Deep Dive into CDCL Pseudo-Boolean Solvers

focusing on the implementation in *Sat4j*

Daniel Le Berre[1], Romain Wallon[2]

Februrary 23rd, 2021

[1]CRIL, Univ Artois & CNRS
[2]Laboratoire d'Informatique de l'X (LIX), École Polytechnique

Context

The CDCL Revolution

In the early 2000s, a revolution in the architecture of SAT solvers happened, with the wide adoption of the CDCL approach (Silva and Sakallah, 1996) and the use of efficient heuristics and data structures (Moskewicz et al., 2001; Eén and Sörensson, 2004)

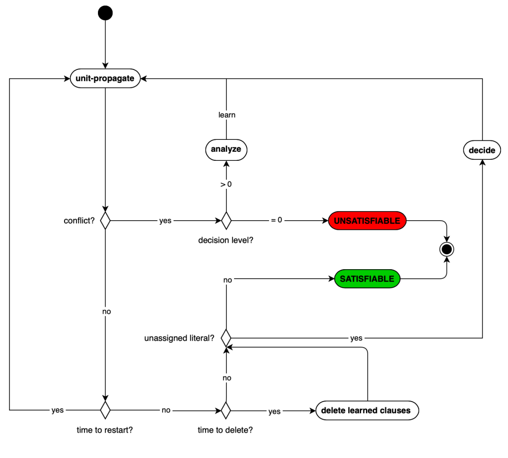
- black box approach
- working on a wide range of (application) problems
- two order of magnitude speedup on some benchmarks compared to previous generation

The CDCL Revolution

In the early 2000s, a revolution in the architecture of SAT solvers happened, with the wide adoption of the CDCL approach (Silva and Sakallah, 1996) and the use of efficient heuristics and data structures (Moskewicz et al., 2001; Eén and Sörensson, 2004)

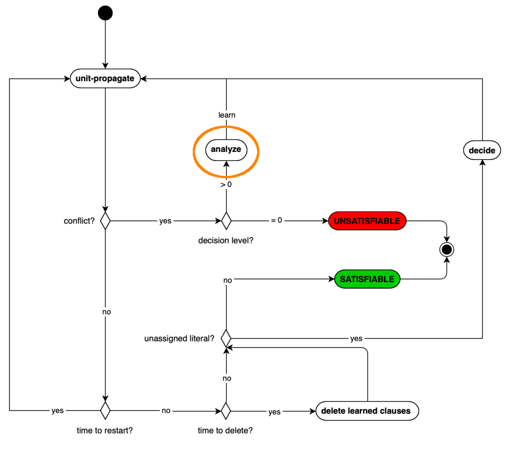
- black box approach
- working on a wide range of (application) problems
- two order of magnitude speedup on some benchmarks compared to previous generation

Modern SAT solvers can now deal with problems containing millions of variables and clauses



Overview of the CDCL Algorithm

Extending the CDCL Architecture



Use of the proof system in the CDCL Algorithm

Some CDCL Invariants

Important invariants of the CDCL algorithm in SAT solvers are:

- Constraints propagate only once
- Constraints have a single assertion level
- Combination of a reason and conflict leads to a conflict
- Syntactical assertion detection

Some CDCL Invariants

Important invariants of the CDCL algorithm in SAT solvers are:

- Constraints propagate only once
- Constraints have a single assertion level
- Combination of a reason and conflict leads to a conflict
- Syntactical assertion detection

We will break these invariants with PB solvers!

Sat4j (Le Berre and Parrain, 2010)

- Open Source SAT solver library in Java developed since 2004
- Support for pseudo-Boolean solving and MAXSAT
- Native PB constraints support
- Various proof systems support with PB constraints
- Available from `http://sat4j.org/`

Pseudo-Boolean Constraints

Why using PB Constraints?

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

Why using PB Constraints?

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"*

Why using PB Constraints?

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

Pseudo-Boolean (PB) Constraints

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

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

in which

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

Pseudo-Boolean (PB) Constraints

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

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

in which

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

For example:

$$-3a + 4b - 7c + d \leq -5$$

Normalized PB Constraints

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 α_i are non-negative integers
- ℓ_i are literals
- the degree δ is a non-negative integer

Normalized PB Constraints

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 α_i are non-negative integers
- ℓ_i are literals
- the degree δ is a non-negative integer

For example:

$$-3a + 4b - 7c + d \leq -5 \equiv 3a + 4\bar{b} + 7c + \bar{d} \geq 10$$

Fun Facts

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

Fun Facts

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

Fun Facts

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

Fun Facts

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

The constraint above can be rewritten as $c \wedge 3a + 4\bar{b} + \bar{d} \geq 3$
but also as $c \wedge (a \vee \bar{b})$

A PB Encoding for Pigeonhole-Principle Formulae

We consider Boolean variables p_{ij} denoting that pigeon i is put in hole j

A PB Encoding for Pigeonhole-Principle Formulae

We consider Boolean variables p_{ij} denoting that pigeon i is put in hole j

"Pigeon i should be in a hole" is encoded as

$$\sum_{j=1}^{n-1} p_{ij} \geq 1$$

A PB Encoding for Pigeonhole-Principle Formulae

We consider Boolean variables p_{ij} denoting that pigeon i is put in hole j

"Pigeon i should be in a hole" is encoded as

$$\sum_{j=1}^{n-1} p_{ij} \geq 1$$

"Hole j cannot host more than one pigeon" is encoded as

$$\sum_{i=1}^{n} p_{ij} \leq 1$$

A PB Encoding for Pigeonhole-Principle Formulae

We consider Boolean variables p_{ij} denoting that pigeon i is put in hole j

"Pigeon i should be in a hole" is encoded as

$$\sum_{j=1}^{n-1} p_{ij} \geq 1$$

"Hole j cannot host more than one pigeon" is encoded as

$$\sum_{i=1}^{n} p_{ij} \leq 1$$

Let us see how to prove the unsatifiability of this formula

Example of a Human Proof ($n = 3$)

$$(1)\ p_{11} + p_{12} \geq 1$$

$$(2)\ p_{21} + p_{22} \geq 1$$

$$(3)\ p_{31} + p_{32} \geq 1$$

$$(4)\ p_{11} + p_{21} + p_{31} \leq 1$$

$$(5)\ p_{12} + p_{22} + p_{32} \leq 1$$

Example of a Human Proof ($n = 3$)

$$(1)\ p_{11} + p_{12} \geq 1$$

$$(2)\ p_{21} + p_{22} \geq 1$$

$$(3)\ p_{31} + p_{32} \geq 1$$

$$(4)\ \overline{p_{11}} + \overline{p_{21}} + \overline{p_{31}} \geq 2$$

$$(5)\ \overline{p_{12}} + \overline{p_{22}} + \overline{p_{32}} \geq 2$$

Example of a Human Proof ($n = 3$)

$$(1)\ p_{11} + p_{12} \geq 1$$

$$(2)\ p_{21} + p_{22} \geq 1$$

$$(3)\ p_{31} + p_{32} \geq 1$$

$$(4)\ \overline{p_{11}} + \overline{p_{21}} + \overline{p_{31}} \geq 2$$

$$(5)\ \overline{p_{12}} + \overline{p_{22}} + \overline{p_{32}} \geq 2$$

$$(1) + (2) + (3) + (4) = (6)\ p_{12} + p_{22} + p_{32} \geq 2$$

Example of a Human Proof ($n = 3$)

$$(1)\ p_{11} + p_{12} \geq 1$$

$$(2)\ p_{21} + p_{22} \geq 1$$

$$(3)\ p_{31} + p_{32} \geq 1$$

$$(4)\ \overline{p_{11}} + \overline{p_{21}} + \overline{p_{31}} \geq 2$$

$$(5)\ \overline{p_{12}} + \overline{p_{22}} + \overline{p_{32}} \geq 2$$

$$(1) + (2) + (3) + (4) = (6)\ p_{12} + p_{22} + p_{32} \geq 2$$

$$(5) + (6) = (7)\ 3 \geq 4$$

Human vs Solver, Complexity Theory vs Modeling

In theory, the input must be the same when talking about complexity

- requires, e.g., input in CNF for comparing resolution vs cutting-planes
- does not allow efficient encodings which rely on the addition of new variables
- rely on “recovering” the cardinality constraints using domain knowledge

Human vs Solver, Complexity Theory vs Modeling

In theory, the input must be the same when talking about complexity

- requires, e.g., input in CNF for comparing resolution vs cutting-planes
- does not allow efficient encodings which rely on the addition of new variables
- rely on "recovering" the cardinality constraints using domain knowledge

In practice, the way the constraints are expressed matters:

- easier to read, to understand the model for a human
- the number of constraints may be different ($\frac{n*(n-1)}{2}$ vs $n-1$)
- the solver can apply new inference rules (e.g., cutting-planes) on higher abstraction constraints

Human vs Solver, Complexity Theory vs Modeling

In theory, the input must be the same when talking about complexity

- requires, e.g., input in CNF for comparing resolution vs cutting-planes
- does not allow efficient encodings which rely on the addition of new variables
- rely on "recovering" the cardinality constraints using domain knowledge

In practice, the way the constraints are expressed matters:

- easier to read, to understand the model for a human
- the number of constraints may be different ($\frac{n*(n-1)}{2}$ vs $n-1$)
- the solver can apply new inference rules (e.g., cutting-planes) on higher abstraction constraints

In practice, current PB solvers behave as (slow) SAT solvers when given a CNF formula as input

Fitting Cutting-Planes into the CDCL architecture

Cutting Planes and Generalized Resolution

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

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\overline{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Cutting Planes and Generalized Resolution

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

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\bar{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Cutting Planes and Generalized Resolution

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

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i\ell_i \geq \delta_1 \qquad \beta\bar{\ell} + \sum_{i=1}^{n} \beta_i\ell_i \geq \delta_2}{\sum_{i=1}^{n}(\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i\ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Cutting Planes and Generalized Resolution

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

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\bar{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Cutting Planes and Generalized Resolution

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

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\bar{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Cutting Planes and Generalized Resolution

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

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i\ell_i \geq \delta_1 \qquad \beta\bar{\ell} + \sum_{i=1}^{n} \beta_i\ell_i \geq \delta_2}{\sum_{i=1}^{n}(\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i\ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

As with the resolution rule in classical SAT solvers, these two rules can be used to learn new constraints during conflict analysis

Analyzing Conflicts

Suppose that we have the following constraints:

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

$5a + 4b + c + d \geq 6$

Analyzing Conflicts

Suppose that we have the following constraints:

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

$5a + 4b + c + d \geq 6$

Analyzing Conflicts

Suppose that we have the following constraints:

$6\overline{b} + 6c + 4e + f + g + h \geq 7$

$5a + 4b + c + d \geq 6$

Analyzing Conflicts

Suppose that we have the following constraints:

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

$5a + 4b + c + d \geq 6$

Analyzing Conflicts

Suppose that we have the following constraints:

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

(reason for $\bar{b}$)

$5a + 4b + c + d \geq 6$

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)

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:

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

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:

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

The constraint we obtain here is no longer conflicting!

Weakening

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

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

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

Weakening

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

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

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

Weakening

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

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

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

Weakening

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

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

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

$$5a + 5b + 3c \geq 8$$

Weakening

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

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

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

$$5a + 5b + 3c \geq 8$$

Weakening

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

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

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

$$5a + 5b \geq 8 - 3$$

Weakening

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

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

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

$$5a + 5b \geq 5$$

Weakening

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

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

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

Weakening

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

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

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

Weakening

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

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

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

$$5a + 5b + 3c \geq 8$$

Weakening

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

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

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

$$5a + 5b + 3c \geq 8$$

Weakening

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

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

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

$$5a + (5 - 2)b + 3c \geq 8 - 2$$

Weakening

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

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

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

$$5a + 3b + 3c \geq 6$$

Weakening

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

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

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

$$5a + 3b + 3c \geq 6$$

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

$$\dfrac{\dfrac{3\bar{a} + 3\bar{b} + c + d + e \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

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

$$\frac{\dfrac{3\bar{a} + 3\bar{b} + c + d + e \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

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

$$\frac{\dfrac{\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

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

$$\cfrac{\cfrac{\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

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

$$\dfrac{\dfrac{\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

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

$$\dfrac{\dfrac{\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

$$\dfrac{\dfrac{2a + b + c + f \geq 2}{2a + b + f \geq 2 - 1 = 1}}{a + b + f \geq 1}$$

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

$$\dfrac{\dfrac{\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

$$\dfrac{\dfrac{2a + b + \boxed{c} + f \geq 2}{2a + b + f \geq 2 - 1 = 1}}{a + b + f \geq 1}$$

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

$$\cfrac{\cfrac{\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

$$\cfrac{\cfrac{2a + b + \boxed{c} + f \geq 2}{2a + b + f \geq 2 - 1 = 1}}{a + b + f \geq 1}$$

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

$$\dfrac{\dfrac{\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1}$$

$$\dfrac{\dfrac{2a + b + \boxed{c} + f \geq 2}{2a + b + f \geq 2 - 1 = 1}}{a + b + f \geq 1}$$

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

$$\dfrac{\dfrac{\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6}{3\bar{b} + c \geq 6 - 3 - 1 - 1 = 1}}{\bar{b} + c \geq 1} \qquad \dfrac{\dfrac{2a + b + \boxed{c} + f \geq 2}{2a + b + f \geq 2 - 1 = 1}}{a + b + f \geq 1}$$

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

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \qquad \rho \in \mathbb{N}^*}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil \ell_i \geq \lceil \frac{\delta}{\rho} \rceil} \text{(division)}$$

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

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \qquad \rho \in \mathbb{N}^*}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil \ell_i \geq \lceil \frac{\delta}{\rho} \rceil} \text{(division)}$$

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

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \qquad \rho \in \mathbb{N}^*}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil \ell_i \geq \lceil \frac{\delta}{\rho} \rceil} \text{(division)}$$

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

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \qquad \rho \in \mathbb{N}^*}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil \ell_i \geq \lceil \frac{\delta}{\rho} \rceil} \text{ (division)}$$

$$\cfrac{8a + 7b + 7c + 2d + 2e + f \geq 11}{\cfrac{7b + 7c + 2d + 2e \geq 2}{b + c + d + e \geq 1}}$$

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

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \qquad \rho \in \mathbb{N}^*}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil \ell_i \geq \lceil \frac{\delta}{\rho} \rceil} \text{(division)}$$

$$\cfrac{8a + 7b + 7c + 2d + 2e + f \geq 11}{\cfrac{7b + 7c + 2d + 2e \geq 2}{b + c + d + e \geq 1}}$$

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

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \qquad \rho \in \mathbb{N}^*}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil \ell_i \geq \lceil \frac{\delta}{\rho} \rceil} \text{(division)}$$

$$\cfrac{8a + 7b + 7c + 2d + 2e + f \geq 11}{\cfrac{7b + 7c + 2d + 2e \geq 2}{b + c + d + e \geq 1}}$$

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

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \qquad \rho \in \mathbb{N}^*}{\sum_{i=1}^{n} \left\lceil \frac{\alpha_i}{\rho} \right\rceil \ell_i \geq \left\lceil \frac{\delta}{\rho} \right\rceil} \text{(division)}$$

$$\cfrac{8a + 7b + 7c + 2d + 2e + f \geq 11}{\cfrac{7b + 7c + 2d + 2e \geq 2}{b + c + d + e \geq 1}}$$

Weakening and Division

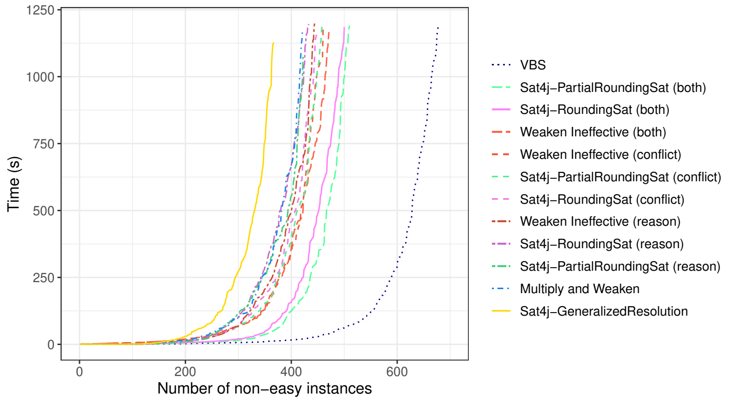
In *RoundingSat* (Elffers and Nordström, 2018), the coefficient is rounded to one thanks to the division rule, applied after having weakened away some unfalsified literals

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta \qquad \rho \in \mathbb{N}^*}{\sum_{i=1}^{n} \lceil \frac{\alpha_i}{\rho} \rceil \ell_i \geq \lceil \frac{\delta}{\rho} \rceil} \text{(division)}$$

$$\cfrac{8a + 7b + 7c + 2d + 2e + f \geq 11}{\cfrac{7b + 7c + 2d + 2e \geq 2}{b + c + d + e \geq 1}}$$

It is also possible to apply partial weakening before division to infer stronger constraints

Many Different Strategies



When to Stop Conflict Analysis?

For clausal analysis:

- stop when a single literal from current decision level remains
- backjump at the deepest decision level but current one among the literals

When to Stop Conflict Analysis?

For clausal analysis:

- stop when a single literal from current decision level remains
- backjump at the deepest decision level but current one among the literals

For PB analysis, no such syntactical detection:

- depends on the weights of the literals assigned at each decision level
- backjump at the first decision level propagating a truth value

An Achilles Heel in the Cutting Planes Proof System

Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

$$\frac{3d + a + b + c \geq 3 \qquad 3\bar{d} + 2a + 2b \geq 3}{3a + 3b + c \geq 3}$$

Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

$$\frac{3d + a + b + c \geq 3 \qquad 3\bar{d} + 2a + 2b \geq 3}{3a + 3b + c \geq 3}$$

Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

$$\frac{3d + a + b + c \geq 3 \qquad 3\bar{d} + 2a + 2b \geq 3}{3a + 3b + c \geq 3}$$

Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

$$\frac{3d + a + b + c \geq 3 \qquad 3\bar{d} + 2a + 2b \geq 3}{3a + 3b + c \geq 3}$$

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

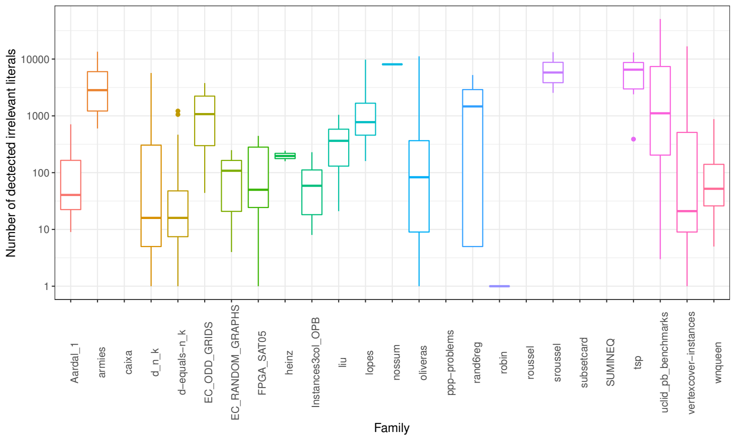
Irrelevant Literals (Le Berre et al., 2020a)

