From clauses to pseudo-Boolean constraints in a Boolean solver

Daniel Le Berre

joint work with Armin Biere, Emmanuel Lonca, Pierre Marquis, Stefan Mengel, Norbert Manthey, Anne Parrain, Romain Wallon

CNRS, Université d'Artois, FRANCE

{leberre}@cril.univ-artois.fr

SAT+SMT school, IIT Bombay, India, 10 December 2019

Outline

Motivating example

Definitions and properties

Handling Pseudo-Boolean constraints instead of clauses

Conflict Driven “cutting planes” reasoning

A note about solving Optimization problems

Cardinality detection

On the limits of current PB solvers

Simple decision problem

Can we sit m researchers on $m-1$ seats?

Simple decision problem

Can we sit m researchers on $m - 1$ seats?

More precisely, we consider that

- Each researcher should have a seat
- Each seat cannot host more than a researcher

Can we answer that question with a SAT solver?

- Each Boolean variable x_{ij} denote that research i is seated on seat j
- "Each researcher should have a seat" translate to

$$\bigvee_{j=1}^{m-1} x_{ij}$$

  for each researcher i
- "Each seat cannot host more than a researcher"

$$\neg x_{ij} \vee \neg x_{kj}$$

  for each seat j, with $1 \leq i < k \leq m$

Can we answer that question with a SAT solver?

- Each Boolean variable x_{ij} denote that research i is seated on seat j
- "Each researcher should have a seat" translate to

$$\bigvee_{j=1}^{m-1} x_{ij}$$

for each researcher i
- "Each seat cannot host more than a researcher"

$$\neg x_{ij} \vee \neg x_{kj}$$

for each seat j, with $1 \leq i < k \leq m$

A modern CDCL SAT solver without specific counting features will not answer that question in reasonable time for $m > 20$

Can we answer that question with a PB solver?

- Each Boolean variable x_{ij} denote that research i is seated on seat j
- "Each researcher should have a seat" translate to

$$\sum_{j=1}^{m-1} x_{ij} \geq 1$$

  for each researcher i
- "Each seat cannot host more than a researcher"

$$\sum_{i=1}^{m} x_{ij} \leq 1$$

  for each seat j

Can we answer that question with a PB solver?

- Each Boolean variable x_{ij} denote that research i is seated on seat j
- "Each researcher should have a seat" translate to

$$\sum_{j=1}^{m-1} x_{ij} \geq 1$$

  for each researcher i
- "Each seat cannot host more than a researcher"

$$\sum_{i=1}^{m} x_{ij} \leq 1$$

  for each seat j

A modern PB solver based on resolution will not answer that question in reasonable time for $m > 20$

Can we answer that question with a PB solver?

- Each Boolean variable x_{ij} denote that research i is seated on seat j
- "Each researcher should have a seat" translate to

$$\sum_{j=1}^{m-1} x_{ij} \geq 1$$

  for each researcher i
- "Each seat cannot host more than a researcher"

$$\sum_{i=1}^{m} x_{ij} \leq 1$$

  for each seat j

A modern PB solver based on CuttingPlanes will answer that question in a matter of seconds (until the input is too large)

Principle of the human proof for m=3

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

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

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

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

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

Principle of the human proof for m=3

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

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

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

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

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

Principle of the human proof for m=3

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

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

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

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

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

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

Principle of the human proof for m=3

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

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

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

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

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

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

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

Human vs Solver, Complexity Theory vs Modeling

- 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 ($\frac{m*(m-1)}{2}$ vs $m-1$)
  - the solver can apply new inference rules (e.g. Cutting Plane) on higher abstraction constraints
- In theory, the input must be the same when talking about complexity
  - requires e.g. input in CNF for comparing resolution vs Cutting Plane
  - does not allow efficient encodings which rely on the addition of new variables
  - rely on "recovering" the cardinality constraints using domain knowledge

From clauses to cardinality constraints: principle

- Given binary clauses

$$\neg x_{ij} \vee \neg x_{kj}, 1 \leq i < k \leq m$$

  for each seat j
- Translate each binary clause $\neg x_{ij} \vee \neg x_{kj}$ into the equivalent constraint $\overline{x_{ij}} + \overline{x_{kj}} \geq 1$
- Sum up all those constraints related to seat j and three researchers u, v, w to obtain $2 * \overline{x_{uj}} + 2 * \overline{x_{vj}} + 2 * \overline{x_{kj}} \geq 3$
- Divide by 2 and round up the RHS to the nearest integer.
- Repeat with one more researcher on derived cardinalities

UNIVERSITÉ D'ARTOIS

From clauses to cardinality constraints: example

$$\neg x_{11} \vee \neg x_{21} \quad \neg x_{11} \vee \neg x_{31} \quad \neg x_{21} \vee \neg x_{31}$$

$$\overline{x_{11}} + \overline{x_{21}} \geq 1 \quad \overline{x_{11}} + \overline{x_{31}} \geq 1 \quad \overline{x_{21}} + \overline{x_{31}} \geq 1$$

$$2 * \overline{x_{11}} + 2 * \overline{x_{21}} + 2 * \overline{x_{31}} \geq 3$$

$$\overline{x_{11}} + \overline{x_{21}} + \overline{x_{31}} \geq 2$$

$$\equiv$$

$$x_{11} + x_{21} + x_{31} \leq 1$$

From clauses to cardinality constraints: example

$$\neg x_{11} \vee \neg x_{21} \quad \neg x_{11} \vee \neg x_{31} \quad \neg x_{11} \vee \neg x_{41}$$
$$\neg x_{21} \vee \neg x_{31} \quad \neg x_{21} \vee \neg x_{41} \quad \neg x_{31} \vee \neg x_{41}$$

$$\overline{x_{11}} + \overline{x_{21}} \geq 1 \quad \overline{x_{11}} + \overline{x_{31}} \geq 1 \quad \overline{x_{11}} + \overline{x_{41}} \geq 1$$
$$\overline{x_{21}} + \overline{x_{31}} \geq 1 \quad \overline{x_{21}} + \overline{x_{41}} \geq 1 \quad \overline{x_{31}} + \overline{x_{41}} \geq 1$$

$$\overline{x_{11}} + \overline{x_{21}} + \overline{x_{31}} \geq 2$$
$$\overline{x_{11}} + \overline{x_{21}} + \overline{x_{41}} \geq 2$$
$$\overline{x_{11}} + \overline{x_{31}} + \overline{x_{41}} \geq 2$$
$$\overline{x_{21}} + \overline{x_{31}} + \overline{x_{41}} \geq 2$$

$$\overline{x_{11}} + \overline{x_{21}} + \overline{x_{31}} + \overline{x_{41}} \geq 3$$
$$\equiv x_{11} + x_{21} + x_{31} + x_{41} \leq 1$$

Motivation

- CDCL SAT solvers are very efficient (cf yesterday's lectures by Mate)
- Clauses are of limited expressivity to express "counting" constraints
- CDCL proof system is resolution [PD11, AFT11]
- Resolution in CDCL is used during conflict analysis to produce new clauses
- This talk:
  - Consider more expressive constraints: pseudo-Boolean constraints
  - Change he conflict analysis procedure to produce pseudo-Boolean constraints
  - Using the "cutting planes" proof system?
  - Recovering cardinality constraints in practice

Outline of the talk

Motivating example

Definitions and properties

Handling Pseudo-Boolean constraints instead of clauses

Conflict Driven "cutting planes" reasoning

A note about solving Optimization problems

Cardinality detection

On the limits of current PB solvers

Outline

Motivating example

Definitions and properties

Handling Pseudo-Boolean constraints instead of clauses

Conflict Driven "cutting planes" reasoning

A note about solving Optimization problems

Cardinality detection

On the limits of current PB solvers

Linear Pseudo-Boolean constraints (LPB)

$$\sum_{i=1}^{n} a_i x_i \otimes k$$

- boolean variables x_i are integers taking their value in $\{0, 1\}$
  $(x_i \geq 0$ and $x_i \leq 1)$
- $\overline{x_i} = 1 - x$
- coefficients a_i and degree k are integer-valued constants
- $\otimes \in \{<, \leq, =, \geq, >\}$
  with $(< k \leftrightarrow \leq k - 1$ and $= k \leftrightarrow \leq k \wedge \geq k)$

Pseudo-Boolean decision problem: satisfying a set of LPB is NP-complete

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

LPB = Concise boolean function representation

- clauses are specific LPB:

$$\bigvee_{i=1}^{n} l_i \quad \equiv \quad \sum_{i=1}^{n} l_i \geq 1 \quad \equiv \quad \sum_{i=1}^{n} \overline{l_i} \leq n-1$$

$x_1 \vee x_2 \vee x_3$ translates into $x_1 + x_2 + x_3 \geq 1$
or $\overline{x_1} + \overline{x_2} + \overline{x_3} \leq 2$

- cardinality constraints at least/at most 2 out of $\{x_1, x_2, x_3\}$ translate into

$$x_1 + x_2 + x_3 \geq 2$$

$$x_1 + x_2 + x_3 \leq 2$$

- Knapsack constraint: $\sum w_i.x_i \leq W$
- Subset sum constraint: $\sum a_i.x_i = k$

Linear Pseudo Boolean constraints normalization

- remember that $x = 1 - \overline{x}$
- usual form : $\geq$ inequality and positive constants

$$-3x_1 + 4x_2 - 7x_3 + x_4 \leq -5$$

$$\equiv 3x_1 - 4x_2 + 7x_3 - x_4 \geq 5$$

$$\equiv 3x_1 + -4(1 - \overline{x_2}) + 7x_3 + -(1 - \overline{x_4}) \geq 5$$

$$\equiv 3x_1 + 4\overline{x_2} + 7x_3 + \overline{x_4} \geq 10$$

- note that

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq 1$$

is represented

$$\overline{x_1} + \overline{x_2} + \overline{x_3} + \overline{x_4} + \overline{x_5} \geq 4$$

Fun facts about PB constraints 1/3

- In a clause or a cardinality constraints, all literals are equivalent

  $$x_1 + x_2 + x_3 \geq 2$$

  can be equally satisfied by a pair of literals

- In a PB constraints, literals with the same coefficients are equivalent

  $$2x_1 + 2x_2 + x_3 + x_4 \geq 2$$

  x_1 and x_2 are equivalent, so are x_3 and x_4

Fun facts about PB constraints 2/3

- A clause can only propagate 1 literal

$$x_1$$

- A cardinality constraint can propagate only k literals

$$x_1 + x_2 + x_3 + \dots x_{k-1} + x_k \geq k$$

- A PB constraint can propagate between 1 and k literals

$$4x_1 + 4x_2 + x_3 + x_4 + x_5 \geq 9$$

x_1 and x_2 are necessarily true

UNIVERSITÉ D'ARTOIS

Fun facts about PB constraints 3/3

- PB constraints can sometimes be rewritten as a conjunction of simpler constraints

$$10x_1 + 4x_2 + 4x_3 + x_4 + x_5 + x_6 \geq 15$$

$$\equiv$$

$$x_1 \wedge (4x_2 + 4x_3 + x_4 + x_5 + x_6 \geq 5)$$

- A PB constraint may have *irrelevant literals*

$$10x_1 + 4x_2 + 4x_3 + x_4 + x_5 + x_6 \geq 14$$

$$\equiv$$

$$x_1 \wedge (x_2 \vee x_3)$$

The satisfiability of the constraint does not depend on x_4,x_5,x_6

Basic operations on Linear inequalities

addition:
$$\frac{\begin{array}{c}\sum_i a_i.x_i \geq k \\ \sum_i a'_i.x_i \geq k'\end{array}}{\sum_i (a_i + a'_i).x_i \geq k + k'}$$

linear combination:
$$\frac{\begin{array}{c}\sum_i a_i.x_i \geq k \\ \sum_i a'_i.x_i \geq k'\end{array}}{\begin{array}{c}\sum_i (\alpha.a_i + \alpha'.a'_i).x_i \geq \alpha.k + \alpha'.k' \\ \text{with } \alpha > 0 \text{ and } \alpha' > 0\end{array}}$$

division:
$$\frac{\begin{array}{c}\sum_i a_i.x_i \geq k \\ \alpha > 0\end{array}}{\sum_i \frac{a_i.x_i}{\alpha} \geq \frac{k}{\alpha}}$$

TCS division

TCS division:
$$\frac{\sum_i \alpha.a_i.x_i \geq k \qquad \alpha > 0}{\sum_i a_i.x_i \geq \lceil \frac{k}{\alpha} \rceil}$$

tcs division:
$$\frac{2x_2 + 2x_3 + 2x_4 \geq 3}{x_2 + x_3 + x_4 \geq \lceil 3/2 \rceil}$$
$$x_2 + x_3 + x_4 \geq 2$$

UNIVERSITÉ D'ARTOIS

ILP division (Chvátal-Gomory cut)

- When the variables x_i and degree k are integer
- Removes some non integral part of the cut

$$\textit{ILP division:}\quad \frac{\begin{array}{c}\sum_i a_i.x_i \geq k \\ \alpha > 0\end{array}}{\sum_i \lceil \frac{a_i}{\alpha} \rceil .x_i \geq \lceil \frac{k}{\alpha} \rceil}$$

$$\frac{5x_3 + 3x_4 \geq 5}{\begin{array}{c}\lceil 5/5 \rceil x_3 + \lceil 3/5 \rceil x_4 \geq \lceil 5/5 \rceil \\ x_3 + x_4 \geq 1\end{array}}$$

One can always reduce a LPB constraint to a clause!

UNIVERSITÉ D'ARTOIS

Clashing linear combination

Also called Gaussian or Fourier–Motzkin elimination

- Apply linear combination between LPB constraints with at least one opposite literal.
- Generalization of resolution [Hoo88]

clashing combination:

$$\frac{\begin{array}{c}\sum_i a_i.x_i + \alpha' \sum_{j=1}^{m} y_j \geq k \\ \sum_i a'_i.x_i + \alpha \sum_{j=1}^{m} \overline{y_j} \geq k'\end{array}}{\sum_i (\alpha.a_i + \alpha'.a'_i).x_i \geq \alpha.k + \alpha'.k' - \alpha.\alpha'.m}$$

with $\alpha > 0$ and $\alpha' > 0$

$$\frac{x_1 + x_2 + 3x_3 + x_4 \geq 3 \qquad 2\overline{x_1} + 2\overline{x_2} + x_4 \geq 3}{\begin{array}{c}2x_1 + 2x_2 + 6x_3 + 2x_4 + 2\overline{x_1} + 2\overline{x_2} + x_4 \geq 2 \times 3 + 3 \\ 2x_1 + 2x_2 + 6x_3 + 2x_4 + 2 - 2x_1 + 2 - 2x_2 + x_4 \geq 9 \\ 6x_3 + 3x_4 \geq 5\end{array}}$$

Note that $2x + 2\overline{x} = 2$, not 0!

Note that the coefficients are growing!

Some remarks about clashing combination

- Clashing combination looks like resolution?

$$\frac{x_1 + x_3 + x_4 \geq 1 \qquad \overline{x_1} + x_2 + x_5 \geq 1}{x_2 + x_3 + x_4 + x_5 \geq 1}$$

- What about common literals?

$$\frac{x_1 + x_2 + x_3 + x_4 \geq 1 \qquad \overline{x_1} + x_2 + x_4 \geq 1}{2x_2 + x_3 + 2x_4 \geq 1}$$

- With more than one variable?

$$\frac{x_1 + x_2 + x_3 + x_4 \geq 1 \qquad \overline{x_1} + \overline{x_2} + x_4 \geq 1}{x_3 + 2x_4 \geq 0}$$

Saturation

coefficients can be trimmed to the value of the degree

$$\textit{saturation:} \quad \frac{\sum_i a_i.x_i + \sum_j b_j.y_j \geq k \qquad b_j > k}{\sum_i a_i.x_i + \sum_j k.y_j \geq k}$$

$$\frac{6x_3 + 3x_4 \geq 5}{5x_3 + 3x_4 \geq 5}$$

$$\frac{2x_2 + x_3 + 2x_4 \geq 1}{x_2 + x_3 + x_4 \geq 1}$$

Weakening

We can reduce the degree of the constraint by "satisfying" any of its literals

$$\textit{weakening:}\quad \frac{\sum_{i \neq j} a_i.x_i + a_j.x_j \geq k}{\sum_{i \neq j} a_i.x_i \geq k - a_j}$$

$$\frac{5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8}{3x_2 + 2x_3 + 2x_4 + x_5 \geq 3}$$

Useful for reducing the value of the degree!

[Apply linear combination rule with $\overline{x_j} \geq 0$]

Reduction to cardinality

Extract a cardinality constraint from a LPB constraint

$$\textit{reduce to card:} \quad \frac{\begin{array}{c}\sum_{i=1}^{n} a_i.x_i \geq k \\ a_1 \geq a_2 \geq \ldots a_n\end{array}}{\begin{array}{c}\sum_{i=1}^{n} x_i \geq k' \\ \text{with } \sum_{i=1}^{k'-1} a_i < k \leq \sum_{i=1}^{k'} a_i\end{array}}$$

$$\frac{5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8}{x_1 + x_2 + x_3 + x_4 + x_5 \geq 2}$$

The various Cutting Planes

- Linear combination + ILP division = Chvátal-Gomory ILP cutting planes
- Addition + TCS division = Proof complexity cutting planes
- Linear clashing combination + saturation = Hooker's generalized resolution cutting planes

Integrating Cutting Planes in a CDCL solver: replace Resolution during Conflict Analysis by Hooker's Cutting Planes

Outline

Requirements for constraints in a CDCL solver

- Detect falsified state
- Detect propagation of literals
- Provide a "reason" during conflict analysis

Some remarks about clauses

$$l_1 \vee l_2 \vee \ldots \vee l_n$$

- Falsified when all its literals are falsified

$$l_1 \vee l_2 \vee \ldots \vee l_n$$

- Propagates when all but one literals are falsified

$$l_1 \vee l_2 \vee \ldots \vee l_n$$

- Propagates one literal
- Appears at most once as a reason for an assignment

Chaff: 2 watched literals per clause

Some remarks about cardinality constraints

$$l_1 + l_2 + \ldots + l_n \geq k$$

- Falsified when at least $n - k + 1$ literals are falsified

$$l_1 + l_2 + l_3 + l_4 + l_5 + l_6 \geq 4$$

  Note unassigned literals!

- Propagates when exactly $n - k$ literals are falsified

$$l_1 + l_2 + l_3 + l_4 + l_5 + l_6 \geq 4$$

- Propagates k literals
- Appears at most once as a reason for at most k consecutive assignments.

Extended $k + 1$ watched literals per cardinality

Some remarks about LBP constraints

$$a_1.l_1 + a_2.l_2 + ... + a_n.l_n \geq k$$

$$A = \sum_i a_i$$

Slack s: $A - k - \sum_{l_i falsified} a_i$

- Falsified when $s < 0$ (depends on falsified literals)

$$5l_1 + 3l_2 + 2l_3 + l_4 + l_5 + l_6 \geq 6$$

- Propagates remaining literals when $s = 0$

$$5l_1 + 3l_2 + 2l_3 + l_4 + l_5 + l_6 \geq 6$$

- Propagates literals x_i for which $s < a_i$
- May appear several times as a reason for non consecutive assignments

Extended watched literals based on coefficients!

Watched Literals for LPB constraints

Described in Galena [CK03] and BChaff [Par04], may have already existed in PBS or Satzoo.

- General case:
  Let $M = max(a_i)$
  NbWatch = minimal number of literals x_i such that
  $\sum a_i \geq k + M$.
- Cardinality constraints:
  $M = 1$
  $NbWatch = k + 1$
- Clauses:
  $M = 1$
  $k = 1$
  $NbWatch = 2$

Watched literals: consequences

- In LPB constraints, the number of WL is varying during the search.
- In cardinality constraints, the greater the degree, the greater the number of WL.
- Clauses are the best case!
- Big difference for LPB constraint learning

Forced truth values: Implicative and Assertive constraints

- *unit clause*: a clause that propagates one truth value to be satisfiable
- *implicative constraint*: a constraint which propagates *at least* one truth value to be satisfiable.
- a LPB constraint C is *implicative* iff $\exists a_i x_i \in C$ such that $\sum_{j \neq i} a_j < k$ or $\sum a_j - k < a_i$.

Forced truth values: Implicative and Assertive constraints

- *unit clause*: a clause that propagates one truth value to be satisfiable
- *implicative constraint*: a constraint which propagates *at least* one truth value to be satisfiable.
- a LPB constraint C is *implicative* iff $\exists a_i x_i \in C$ such that $\sum_{j \neq i} a_j < k$ or $\sum a_j - k < a_i$.

Example

$$4x_1 + 3x_2 + x_3 + x_4 \geq 8$$

propagates x_1 and x_2

- $3 + 1 + 1 < 8$ so x_1 must be satisfied, same thing on $3x_2 + x_3 + x_4 \geq 4$.

Forced truth values: Implicative and Assertive constraints

- *unit clause*: a clause that propagates one truth value to be satisfiable
- *implicative constraint*: a constraint which propagates *at least* one truth value to be satisfiable.
- a LPB constraint C is *implicative* iff $\exists a_i x_i \in C$ such that $\sum_{j \neq i} a_j < k$ or $\sum a_j - k < a_i$.

Example

$$4x_1 + 3x_2 + x_3 + x_4 \geq 8$$

propagates x_1 and x_2

- $3 + 1 + 1 < 8$ so x_1 must be satisfied, same thing on $3x_2 + x_3 + x_4 \geq 4$.
- One can note that $\sum a_j - k = 1$ so any literal x_i with a coef greater than 1 must be propagated.

Forced truth values: Implicative and Assertive constraints

- *unit clause*: a clause that propagates one truth value to be satisfiable
- *implicative constraint*: a constraint which propagates *at least* one truth value to be satisfiable.
- a LPB constraint C is *implicative* iff $\exists a_i x_i \in C$ such that $\sum_{j \neq i} a_j < k$ or $\sum a_j - k < a_i$.

Example

$$4x_1 + 3x_2 + x_3 + x_4 \geq 8$$

propagates x_1 and x_2

- $3 + 1 + 1 < 8$ so x_1 must be satisfied, same thing on $3x_2 + x_3 + x_4 \geq 4$.
- One can note that $\sum a_j - k = 1$ so any literal x_i with a coef greater than 1 must be propagated.
- Rewrite into $x_1 \wedge x_2 \wedge (x_3 + x_4 \geq 1)$?

Outline

Problems with the integration of Cutting Planes

- Derived LPB constraint must be redondant (logical consequence)
  no problem here
- Derived LPB constraint must be falsified at current decision level
  free for resolution, requires special care for CP
- Derived LPB constraint must be assertive at backtrack level
  syntactical test for clauses, not for PB constraints

Computing the backtrack level

- Just a *max* for clauses
- More complicated for LPBC: an LPB constraint may be assertive at different backtrack levels.
  - Decision literals are no longer "UIP"!
  - Need to backtrack to the first one

Example

Given the decisions $x_1, \neg x_2, \neg x_3$
and the falsified LBP $3x_1 + 2x_2 + x_3 + x_4 \geq 5$.
Where should I backtrack?

UNIVERSITÉ D'ARTOIS

Computing the backtrack level

- Just a *max* for clauses
- More complicated for LPBC: an LPB constraint may be assertive at different backtrack levels.
  - Decision literals are no longer "UIP"!
  - Need to backtrack to the first one

Example

Given the decisions $x_1, \neg x_2, \neg x_3$
and the falsified LBP $3x_1 + 2x_2 + x_3 + x_4 \geq 5$.
Where should I backtrack?
backtrack to $x_1, \neg x_2$ to propagate x_3 and x_4?

Computing the backtrack level

- Just a *max* for clauses
- More complicated for LPBC: an LPB constraint may be assertive at different backtrack levels.
  - Decision literals are no longer "UIP"!
  - Need to backtrack to the first one

Example

Given the decisions $x_1, \neg x_2, \neg x_3$
and the falsified LBP $3x_1 + 2x_2 + x_3 + x_4 \geq 5$.
Where should I backtrack?
backtrack to $x_1, \neg x_2$ to propagate x_3 and x_4?
or to decision level 0 to propagate x_1?

Computing an assertive clause

- Let C be a falsified constraint
- $S = lit(C)_{>dl}$
- $D = lit(C)_{=dl}$

1. Pick the reason R for the latest assignment a in C
2. Compute $S = S \cup lit(R)_{>dl}$ and $D = D \cup lit(R)_{=dl} \setminus \{a\}$

- Repeat $1 - 2$ until $|D| = 1$

Computing an assertive LPB constraint

1. Let C be a falsified constraint
2. Pick the reason R for the latest assignment a in C
3. compute α and α' to remove a from C.
4. Weaken R if needed to ensure that the LPB constraint generated by applying linear combination is falsified (reduction)
5. Apply clashing combination: $C = CC(C, R, \alpha, \alpha')$
6. Apply saturation
7. Update the slack of the generated constraint
8. Repeat 2-7 until the slack is 0

Use arbitrary precision arithmetic to prevent overflow

Computing an assertive LPB constraint

1. Let C be a falsified constraint
2. Pick the reason R for the latest assignment a in C
3. compute α and α' to remove a from C.
4. Weaken R if needed to ensure that the LPB constraint generated by applying linear combination is falsified (reduction)
5. Apply clashing combination: $C = CC(C, R, \alpha, \alpha')$
6. Apply saturation
7. Update the slack of the generated constraint
8. Repeat 2-7 until the slack is 0

Use arbitrary precision arithmetic to prevent overflow
Not needed if reduced to cardinality constraint

Example

$$\begin{cases} (C_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (C_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (C_3) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

$$\neg x_5^0, x_1^0[C_1], \neg x_4^1, x_3^1[C_3], x_2^1[C_1]$$

$Poss(C_1) = +2,\ Poss(C_2) = -2$
Red. x_1: (C_1') $3x_2 + 2x_3 + 2x_4 + x_5 \geq 3$ **poss=+2**
Red. x_3: (C_1'') $x_2 + x_4 + x_5 \geq 1$ **poss=0**
$CC(C_2, 3 \times C_1'') = 2\overline{x_1}^0 + 2\overline{x_3}^1 + x_4^1 + 2x_5^0 \geq 2$
Assertive at decision level 0 (x_3 is propagated to 1).

Would learn $\overline{x_1} + x_4 + x_5 \geq 1$ with clause learning.
Assertive at decision level 0 (x_4 is propagated to 1).

UNIVERSITÉ D'ARTOIS

A brief history of LPB constraints within SAT solvers

[Bar95] DPLL extension to LPB [opbdp]
[Wal97] (and [Pre02, Pre04]) local search for LPB
[MFSO97] B'n'B LPB solver (GRASP) [bsolo]
[WKS01] incremental SAT with LPB (GRASP) [satire]
[ARMS02, Sak03] LPB contraints with Chaff/CDCL solver [pbs, see also satzoo (minisat)]
[Gin02] extended RelSAT to LPB (LPB learning)
[CK03] CDCL with LPB learning [galena]
[Par04] describe a generic CDCL solver based on group theory handling arbitrary boolean gates.
[SS06] CDCL solver able to learn temporary LPB constraints [pueblo]
[ALS09] Generalization of PBO [WBO/OpenWBO]
[EN18] Specific division rule [RoundingSAT]

A brief history of LPB constraints within SAT solvers

[Bar95] DPLL extension to LPB [opbdp]
[Wal97] (and [Pre02, Pre04]) local search for LPB
[MFSO97] B'n'B LPB solver (GRASP) [bsolo]
[WKS01] incremental SAT with LPB (GRASP) [satire]
[ARMS02, Sak03] LPB contraints with Chaff/CDCL solver
[pbs, see also satzoo (minisat)]
[Gin02] extended RelSAT to LPB (LPB learning)
[CK03] CDCL with LPB learning [galena]
[Par04] describe a generic CDCL solver based on group theory handling arbitrary boolean gates.
[SS06] CDCL solver able to learn temporary LPB constraints [pueblo]
[ALS09] Generalization of PBO [WBO/OpenWBO]
[EN18] Specific division rule [RoundingSAT]
Main interest moved to MAXSAT since a decade,

A brief history of LPB constraints within SAT solvers

[Bar95] DPLL extension to LPB [opbdp]
[Wal97] (and [Pre02, Pre04]) local search for LPB
[MFSO97] B'n'B LPB solver (GRASP) [bsolo]
[WKS01] incremental SAT with LPB (GRASP) [satire]
[ARMS02, Sak03] LPB contraints with Chaff/CDCL solver [pbs, see also satzoo (minisat)]
[Gin02] extended RelSAT to LPB (LPB learning)
[CK03] CDCL with LPB learning [galena]
[Par04] describe a generic CDCL solver based on group theory handling arbitrary boolean gates.
[SS06] CDCL solver able to learn temporary LPB constraints [pueblo]
[ALS09] Generalization of PBO [WBO/OpenWBO]
[EN18] Specific division rule [RoundingSAT]
Main interest moved to MAXSAT since a decade, Major work on CNF encoding of cardinality and LBP constraints (Minisat+ effect)

SAT4J Pseudo

- Implements the LPB learning described in PBChaff [Gin02] and Galena[CK03]
  - Cardinality learning preferred to LPB learning
  - No management of integer overflow
  - Solvers no longer developed
- Based on Minisat 1 specification implemented in Java
- Two versions available: resolution based inference or Hooker's generalized resolution "cutting planes" based inference.

LPB constraints case: what can go wrong

Boolean propagation lazy data structure for maintaining an alert value require more bookkeeping than for clauses.

Assertive constraints cannot syntactically be identified.

Linear combination between two conflictual constraints doesn't necessary result in a falsified constraint! Weakening may be needed to obtain a cutting plane.

Coefficient management In some cases, the coefficients of the LPB keep growing.

Consequence: learning PB constraints does slow down the solver!

Solutions:

- Reduce learned clauses to Cardinality constraints (Galena, PBChaff)
- Learn both a clause and a PB constraint, then eventually remove the PB constraint (Pueblo).
- Learn clauses (Minisat+, PBS).

Outline

Optimization using strengthening (linear search)

input : A set of clauses, cardinalities and pseudo-boolean constraints setOfConstraints and an objective function objFct to minimize

output: a model of setOfConstraints, or UNSAT if the problem is unsatisfiable.

answer $\leftarrow$ `isSatisfiable` (setOfConstraints);
if answer *is* UNSAT **then**
  return UNSAT
end
repeat
  model $\leftarrow$ answer;
  answer $\leftarrow$ `isSatisfiable` (setOfConstraints $\cup$ $\{$objFct $<$ objFct (model)$\}$);
until *(*answer *is* UNSAT*)*;
return model;

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Objective function

$$\text{min:} \quad 4x_2 + 2x_3 + x_5$$

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Model

$$\overline{x_1}, x_2, \overline{x_3}, x_4, x_5$$

Objective function

$$\text{min:} \quad 4x_2 + 2x_3 + x_5$$

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Model

$\overline{x_1}, x_2, \overline{x_3}, x_4, x_5$

Objective function

min: $4x_2 + 2x_3 + x_5$

$<$

Objective function value

5

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Objective function

min: $4x_2 + 2x_3 + x_5 \quad < \quad 5$

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Model

$x_1, \overline{x_2}, x_3, \overline{x_4}, x_5$

Objective function

min: $4x_2 + 2x_3 + x_5 \quad < \quad 5$

UNIVERSITÉ D'ARTOIS

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Model

$x_1, \overline{x_2}, x_3, \overline{x_4}, x_5$

Objective function

min: $4x_2 + 2x_3 + x_5$

$<$

Objective function value

$3 < 5$

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Objective function

$$\text{min:} \quad 4x_2 + 2x_3 + x_5 \qquad < \qquad 3$$

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Model

$$x_1, \overline{x_2}, \overline{x_3}, x_4, x_5$$

Objective function

min: $4x_2 + 2x_3 + x_5 \quad < \quad 3$

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Model

$$x_1, \overline{x_2}, \overline{x_3}, x_4, x_5$$

Objective function

min: $4x_2 + 2x_3 + x_5$

$<$

Objective function value

$1 < 3$

UNIVERSITÉ D'ARTOIS

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Objective function

min: $4x_2 + 2x_3 + x_5 \quad < \quad 1$

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Objective function

min: $4x_2 + 2x_3 + x_5 \quad < \quad 1$

Optimization algorithm

Formula :

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

Objective function

$$\text{min:} \quad 4x_2 + 2x_3 + x_5$$

The objective function value 1 is optimal for the formula.

$x_1, \overline{x_2}, \overline{x_3}, x_4, x_5$ is an optimal solution.

Remarks about the optimization procedure

- No need for an initial upper bound!
- Phase selection strategy takes into account the objective function.
- External to the PB solver: can use any PB solver.
- SAT, SAT, SAT, ..., SAT, UNSAT pattern
- SAT answer usually easier to provide than UNSAT one
- In practice: optimality is often hard to prove for the Resolution based PB solver (pigeon hole?).
- Ideally, would like to run the CP PB solver to prove optimality at the end.
- Problem: how to detect that we need to prove optimality?

UNIVERSITÉ D'ARTOIS

Remarks about the optimization procedure

- No need for an initial upper bound!
- Phase selection strategy takes into account the objective function.
- External to the PB solver: can use any PB solver.
- SAT, SAT, SAT, ..., SAT, UNSAT pattern
- SAT answer usually easier to provide than UNSAT one
- In practice: optimality is often hard to prove for the Resolution based PB solver (pigeon hole?).
- Ideally, would like to run the CP PB solver to prove optimality at the end.
- Problem: how to detect that we need to prove optimality?
- Nice idea suggested by Olivier Roussel submitted to PB 2010: run the Res and CP PB solvers in parallel!

UNIVERSITÉ D'ARTOIS

Optimization with solvers running in parallel

input : A set of clauses, cardinalities and pseudo-boolean constraints setOfConstraints and an objective function objFct to minimize

output: a model of setOfConstraints, or UNSAT if the problem is unsatisfiable.

answer ← `isSatisfiable` (setOfConstraints);
if answer *is* UNSAT **then**
  return UNSAT
end
repeat
  model ← answer;
  answer ← `isSatisfiable` (setOfConstraints ∪ {objFct < objFct (model)});
until (answer *is* UNSAT);
return model;

logic-synthesis/normalized-jac3.opb @ PB2010

```
% Cutting Planes
1.17/0.78 c #vars       1731
1.17/0.78 c #constraints  1254
1.76/1.03 c SATISFIABLE
1.76/1.03 c OPTIMIZING...
1.76/1.03 o 26
3.40/1.91 o 25
5.93/3.41 o 24
6.97/4.33 o 23
7.49/4.88 o 22
8.44/5.72 o 21
9.00/6.27 o 20
9.62/6.87 o 19
10.44/7.61 o 18
11.54/8.79 o 17
13.03/10.13 o 16
25.34/22.07 o 15
1800.11/1773.42 s SATISFIABLE
```

```
% Resolution
1.17/0.75 c #vars       1731
1.17/0.75 c #constraints  1254
1.57/0.91 c SATISFIABLE
1.57/0.91 c OPTIMIZING...
1.57/0.91 o 26
2.55/1.42 o 23
2.96/1.60 o 22
3.35/1.80 o 21
16.34/14.32 o 20
55.04/52.91 o 19
766.33/763.00 o 18
1800.04/1795.76 s SATISFIABLE
```

logic-synthesis/normalized-jac3.opb @ PB2010

```
% Cutting Planes
1.17/0.78 c #vars      1731
1.17/0.78 c #constraints  1254
1.76/1.03 c SATISFIABLE
1.76/1.03 c OPTIMIZING...
1.76/1.03 o 26
3.40/1.91 o 25
5.93/3.41 o 24
6.97/4.33 o 23
7.49/4.88 o 22
8.44/5.72 o 21
9.00/6.27 o 20
9.62/6.87 o 19
10.44/7.61 o 18
11.54/8.79 o 17
13.03/10.13 o 16
25.34/22.07 o 15
1800.11/1773.42 s SATISFIABLE
```

```
% Res // CP
1.35/0.84 c #vars      1731
1.35/0.84 c #constraints  1254
1.99/1.85 c SATISFIABLE
1.99/1.85 c OPTIMIZING...
1.99/1.85 o 26 (CuttingPlanes)
2.61/2.89 o 25 (Resolution)
3.91/3.92 o 24 (Resolution)
4.12/5.00 o 23 (Resolution)
5.92/6.01 o 22 (Resolution)
7.72/7.04 o 21 (Resolution)
9.63/8.07 o 20 (CuttingPlanes)
13.04/10.09 o 19 (CuttingPlanes)
15.66/12.10 o 18 (CuttingPlanes)
20.27/15.14 o 17 (CuttingPlanes)
70.03/41.35 o 16 (CuttingPlanes)
218.63/118.14 o 15 (CuttingPlanes)
305.11/164.68 s OPTIMUM FOUND
```

logic-synthesis/normalized-jac3.opb @ PB2010

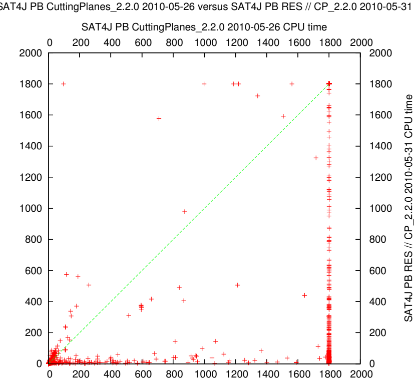
Cutting Planes

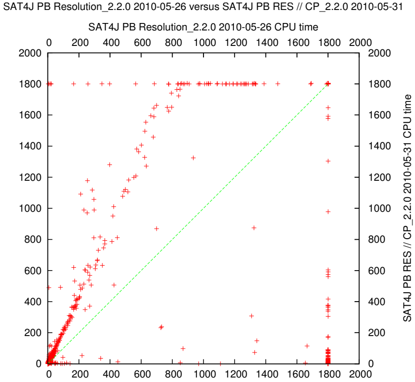
```
1800.11/1773.42 s SATISFIABLE
1800.11/1773.41 c learnt clauses : 2618
1800.11/1773.42 c speed (assignments/second) : 226
```

Res // CP

```
305.11/164.68 s OPTIMUM FOUND
305.11/164.68 c learnt clauses : 1318
305.11/164.68 c speed (assignments/second) : 3927
```

Scatter plots Res // CP vs CP, Resolution





UNIVERSITÉ D'ARTOIS

Regarding the idea to run the two solvers in //

- Res // CP globally better than Res or CP solver during PB 2010 in number of benchmarks solved.
- Res // CP twice as slow as Res on many benchmarks.
- Decision problems: solves the union of the benchmarks solved by Res and CP in half the timeout (CPU time taken into account, not wall clock time).
- Optimization problems: "cooperation" of solvers allow to solve new benchmarks!

The Pseudo Boolean evaluations

http://www.cril.univ-artois.fr/PB16/

- Organized by Olivier Roussel and Vasco Manquinho from 2005 to 2012, and 2016
- Uniform input format: OPB files
- Independent assessment of the PB solvers
- Detailed results available for each solver
- Various technologies used since 2006
- WBO category since 2010

Partial results of the PB12 evaluation

	MiniSat+	Cplex	Clasp	Sat4j Res // CP	Bsolo	Sat4j Res	
Dec. (#355)	91	88	97	**119**	115	91	UNS
	129	104	**149**	130	123	140	SAT
Opt S (#657)	**22**	21	21	**22**	21	21	UNS
	257	**355**	260	253	279	257	OPT
Opt B (#416)	**23**	-	-	**23**	-	**23**	UNS
	15	-	-	**80**	-	74	OPT

See http://www.cril.univ-artois.fr/PB12/results/results.php?idev=67 for details

Partial results of the PB16 evaluation

	Min-iSat+	Open-WBO	Sat4j Res // CP	cdcl-cp	NaPS	
Dec.	935	1049	1052	**1092**	1023	UNS
(#1783)	**384**	329	315	303	338	SAT
Opt S	76	45	**89**	**89**	85	UNS
(#1600)	713	781	672	685	**802**	OPT
Opt B	**70**	-	**70**	-	69	UNS
(#1109)	166	-	196	-	**305**	OPT

See `http://www.cril.univ-artois.fr/PB16/results/ranking.php?idev=81` for details

Outline

Semantic cardinality detection

Armin Biere, Daniel Le Berre, Emmanuel Lonca, Norbert Manthey: Detecting Cardinality Constraints in CNF. SAT 2014: 285-301

- 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

Cryptography instance: cardinality constraints *vs.* clauses

- sha1-006.cnf : 478484 clauses

- sha1-006.{cnf/opb}:

Threshold	size	count
1	3	17
2	4	321
2	5	3
3	5	872
3	6	13
4	6	3248

Threshold	size	count
4	7	50
5	7	36403
5	8	66
6	8	41643
6	9	656
and 41787 remaining clauses		

- sha1-006.{cnf/opb} contains 125079 constraints : reduced by a factor of 4

PHP: cardinality constraints *vs.* clauses

PHP: inconsistency proof computation time

- pigeons-100-hole.cnf:
  - resolution $\rightarrow$ timeout (900s)
  - generalized resolution[Hoo88] $\rightarrow$ timeout (900s)
- pigeons-100-hole.opb:
  - resolution $\rightarrow$ timeout (900s)
  - generalized resolution[Hoo88] $\rightarrow < 1s$.

- Cardinality constraints allow the use of stronger proof systems

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)
- 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 [CCT87]
- Nested encoding
- Two product encoding [Che10]
- Sequential encoding [Sin05]
- Commander encoding [FG10]
- Ladder encoding [GN04]
- Adder encoding [ES06]
- Cardinality Networks [ANORC09]
- ...

Syntactic *vs.* semantic detection

- Syntactic detection:
  - need of an *ad hoc* algorithm for each $\{\text{encoding}, k\}$
- Our semantic detection:
  - 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

Semantic detection of AtMost-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:

$$\sum_{i=1}^{n} (\neg x_i) + m_1 + \ldots + m_p \leq n-1$$

UNIVERSITÉ D'ARTOIS

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

UNIVERSITÉ D'ARTOIS

Semantic detection of AtMost-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 AtMost-k constraint: example

formula :

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

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

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

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

Semantic detection of AtMost-k constraint: example

formula :

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

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

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

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

Semantic detection of AtMost-k constraint: example

formula :

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

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

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

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

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

Semantic detection of AtMost-k constraint: example

formula :

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

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

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

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

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

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

Cardinality constraint extension

Cardinality constraint extension:

1. let $\alpha = \sum_{i=1}^{n} x_i \leq k$
2. initialization of the propagation set $\gamma = \{v_i, \neg v_i \mid v \in \mathsf{PS}\}$
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

UNIVERSITÉ D'ARTOIS

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

Sat4jCP uses Generalized Resolution, not Cutting Planes, i.e. can only derive clauses when applied to clauses.[1]

[1] Thanks to Jakob Nordström 's group for discussions on that subject

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)

UNIVERSITÉ D'ARTOIS

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)

Lingeling efficient for pairwise encoding only (the best)

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)

SBSAT efficient for small instances ; best on ladder encoding

UNIVERSITