
Conclusion

- The main advantage of using PB constraints from a knowledge representation perspective is their succinctness.

- 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.
Conclusion

- The main advantage of using PB constraints from a knowledge representation perspective is their succinctness

- 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
- Combine the weakening strategies to exploit their complementarity
- 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 weakening strategies to exploit their complementarity.
- Identify possible interactions between the new heuristics.
- Implement the new strategies in other solvers.
- Consider their impact on the resolution of optimization problems.

- 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.
• 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 weakening strategies to exploit their complementarity
• Identify possible interactions between the new heuristics

• Implement the new strategies in other solvers
• Consider their impact on the resolution of optimization problems

• 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 weakening strategies to exploit their **complementarity**
- Identify possible **interactions** between the new heuristics

- Implement the new strategies in **other solvers**
- Consider their impact on the resolution of **optimization problems**

- 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


I am a committer of Sat4j\(^1\) in which I implemented several features:

- Detection and removal of irrelevant literals
- Different weakening strategies (including RoundingSat’s)
- New heuristics dedicated to the resolution of PB problems

All these implementations have also been rigorously experimented and evaluated, before being presented in different venues.

I also contributed to the development of Metrics\(^2\), a Python library and app for analyzing experimental results.

---

\(^1\)https://gitlab.ow2.org/sat4j/sat4j
\(^2\)https://github.com/crillab/metrics
Thanks for your attention! Questions?

**Succinctness [IJCAI'18]**

Succinctness captures the ability of a language to represent information using little space.

![Diagram showing the hierarchy of various logical representations](image)

The main advantage of PB constraints is their succinctness w.r.t. clauses, and the reasoning power brought by the cutting planes proof system.

**Production of Irrelevant Literals**

Statistics about the production of irrelevant literals in *Sat4j-GeneralizedResolution* for each family of benchmarks (logarithmic scale).

**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:

\[
3a + 3b + c + d + e \geq 6 \\
3b + c \geq 6 - 3 - 1 - 1 = 1 \\
b + c \geq 1
\]

\[
2a + b + c + f \geq 2 \\
2a + b + f \geq 2 - 1 = 1 \\
a + b + f \geq 1
\]

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).

**Runtime of Sat4j with Different Configurations**

All configurations are improved by the combination of the new strategies.
Pseudo-Boolean Reasoning and Compilation

PhD Defense

Romain Wallon
December 14th, 2020