Cutting planes rules may introduce irrelevant literals

$$\frac{3d + a + b + c \geq 3 \qquad 3\bar{d} + 2a + 2b \geq 3}{3a + 3b + \cancel{c} \geq 3}$$

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

Production of Irrelevant Literals



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

Artificially Relevant Literals

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

$$\cfrac{\cfrac{3a + 3b + c \geq 3 \qquad 3\bar{a} + 3d + 2c \geq 3}{3b + 3c + 3d \geq 3}}{b + c + d \geq 1}$$

Artificially Relevant Literals

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

$$\dfrac{\dfrac{3a + 3b + c \geq 3 \qquad 3\bar{a} + 3d + 2c \geq 3}{3b + 3c + 3d \geq 3}}{b + c + d \geq 1}$$

Artificially Relevant Literals

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

$$\dfrac{\dfrac{3a + 3b + c \geq 3 \qquad 3\bar{a} + 3d + 2c \geq 3}{3b + 3c + 3d \geq 3}}{b + c + d \geq 1}$$

Artificially Relevant Literals

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

$$\dfrac{\dfrac{3a + 3b + \cancel{c} \geq 3 \qquad 3\bar{a} + 3d + 2c \geq 3}{3b + 3c + 3d \geq 3}}{b + c + d \geq 1}$$

Artificially Relevant Literals

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

$$\dfrac{\dfrac{3a + 3b + \cancel{c} \geq 3 \qquad 3\bar{a} + 3d + \cancel{2c} \geq 3}{3b + 3c + 3d \geq 3}}{b + c + d \geq 1}$$

Artificially Relevant Literals

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

$$\dfrac{\dfrac{3a + 3b + \cancel{c} \geq 3 \qquad 3\bar{a} + 3d + \cancel{2c} \geq 3}{3b + \cancel{3c} + 3d \geq 3}}{b + c + d \geq 1}$$

Artificially Relevant Literals

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

$$\dfrac{\dfrac{3a + 3b + \cancel{c} \geq 3 \qquad 3\bar{a} + 3d + \cancel{2c} \geq 3}{3b + \cancel{3c} + 3d \geq 3}}{b + \cancel{c} + d \geq 1}$$

Artificially Relevant Literals

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

$$\dfrac{\dfrac{3a + 3b + \cancel{c} \geq 3 \qquad 3\bar{a} + 3d + \cancel{2c} \geq 3}{3b + \cancel{3c} + 3d \geq 3}}{b + \cancel{c} + d \geq 1}$$

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

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

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

Detecting Irrelevant Literals (1)

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

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

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

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

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

Detecting Irrelevant Literals (1)

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

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

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

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 dynamic programming algorithm can decide whether any of the equalities has a solution in pseudo-polynomial time with a single run

Detecting Irrelevant Literals (2)

As coefficients and degrees may be very big in the derived PB constraints, solving subset-sum on the corresponding instances would be very inefficient

Detecting Irrelevant Literals (2)

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

Detecting Irrelevant Literals (2)

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)

Detecting Irrelevant Literals (2)

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

Removing Irrelevant Literals

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

$$3a + 3b + 2c \geq 3$$

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:

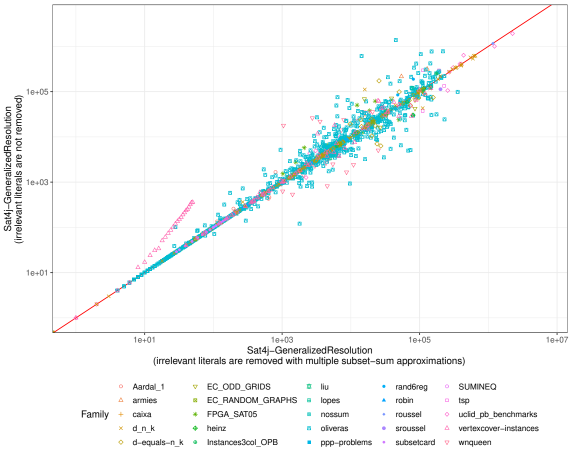
$$3a + 3b \geq 3$$

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

$$3a + 3b \geq 3 - 2 = 1$$

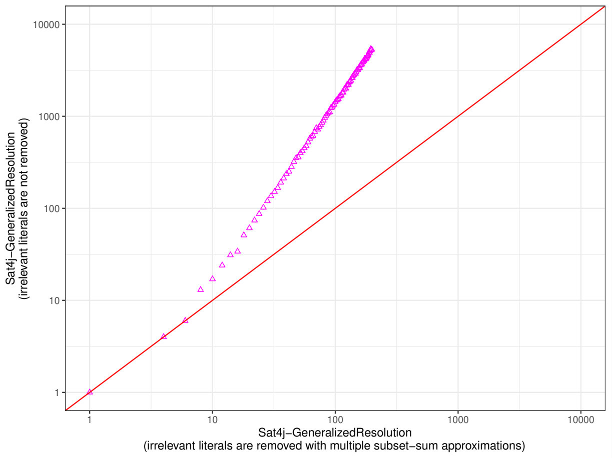
In practice, we use a heuristic to decide which strategy to apply, as none of them is better than the other

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)

Focus on the Vertex-Cover Family: Experimental Results



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)

Focus on the Vertex-Cover Family: *Sat4j*'s Behavior

When given an instance of this family, the first constraint learned by *Sat4j* has the form

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

Focus on the Vertex-Cover Family: *Sat4j*'s Behavior

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

Focus on the Vertex-Cover Family: ***Sat4j***'s Behavior

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*

Focus on the Vertex-Cover Family: ***Sat4j*'s Behavior**

When given an instance of this family, the first constraint learned by *Sat4j* has the form

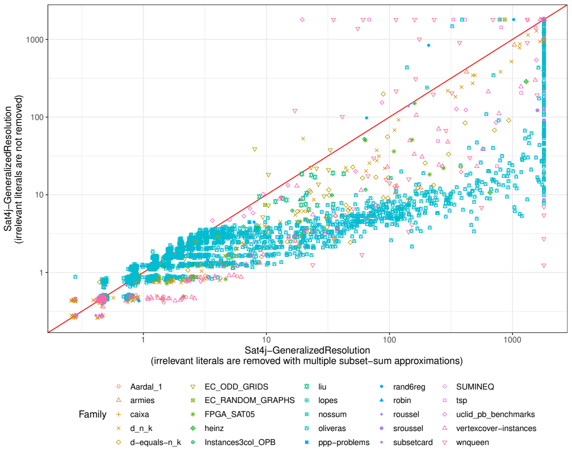
$$nx + x_1 + \ldots + x_{n-1} \geq n$$

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)

Weakening Ineffective Literals

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

Weakening Ineffective Literals

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

$$\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6$$

$$2a + b + \boxed{c} + f \geq 2$$

Weakening Ineffective Literals

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

$$\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6$$

$$2a + b + \boxed{c} + f \geq 2$$

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 literal may be ineffective

$$\boxed{3\bar{a}} + 3\bar{b} + c + \boxed{d} + \boxed{e} \geq 6$$

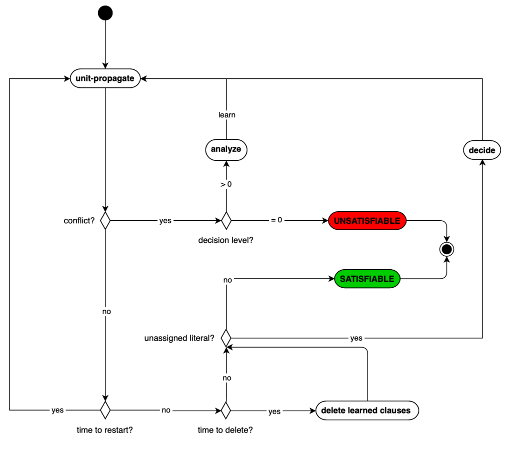
$$2a + b + \boxed{c} + f \geq 2$$

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

CDCL Architecture Recap



Overview of the CDCL Algorithm

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

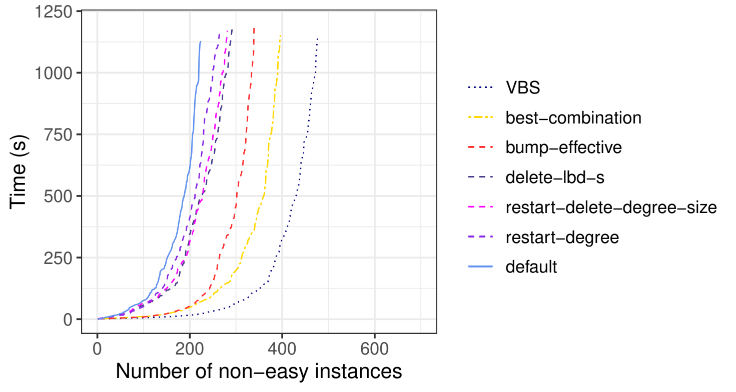
It is well known that, in addition to conflict analysis, several features of SAT solvers are crucial for solving problems efficiently, such as:

- branching heuristic
- learned constraint deletion strategy
- restart policy

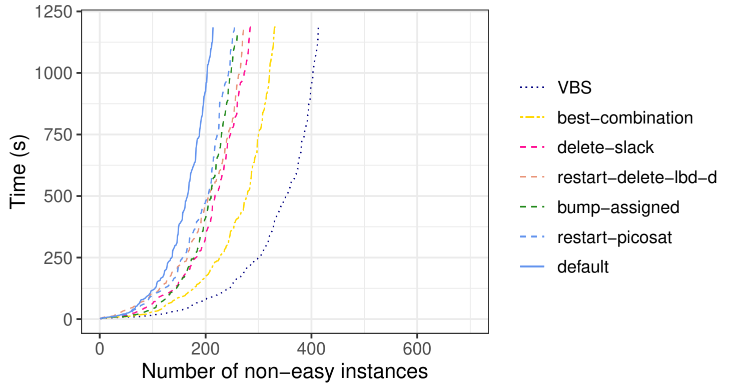
These features are mostly reused as is by current PB solvers, without taking into account the particular properties of PB constraints

Our main finding is that considering the size of the coefficients and the current partial assignment allows to significantly improve the solver

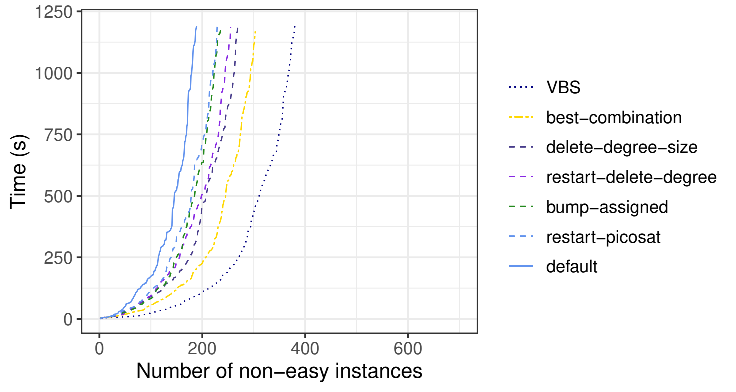
Experimental Results (Sat4j-GeneralizedResolution)



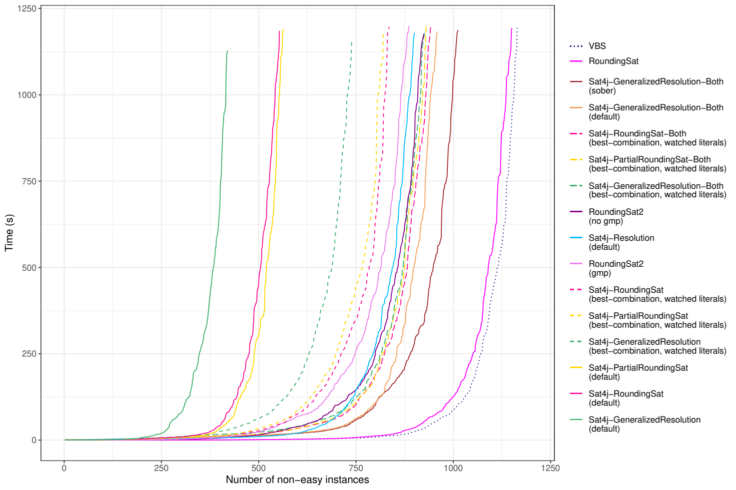
Experimental Results (Sat4j-RoundingSat)



Experimental Results (Sat4j-PartialRoundingSat)



Comparison of Sat4j with RoundingSat



Deeper Dive into Sat4j

Leveraging Properties of PB Constraints for Fine Tuning Sat4j

Let us consider again a confict analysis

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

$$6a(?@?) + 3b(?@?) + 3c(?@?) + 3d(?@?) + 3f(?@?) \geq 9$$

Leveraging Properties of PB Constraints for Fine Tuning Sat4j

Let us consider again a confict analysis

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

$$6a(?@?) + 3b(1@1) + 3c(?@?) + 3d(?@?) + 3f(?@?) \geq 9$$

Leveraging Properties of PB Constraints for Fine Tuning Sat4j

Let us consider again a confict analysis

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

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

Leveraging Properties of PB Constraints for Fine Tuning Sat4j

Let us consider again a confict analysis

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

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

Leveraging Properties of PB Constraints for Fine Tuning Sat4j

Let us consider again a confict analysis

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

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

Leveraging Properties of PB Constraints for Fine Tuning Sat4j

Let us consider again a confict 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:

$$\frac{3\bar{a} + 3\bar{f} + d + e \geq 5 \qquad 6a + 3b + 3c + 3d + 3f \geq 9}{3a(0@3) + 3b(1@1) + 3c(0@2) + 2\bar{d}(1@3) + e(?@?) \geq 7}$$

Leveraging Properties of PB Constraints for Fine Tuning Sat4j

Let us consider again a confict 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:

$$\frac{3\bar{a} + 3\bar{f} + d + e \geq 5 \qquad 6a + 3b + 3c + 3d + 3f \geq 9}{3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) \geq 7}$$

Leveraging Properties of PB Constraints for Fine Tuning Sat4j

Let us consider again a confict 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:

$$\frac{3\bar{a} + 3\bar{f} + d + e \geq 5 \qquad 6a + 3b + 3c + 3d + 3f \geq 9}{3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) \geq 7}$$

The PB constraints involved in this conflict analysis have very different properties compared to clauses!

(E)VSIDS for Making Decisions: Classical Implementation

All variables encountered during conflict analysis are bumped

(E)VSIDS for Making Decisions: Classical Implementation

All variables encountered during conflict analysis are bumped

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

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

(E)VSIDS for Making Decisions: Classical Implementation

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

(E)VSIDS for Making Decisions: Coefficients

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:

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

(E)VSIDS for Making Decisions: Coefficients

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:

(E)VSIDS for Making Decisions: Coefficients

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)

(E)VSIDS for Making Decisions: Coefficients

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)

(E)VSIDS for Making Decisions: Coefficients

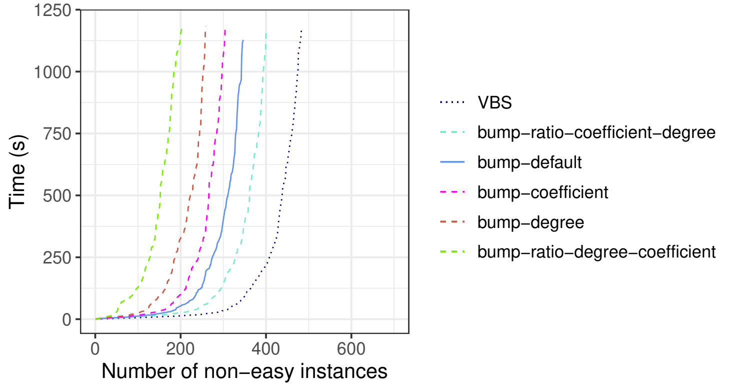
A first approach for adapting VSIDS to PB constraints has been proposed in (Dixon, 2004), but it only takes into account the original cardinality constraints, and thus not the reason we have here:

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

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)
- `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



(E)VSIDS for Making Decisions: Assignments

Observe also that some literals are unassigned in the reason:

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

(E)VSIDS for Making Decisions: Assignments

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

(E)VSIDS for Making Decisions: Assignments

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 assignement into account with 3 new strategies:

(E)VSIDS for Making Decisions: Assignments

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 assignement into account with 3 new strategies:

- `bump-assigned`: the score of each assigned variable is incremented (a, f and d)

(E)VSIDS for Making Decisions: Assignments

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 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)

(E)VSIDS for Making Decisions: Assignments

Observe also that some literals are unassigned in the reason:

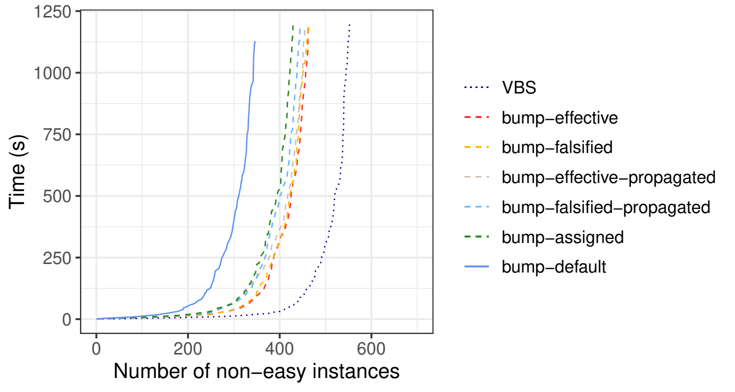
$$3\bar{a} + 3\bar{f} + d + e \geq 5$$

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 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)
- `bump-effective`: the score of each effective variable is incremented (f and d)

(E)VSIDS for Making Decisions: Experiments



Quality of Learned Constraints: Classical Implementations

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

Quality of Learned Constraints: Classical Implementations

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

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

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

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:

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

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.

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

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

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

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

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

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!

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!

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

Quality of Learned Constraint: Deletion

Deleting constraints is required by SAT solvers to limit the memory usage and to prevent unit propagation from slowing down

Quality of Learned Constraint: Deletion

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

Quality of Learned Constraint: Deletion

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

Quality of Learned Constraint: Deletion

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

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

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

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

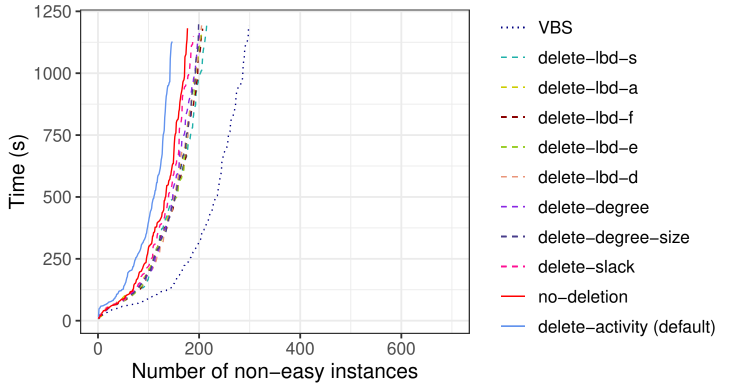
Following *Glucose*'s approach (Audemard and Simon, 2012), we consider adaptive restarts based on the quality of recently learned constraints

Whenever the most recent constraints are of poor quality compared to all the others, a restart is performed

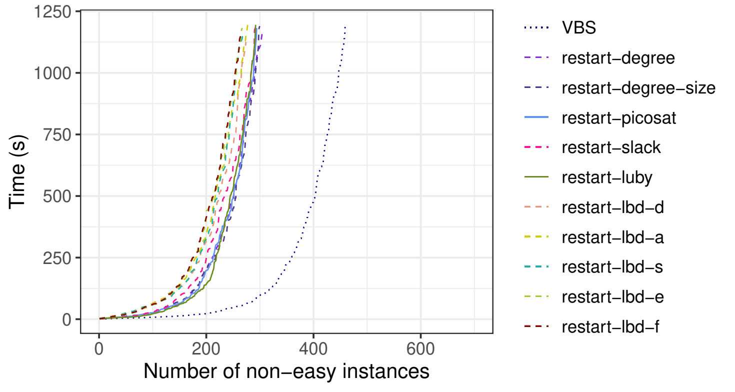
We thus introduce the following restart strategies, based on the different quality measures we presented

- `restart-degree`
- `restart-degree-size`
- `restart-lbd-a`
- `restart-lbd-s`
- `restart-lbd-d`
- `restart-lbd-f`
- `restart-lbd-e`

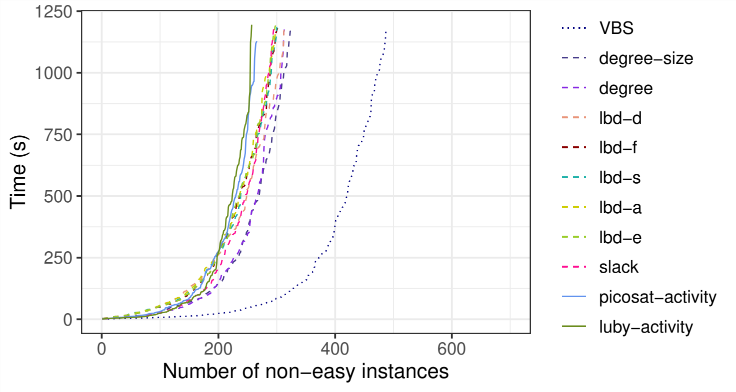
Quality of Learned Constraints: Experiments (Deletion)



Quality of Learned Constraints: Experiments (Restarts)



Quality of Learned Constraints: Experiments (Del. + Restarts)



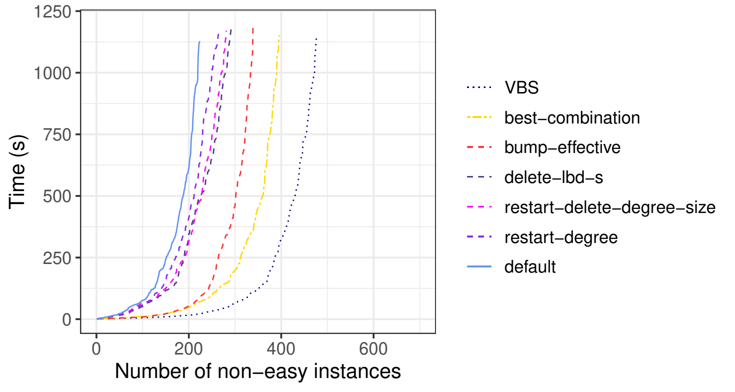
Putting Things Together: Brief Recap

For the moment, we have observed that the following individual strategies have the most important impact on the performance of the solver:

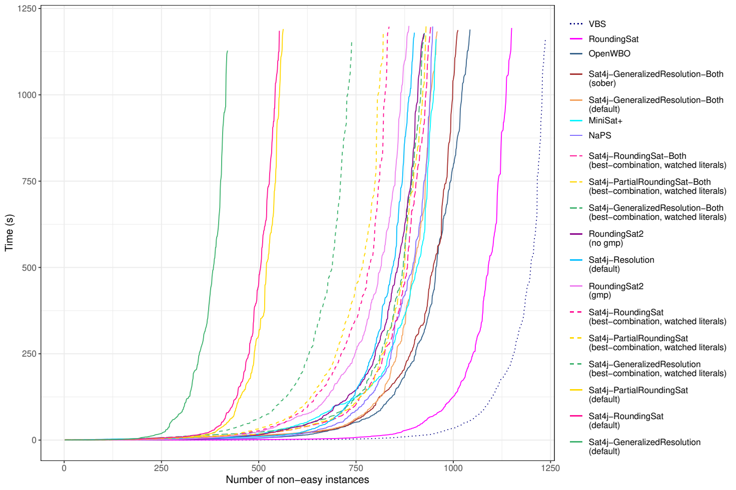
- `bump-effective`
- `delete-lbd-s`
- `restart-degree`

Let us combine all these strategies!

Putting Things Together: Experiments



Comparison of Sat4j with State-of-the-Art Solvers



Recovering Cardinality Constraints

Semantic cardinality detection

- Theory tells us that Cutting Planes should work on CNF
- Current implementations do not
- Can we find a way to help PB solvers work on CNF?
- Caution: we need a general process, not one dedicated to a given problem or constraint

Example of a Human Proof ($n = 3$) again

$$(1)\ p_{11} + p_{12} \geq 1 \ (2)\ p_{21} + p_{22} \geq 1 \ (3)\ p_{31} + p_{32} \geq 1$$

$$(4a)\ \overline{p_{11}} + \overline{p_{21}} \geq 1 \ (4b)\ \overline{p_{11}} + \overline{p_{31}} \geq 1 \ (4c)\ \overline{p_{21}} + \overline{p_{31}} \geq 1$$

$$(5a)\ \overline{p_{12}} + \overline{p_{22}} \geq 1 \ (5b)\ \overline{p_{12}} + \overline{p_{32}} \geq 1 \ (5c)\ \overline{p_{22}} + \overline{p_{32}} \geq 1$$

Example of a Human Proof ($n = 3$) again

$$(1)\ p_{11} + p_{12} \geq 1\ (2)\ p_{21} + p_{22} \geq 1\ (3)\ p_{31} + p_{32} \geq 1$$

$$(4a)\ \overline{p_{11}} + \overline{p_{21}} \geq 1\ (4b)\ \overline{p_{11}} + \overline{p_{31}} \geq 1\ (4c)\ \overline{p_{21}} + \overline{p_{31}} \geq 1$$

$$(5a)\ \overline{p_{12}} + \overline{p_{22}} \geq 1\ (5b)\ \overline{p_{12}} + \overline{p_{32}} \geq 1\ (5c)\ \overline{p_{22}} + \overline{p_{32}} \geq 1$$

$$(4a) + (4b) + (4c) = (4d)\ 2 * \overline{p_{11}} + 2 * \overline{p_{21}} + 2 * \overline{p_{31}} \geq 3$$

$$(5a) + (5b) + (5c) = (5d)\ 2 * \overline{p_{12}} + 2 * \overline{p_{22}} + 2 * \overline{p_{32}} \geq 3$$

Example of a Human Proof ($n = 3$) again

$$(1)\ p_{11} + p_{12} \geq 1 \ (2)\ p_{21} + p_{22} \geq 1 \ (3)\ p_{31} + p_{32} \geq 1$$

$$(4a)\ \overline{p_{11}} + \overline{p_{21}} \geq 1 \ (4b)\ \overline{p_{11}} + \overline{p_{31}} \geq 1 \ (4c)\ \overline{p_{21}} + \overline{p_{31}} \geq 1$$

$$(5a)\ \overline{p_{12}} + \overline{p_{22}} \geq 1 \ (5b)\ \overline{p_{12}} + \overline{p_{32}} \geq 1 \ (5c)\ \overline{p_{22}} + \overline{p_{32}} \geq 1$$

$$(4a) + (4b) + (4c) = (4d)\ 2 * \overline{p_{11}} + 2 * \overline{p_{21}} + 2 * \overline{p_{31}} \geq 3$$

$$(5a) + (5b) + (5c) = (5d)\ 2 * \overline{p_{12}} + 2 * \overline{p_{22}} + 2 * \overline{p_{32}} \geq 3$$

$$(4d)/2 = (4)\ \overline{p_{11}} + \overline{p_{21}} + \overline{p_{31}} \geq 2$$

$$(5d)/2 = (5)\ \overline{p_{12}} + \overline{p_{22}} + \overline{p_{32}} \geq 2$$

Example of a Human Proof ($n = 3$) again

$$(1)\ p_{11} + p_{12} \geq 1 \ (2)\ p_{21} + p_{22} \geq 1 \ (3)\ p_{31} + p_{32} \geq 1$$

$$(4a)\ \overline{p_{11}} + \overline{p_{21}} \geq 1 \ (4b)\ \overline{p_{11}} + \overline{p_{31}} \geq 1 \ (4c)\ \overline{p_{21}} + \overline{p_{31}} \geq 1$$

$$(5a)\ \overline{p_{12}} + \overline{p_{22}} \geq 1 \ (5b)\ \overline{p_{12}} + \overline{p_{32}} \geq 1 \ (5c)\ \overline{p_{22}} + \overline{p_{32}} \geq 1$$

$$(4a) + (4b) + (4c) = (4d)\ 2 * \overline{p_{11}} + 2 * \overline{p_{21}} + 2 * \overline{p_{31}} \geq 3$$

$$(5a) + (5b) + (5c) = (5d)\ 2 * \overline{p_{12}} + 2 * \overline{p_{22}} + 2 * \overline{p_{32}} \geq 3$$

$$(4d)/2 = (4)\ \overline{p_{11}} + \overline{p_{21}} + \overline{p_{31}} \geq 2$$

$$(5d)/2 = (5)\ \overline{p_{12}} + \overline{p_{22}} + \overline{p_{32}} \geq 2$$

$$(1) + (2) + (3) + (4) = (6)\ p_{12} + p_{22} + p_{32} \geq 2$$

Example of a Human Proof ($n = 3$) again

$$(1)\ p_{11} + p_{12} \geq 1\ (2)\ p_{21} + p_{22} \geq 1\ (3)\ p_{31} + p_{32} \geq 1$$

$$(4a)\ \overline{p_{11}} + \overline{p_{21}} \geq 1\ (4b)\ \overline{p_{11}} + \overline{p_{31}} \geq 1\ (4c)\ \overline{p_{21}} + \overline{p_{31}} \geq 1$$

$$(5a)\ \overline{p_{12}} + \overline{p_{22}} \geq 1\ (5b)\ \overline{p_{12}} + \overline{p_{32}} \geq 1\ (5c)\ \overline{p_{22}} + \overline{p_{32}} \geq 1$$

$$(4a) + (4b) + (4c) = (4d)\ 2 * \overline{p_{11}} + 2 * \overline{p_{21}} + 2 * \overline{p_{31}} \geq 3$$

$$(5a) + (5b) + (5c) = (5d)\ 2 * \overline{p_{12}} + 2 * \overline{p_{22}} + 2 * \overline{p_{32}} \geq 3$$

$$(4d)/2 = (4)\ \overline{p_{11}} + \overline{p_{21}} + \overline{p_{31}} \geq 2$$

$$(5d)/2 = (5)\ \overline{p_{12}} + \overline{p_{22}} + \overline{p_{32}} \geq 2$$

$$(1) + (2) + (3) + (4) = (6)\ p_{12} + p_{22} + p_{32} \geq 2$$

$$(5) + (6) = (7)\ 3 \geq 4$$

PHP: cardinality constraints *vs.* clauses

PHP: inconsistency proof computation time

- pigeons-100-hole.cnf:
  - resolution $\rightarrow$ timeout (900s)
  - generalized resolution (Hooker, 1988) $\rightarrow$ timeout (900s)
- pigeons-100-hole.opb:
  - resolution $\rightarrow$ timeout (900s)
  - generalized resolution (Hooker, 1988) $\rightarrow < 1s$.

Representation of constraints matters

Cardinality constraints *vs.* clauses

- pros :
  - a cardinality constraint may replace an exponential number of clauses or prevent the use of auxiliary variables
  - allow to use strong proof systems (generalized resolution, cutting planes)
- cons:
  - difficult detection : many encoding exist to translate cardinality constraints into CNF
  - deriving cardinality constraints using Cutting Planes proof system does not fit well with CDCL architecture

Some known encodings

Short list of known encodings :

- Pairwise encoding (Cook et al., 1987)
- Nested encoding
- Two product encoding (Chen, 2010)
- Sequential encoding (Sinz, 2005)
- Commander encoding (Frisch and Giannaros, 2010)
- Ladder encoding (Gent and Nightingale, 2004)
- Adder encoding (Eén and Sörensson, 2006)
- Cardinality Networks (Asín et al., 2009)
- ...

Syntactic *vs.* semantic detection

- Syntactic detection:
  - need of an *ad hoc* algorithm for each {encoding,k}
- Our semantic detection (Biere et al., 2014)
  - based on unit propagation
  - adapted to any encoding preserving arc-consistency
  - may potentially detect constraints that were not known at encoding time
  - detection may be altered by auxiliary variables
- More recent: inprocessing detection (Elffers and Nordström, 2020)

Semantic detection of at-most-k constraint

detecting a cardinality constraint in a semantic way:

1. select a clause of size n, and translate it into an AtMost-k of degree $n-1$:
$$\bigvee_{i=1}^{n} x_i \ \equiv \ \sum_{i=1}^{n} \neg x_i \leq n-1$$
2. look for literals m_j to extend this constraint:
$$\sum_{i=1}^{n} (\neg x_i) + m_1 + ... + m_p \leq n-1$$

Semantic detection of at-most-k constraint: example

formula :

$$\begin{aligned} \neg x_1 &\vee \neg x_2 \\ \neg x_1 &\vee \neg x_4 \\ x_4 &\vee \neg x_3 \\ \neg x_2 &\vee \neg x_5 \\ x_5 &\vee \neg x_3 \end{aligned}$$

detection of $\sum_{i=1}^{3} x_i \leq 1$

Semantic detection of at-most-k constraint: example

$\neg x_1 \vee \neg x_2$

formula :

$\neg x_1 \vee \neg x_2$

$\neg x_1 \vee \neg x_4$

$x_4 \vee \neg x_3$

$\neg x_2 \vee \neg x_5$

$x_5 \vee \neg x_3$

detection of $\sum_{i=1}^{3} x_i \leq 1$

Semantic detection of at-most-k constraint: example

formula :

$\neg x_1 \vee \neg x_2$

$\neg x_1 \vee \neg x_4$

$x_4 \vee \neg x_3$

$\neg x_2 \vee \neg x_5$

$x_5 \vee \neg x_3$

$$\neg x_1 \vee \neg x_2$$
$$\equiv$$
$$x_1 + x_2 \leq 1$$

detection of $\sum_{i=1}^{3} x_i \leq 1$

Semantic detection of at-most-k constraint: example

formula :

$$\neg x_1 \vee \neg x_2$$
$$\neg x_1 \vee \neg x_4$$
$$x_4 \vee \neg x_3$$
$$\neg x_2 \vee \neg x_5$$
$$x_5 \vee \neg x_3$$

$$\neg x_1 \vee \neg x_2 \equiv x_1 + x_2 \leq 1$$

detection of $\sum_{i=1}^{3} x_i \leq 1$

Semantic detection of at-most-k constraint: example

formula :

$\neg x_1 \vee \neg x_2$

$\neg x_1 \vee \neg x_4$

$x_4 \vee \neg x_3$

$\neg x_2 \vee \neg x_5$

$x_5 \vee \neg x_3$

$$\neg x_1 \vee \neg x_2$$
$$\equiv$$
$$x_1 + x_2 \leq 1$$

$$\text{PU}(x_1) = \{\ x_1, \neg x_2, \neg x_3, \neg x_4 \quad \}$$

$$\text{detection of } \sum_{i=1}^{3} x_i \leq 1$$

Semantic detection of at-most-k constraint: example

formula :

$\neg x_1 \vee \neg x_2$

$\neg x_1 \vee \neg x_4$

$x_4 \vee \neg x_3$

$\neg x_2 \vee \neg x_5$

$x_5 \vee \neg x_3$

$$\neg x_1 \vee \neg x_2$$
$$\equiv$$
$$x_1 + x_2 \leq 1$$

$$\mathrm{PU}(x_1) = \{\ x_1, \neg x_2, \neg x_3, \neg x_4 \quad \}$$
$$\mathrm{PU}(x_2) = \{\neg x_1,\ x_2, \neg x_3, \quad \neg x_5\}$$

detection of $\sum_{i=1}^{3} x_i \leq 1$

Semantic detection of at-most-k constraint: example

formula :

$\neg x_1 \vee \neg x_2$

$\neg x_1 \vee \neg x_4$

$x_4 \vee \neg x_3$

$\neg x_2 \vee \neg x_5$

$x_5 \vee \neg x_3$

$$\neg x_1 \vee \neg x_2 \equiv x_1 + x_2 \leq 1$$

$$\mathrm{PU}(x_1) = \{ x_1, \neg x_2, \neg x_3, \neg x_4 \}$$
$$\mathrm{PU}(x_2) = \{ \neg x_1, x_2, \neg x_3, \neg x_5 \}$$

$$\gamma = \{ \neg x_3 \}$$

detection of $\sum_{i=1}^{3} x_i \leq 1$

Semantic detection of at-most-k constraint: example

formula :

$\neg x_1 \vee \neg x_2$

$\neg x_1 \vee \neg x_4$

$x_4 \vee \neg x_3$

$\neg x_2 \vee \neg x_5$

$x_5 \vee \neg x_3$

$$\neg x_1 \vee \neg x_2$$
$$\equiv$$
$$x_1 + x_2 \leq 1$$

$$\mathrm{PU}(x_1) = \{\ x_1, \neg x_2, \neg x_3, \neg x_4\ \}$$
$$\mathrm{PU}(x_2) = \{\neg x_1,\ x_2, \neg x_3,\ \neg x_5\}$$

$$\gamma = \{\ \neg x_3\ \}$$

$$x_1 + x_2 + x_3 \leq 1$$

detection of $\sum_{i=1}^{3} x_i \leq 1$

Cardinality constraint extension

Cardinality constraint extension:

1. let $\alpha = \sum_{i=1}^{n} x_i \leq k$
2. initialization of the propagation set $\gamma = \{v_i, \neg v_i \mid v \in \mathrm{PS}\}$
3. for each subset of k literals x_i, we compute the unit propagation set δ, and we refine the propagation set:

$$\gamma \leftarrow \gamma \cap \delta$$

4. if there exists $m \in \gamma$, then $\alpha = \sum_{i=1}^{n} x_i + \neg m \leq k$ and goto 2

Experimental evaluation

- aim of the experiments: check that detected constraints help a generalized resolution based solver
- solvers:
  - Lingeling: able to detect pairwise encoding
  - Synt.+Sat4jCP, Sem.+Sat4jCP, Sat4jCP w/o preprocessing
  - SBSAT: able to detection cardinality constraints *via* compilation steps
- Intel Xeon@2.66GHz, 32Go RAM, timeouts=900s

Results

Influence of detected constraints for some encodings of PHP:

Preprocessing Solver	#inst.	Lingeling Lingeling	Synt.(Riss) Sat4jCP	Sem.(Riss) Sat4jCP	∅ SBSAT	∅ Sat4jCP
Pairwise	14	14 (3s)	13 (244s)	14 (583s)	6 (0s)	1 (196s)
Binary	14	3 (398s)	2 (554s)	7 (6s)	6 (7s)	2 (645s)
Sequential	14	0 (0s)	14 (50s)	14 (40s)	10 (6s)	1 (37s)
Product	14	0 (0s)	14 (544s)	11 (69s)	6 (25s)	2 (346s)
Commander	14	1 (3s)	7 (0s)	14 (40s)	9 (187s)	1 (684s)
Ladder	14	0 (0s)	11 (505s)	11 (1229s)	12 (26s)	1 (36s)

solved instances (computation time of solved instances)

Results

Influence of detected constraints for *balanced block design* instances:

Preprocessing Solver	#inst.	Lingeling Lingeling	Synt.(Riss) Sat4jCP	Sem.(Riss) Sat4jCP	∅ SBSAT	∅ Sat4jCP
Sgen unsat	13	0 (0s)	13 (0s)	13 (0s)	9 (614s)	4 (126s)
Fixed bandwidth	23	2 (341s)	23 (0s)	23 (0s)	23 (1s)	13 (1800s)
Rand. orderings	168	16 (897s)	168 (7s)	168 (8s)	99 (2798s)	69 (3541s)
Rand. 4-reg.	126	6 (1626s)	126 (4s)	126 (5s)	84 (2172s)	49 (3754s)

solved instances (computation time of solved instances)

Further Results

- “crossed” constraints: Sudoku grid
  - Sudoku 9x9: syntactic preprocessing detects 300/324 constraints, semantic one detects 324/324 constraints
  - Sudoku 16x16: syntactic preprocessing detects 980/1024 constraints, semantic one detects 1024/1024 constraints

- Challenge benchmark of (Van Gelder and Spence, 2010) (clasp unable to solve within 24h): solved within a second thanks to semantic preprocessing (AtMost-3 constraints inside)

Some Open Questions about PB Conflict Analysis

#1 - Improving the Backjump Level

The backjump level computed based on the 1-UIP scheme is not optimal in general in PB solvers

#1 - Improving the Backjump Level

The backjump level computed based on the 1-UIP scheme is not optimal in general in PB solvers

$$
\begin{array}{ll}
 & 8a(0@1) + 8b(1@3) + 6c(0@2) + 6d(0@3) + 4e(1@3) + \underline{2f(0@4)} + \underline{2g(0@4)} + 2m(0@4) \geq 16 \\
\boxplus & 3i(0@2) + 3j(0@4) + \underline{2\bar{f}(1@4)} + \underline{2\bar{g}(1@4)} + h(1@4) \geq 5 \\
\hline
\end{array}
$$

#1 - Improving the Backjump Level

The backjump level computed based on the 1-UIP scheme is not optimal in general in PB solvers

$$
\begin{array}{ll}
 & 8a(0@1) + 8b(1@3) + 6c(0@2) + 6d(0@3) + 4e(1@3) + \underline{2f(0@4)} + \underline{2g(0@4)} + 2m(0@4) \geq 16 \\
\boxplus & 3i(0@2) + 3j(0@4) + \underline{2\bar{f}(1@4)} + \underline{2\bar{g}(1@4)} + h(1@4) \geq 5 \\
\hline
= & 8a(0@1) + 8b(1@3) + 6c(0@2) + 6d(0@3) + 4e(1@3) + 2m(0@4) + 3i(0@2) + 3j(0@4) + h(1@4) \geq 17
\end{array}
$$

#1 - Improving the Backjump Level

The backjump level computed based on the 1-UIP scheme is not optimal in general in PB solvers

$$
\begin{array}{ll}
 & 8a(0@1) + 8b(1@3) + 6c(0@2) + 6d(0@3) + 4e(1@3) + \underline{2f(0@4)} + \underline{2g(0@4)} + 2m(0@4) \geq 16 \\
\boxplus & 3i(0@2) + 3j(0@4) + \underline{2\bar{f}(1@4)} + \underline{2\bar{g}(1@4)} + h(1@4) \geq 5 \\
\hline
= & 8a(0@1) + 8b(1@3) + \underline{6c(0@2)} + \underline{6d(0@3)} + 4e(1@3) + 2m(0@4) + 3i(0@2) + \underline{3j(0@4)} + h(1@4) \geq 17 \\
\boxplus & \underline{6\bar{c}(1@2)} + \underline{6\bar{d}(1@3)} + \underline{3\bar{j}(1@4)} + 3k(0@4) + 3l(0@3) \geq 15 \\
\hline
\end{array}
$$

#1 - Improving the Backjump Level

The backjump level computed based on the 1-UIP scheme is not optimal in general in PB solvers

$$
\begin{array}{ll}
 & 8a(0@1) + 8b(1@3) + 6c(0@2) + 6d(0@3) + 4e(1@3) + \underline{2f(0@4)} + \underline{2g(0@4)} + 2m(0@4) \geq 16 \\
\boxplus & 3i(0@2) + 3j(0@4) + \underline{2\bar{f}(1@4)} + \underline{2\bar{g}(1@4)} + h(1@4) \geq 5 \\
\hline
= & 8a(0@1) + 8b(1@3) + \underline{6c(0@2)} + \underline{6d(0@3)} + 4e(1@3) + 2m(0@4) + 3i(0@2) + \underline{3j(0@4)} + h(1@4) \geq 17 \\
\boxplus & \underline{6\bar{c}(1@2)} + \underline{6\bar{d}(1@3)} + \underline{3\bar{j}(1@4)} + 3k(0@4) + 3l(0@3) \geq 15 \\
\hline
= & 8a(0@1) + 8b(1@3) + 4e(1@3) + 2m(0@4) + 3i(0@2) + 3k(0@4) + 3l(0@3) + h(1@4) \geq 17
\end{array}
$$

#1 - Improving the Backjump Level

The backjump level computed based on the 1-UIP scheme is not optimal in general in PB solvers

$$\begin{array}{ll}
 & 8a(0@1) + 8b(1@3) + 6c(0@2) + 6d(0@3) + 4e(1@3) + \underline{2f(0@4)} + \underline{2g(0@4)} + 2m(0@4) \geq 16 \\
\boxplus & 3i(0@2) + 3j(0@4) + \underline{2\bar{f}(1@4)} + \underline{2\bar{g}(1@4)} + h(1@4) \geq 5 \\
\hline
= & 8a(0@1) + 8b(1@3) + \underline{6c(0@2)} + \underline{6d(0@3)} + 4e(1@3) + 2m(0@4) + 3i(0@2) + \underline{3j(0@4)} + h(1@4) \geq 17 \\
\boxplus & \underline{6\bar{c}(1@2)} + \underline{6\bar{d}(1@3)} + \underline{3\bar{j}(1@4)} + 3k(0@4) + 3l(0@3) \geq 15 \\
\hline
= & 8a(0@1) + 8b(1@3) + 4e(1@3) + 2m(0@4) + 3i(0@2) + 3k(0@4) + 3l(0@3) + h(1@4) \geq 17
\end{array}$$

Continuing conflict analysis after the production of an assertive constraint may allow to improve the backjump level, but it is not always true

#2 - Improving Conflict Detection

Consider these 4 constraints:

$$\chi_1 \equiv a + \bar{b} + \bar{c} \geq 2$$
$$\chi_2 \equiv 3b + 3d + e + f \geq 4$$
$$\chi_3 \equiv 2c + \bar{e} + \bar{f} \geq 2$$
$$\chi_4 \equiv b + \bar{d} + e + f \geq 1$$

#2 - Improving Conflict Detection

Consider these 4 constraints:

$$\chi_1 \equiv a + \bar{b} + \bar{c} \geq 2$$
$$\chi_2 \equiv 3b + 3d + e + f \geq 4$$
$$\chi_3 \equiv 2c + \bar{e} + \bar{f} \geq 2$$
$$\chi_4 \equiv b + \bar{d} + e + f \geq 1$$

If we choose χ_4 as conflict to analyze, we have:

$$\cfrac{\cfrac{\cfrac{\chi_4 \qquad \chi_3}{b + c + \bar{d} \geq 1} \qquad \chi_2}{4b + 3c + e + f \geq 4} \qquad \chi_1}{3a + b + e + f \geq 4}$$

#2 - Improving Conflict Detection

Consider these 4 constraints:

$$\chi_1 \equiv a + \bar{b} + \bar{c} \geq 2$$
$$\chi_2 \equiv 3b + 3d + e + f \geq 4$$
$$\chi_3 \equiv 2c + \bar{e} + \bar{f} \geq 2$$
$$\chi_4 \equiv b + \bar{d} + e + f \geq 1$$

If we choose χ_2 as conflict to analyze, we have:

$$\dfrac{\dfrac{\chi_2 \qquad \chi_3}{3b + 3d + 2c \geq 4} \qquad \chi_1}{3d + 2a + b \geq 4}$$

#2 - Improving Conflict Detection

Consider these 4 constraints:

$$\chi_1 \equiv a + \bar{b} + \bar{c} \geq 2$$
$$\chi_2 \equiv 3b + 3d + e + f \geq 4$$
$$\chi_3 \equiv 2c + \bar{e} + \bar{f} \geq 2$$
$$\chi_4 \equiv b + \bar{d} + e + f \geq 1$$

If we choose χ_2 as conflict to analyze, we have:

$$\cfrac{\cfrac{\chi_2 \qquad \chi_3}{3b + 3d + 2c \geq 4} \qquad \chi_1}{3d + 2a + b \geq 4}$$

None of the two constraints is stronger than the other

Another Perspective on Normalized Forms

Why do we need a *normalized* form?

Recall that all constraints we consider are of the form:

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

Why do we need a *normalized* form?

Recall that all constraints we consider are of the form:

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

This normalized form is needed to apply the cancellation rule, because we need literals to be in the same side of the relational operator:

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\overline{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

Why do we need a *normalized* form?

Recall that all constraints we consider are of the form:

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

This normalized form is needed to apply the cancellation rule, because we need literals to be in the same side of the relational operator:

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\overline{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

So, why don't we use a $\leq$-based normalized form?

Why do we prefer $\geq$ over $\leq$?

The main advantage of using $\geq$ is that it allows to easily represent clauses

$$\ell_1 \vee \ldots \vee \ell_n \equiv \ell_1 + \ldots + \ell_n \geq 1$$

Why do we prefer $\geq$ over $\leq$?

The main advantage of using $\geq$ is that it allows to easily represent clauses

$$\ell_1 \vee \ldots \vee \ell_n \equiv \ell_1 + \ldots + \ell_n \geq 1$$

This allows to reuse many data-structures used to solve CNF formulae, such as watched literals for instance

Why do we prefer $\geq$ over $\leq$?

The main advantage of using $\geq$ is that it allows to easily represent clauses

$$\ell_1 \vee \ldots \vee \ell_n \equiv \ell_1 + \ldots + \ell_n \geq 1$$

This allows to reuse many data-structures used to solve CNF formulae, such as watched literals for instance

Here, the clause above would be written as follows in a $\leq$-based normalized form

$$\ell_1 \vee \ldots \vee \ell_n \equiv \bar{\ell}_1 + \ldots + \bar{\ell}_n \leq n - 1$$

and we would have needed to watched the satisfaction of the literals!

Why do we prefer $\geq$ over $\leq$?

The main advantage of using $\geq$ is that it allows to easily represent clauses

$$\ell_1 \vee \ldots \vee \ell_n \equiv \ell_1 + \ldots + \ell_n \geq 1$$

This allows to reuse many data-structures used to solve CNF formulae, such as watched literals for instance

Here, the clause above would be written as follows in a $\leq$-based normalized form

$$\ell_1 \vee \ldots \vee \ell_n \equiv \bar{\ell}_1 + \ldots + \bar{\ell}_n \leq n - 1$$

and we would have needed to watched the satisfaction of the literals!

However, this normalized form is not always practical...

Normalized Form and PB Optimization

On optimization problems, for instance, the solver is asked to minimize an objective function of the form:

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

Normalized Form and PB Optimization

On optimization problems, for instance, the solver is asked to minimize an objective function of the form:

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

If the solver finds a model of the formula which makes the objective function equal to a value o, then the following constraint is added to the solver:

$$\sum_{i=1}^{n} \alpha_i \ell_i \leq o - 1 \equiv \sum_{i=1}^{n} \alpha_i \bar{\ell}_i \geq \sum_{i=1}^{n} \alpha_i - o + 1$$

Normalized Form and PB Optimization

On optimization problems, for instance, the solver is asked to minimize an objective function of the form:

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

If the solver finds a model of the formula which makes the objective function equal to a value o, then the following constraint is added to the solver:

$$\sum_{i=1}^{n} \alpha_i \ell_i \leq o - 1 \equiv \sum_{i=1}^{n} \alpha_i \bar{\ell}_i \geq \sum_{i=1}^{n} \alpha_i - o + 1$$

If the coefficients α_i are big, the degree of the constraint above will become even bigger, even if the value o is small

The Role of the Saturation Rule

Recall that PB solver can use the saturation rule

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta) \ell_i \geq \delta} \text{ (saturation)}$$

The Role of the Saturation Rule

Recall that PB solver can use the saturation rule

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta) \ell_i \geq \delta} \text{(saturation)}$$

This rule makes the degree of a constraint an upper bound of its coefficients, but if the degree becomes big, then it may allow coefficients to grow while applying consecutive cancellation step

Another Rule for $\leq$ Constraints?

While the cancellation rule can easily be adapted to $\leq$ constraints, saturation is only applicable on $\geq$-constraints...

Another Rule for $\leq$ Constraints?

While the cancellation rule can easily be adapted to $\leq$ constraints, saturation is only applicable on $\geq$-constraints...

If we want to use $\leq$ constraints, another rule is needed

Another Rule for $\leq$ Constraints?

While the cancellation rule can easily be adapted to $\leq$ constraints, saturation is only applicable on $\geq$-constraints...

If we want to use $\leq$ constraints, another rule is needed

To this end, we may borrow the following rule used in MIP preprocessing

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \leq \delta \qquad \sum_{i=1}^{n} \alpha_i \leq \delta}{\left(\alpha + \sum_{i=1}^{n} \alpha_i - \delta\right) \ell + \sum_{i=1}^{n} \alpha_i \ell_i \leq \sum_{i=1}^{n} \alpha_i}$$

Recap on Normalized Constraints

- To apply the cancellation rule, we need to have literals on the same side of the relational operator

Recap on Normalized Constraints

- To apply the cancellation rule, we need to have literals on the same side of the relational operator
- To easily represent clauses and extend SAT solvers, the $\geq$ normalization has been chosen

Recap on Normalized Constraints

- To apply the cancellation rule, we need to have literals on the same side of the relational operator
- To easily represent clauses and extend SAT solvers, the $\geq$ normalization has been chosen
- However, the $\leq$ is sometimes preferable, e.g., on optimization problems

Recap on Normalized Constraints

- To apply the cancellation rule, we need to have literals on the same side of the relational operator
- To easily represent clauses and extend SAT solvers, the $\geq$ normalization has been chosen
- However, the $\leq$ is sometimes preferable, e.g., on optimization problems
- All rules used by PB solvers are not always applicable to both representations, requiring to adapt the proof system to the representation

Recap on Normalized Constraints

- To apply the cancellation rule, we need to have literals on the same side of the relational operator
- To easily represent clauses and extend SAT solvers, the $\geq$ normalization has been chosen
- However, the $\leq$ is sometimes preferable, e.g., on optimization problems
- All rules used by PB solvers are not always applicable to both representations, requiring to adapt the proof system to the representation

Ideally, we could choose either the $\geq$ or the $\leq$ representation depending on which is "better", as this is done in encodings for instance

Conclusion and Perspectives

Conclusion

- 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

- 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

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

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

Deep Dive into CDCL Pseudo-Boolean Solvers

focusing on the implementation in *Sat4j*

Daniel Le Berre[1], Romain Wallon[2]

Februrary 23rd, 2021

[1]CRIL, Univ Artois & CNRS
[2]Laboratoire d'Informatique de l'X (LIX), École Polytechnique

References i

Asín, R., Nieuwenhuis, R., Oliveras, A., and Rodríguez-Carbonell, E. (2009). Cardinality networks and their applications. In Kullmann, O., editor, *SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 167–180. Springer.

Audemard, G. and Simon, L. (2009). Predicting Learnt Clauses Quality in Modern SAT Solvers. In *Proceedings of IJCAI'09*, pages 399–404.

Audemard, G. and Simon, L. (2012). Refining restarts strategies for sat and unsat formulae. pages 118–126.

Biere, A., Le Berre, D., Lonca, E., and Manthey, N. (2014). Detecting cardinality constraints in CNF. In *Theory and Applications of Satisfiability Testing*, pages 285–301.

Chai, D. and Kuehlmann, A. (2003). A fast pseudo-boolean constraint solver. In *Proceedings of the 40th Design Automation Conference, DAC 2003, Anaheim, CA, USA, June 2-6, 2003*, pages 830–835. ACM.

Chen, J.-C. (2010). A new sat encoding of the at-most-one constraint. In *In Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation*.

Cook, W., Coullard, C., and Turán, G. (1987). On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25 – 38.

Dixon, H. (2004). *Automating Pseudo-boolean Inference Within a DPLL Framework*. PhD thesis, Eugene, OR, USA. AAI3153782.

Dixon, H. E. and Ginsberg, M. L. (2002). Inference methods for a pseudo-boolean satisfiability solver. In *AAAI'02*, pages 635–640.

Eén, N. and Sörensson, N. (2004). An extensible sat-solver. In *Theory and Applications of Satisfiability Testing*, pages 502–518.

Eén, N. and Sörensson, N. (2006). Translating pseudo-boolean constraints into sat. *JSAT*, 2(1-4):1–26.

Elffers, J. and Nordström, J. (2018). Divide and conquer: Towards faster pseudo-boolean solving. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1291–1299.

Elffers, J. and Nordström, J. (2020). A cardinal improvement to pseudo-boolean solving. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1495–1503. AAAI Press.

Frisch, A. and Giannaros, P. (2010). Sat encodings of the at-most-k constraint: Some old, some new, some fast, some slow. In *Proceedings of the The 9th International Workshop on Constraint Modelling and Reformulation (ModRef 2010)*.

Gent, I. P. and Nightingale, P. (2004). A new encoding of alldifferent into sat. *Proc. 3rd International Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, pages 95–110.

Hooker, J. N. (1988). Generalized resolution and cutting planes. *Annals of Operations Research*, 12(1):217–239.

Le Berre, D., Marquis, P., Mengel, S., and Wallon, R. (2020a). On irrelevant literals in pseudo-boolean constraint learning. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1148–1154.

Le Berre, D., Marquis, P., and Wallon, R. (2020b). On weakening strategies for PB solvers. In Pulina, L. and Seidl, M., editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 322–331. Springer.

Le Berre, D. and Parrain, A. (2010). The SAT4J library, Release 2.2, System Description. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64.

Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., and Malik, S. (2001). Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 530–535, New York, NY, USA. ACM.

References vi

Silva, J. a. P. M. and Sakallah, K. A. (1996). GRASP – New Search Algorithm for Satisfiability. In *Proceedings of the 1996 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '96, pages 220–227, Washington, DC, USA. IEEE Computer Society.

Sinz, C. (2005). Towards an optimal cnf encoding of boolean cardinality constraints. In van Beek, P., editor, *CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer.

Van Gelder, A. and Spence, I. (2010). Zero-one designs produce small hard sat instances. In Strichman, O. and Szeider, S., editors, *SAT*, volume 6175 of *Lecture Notes in Computer Science*, pages 388–397. Springer.

Whittemore, J. and Sakallah, J. K. K. A. (2001). SATIRE: A new incremental satisfiability engine. In *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*, pages 542–545.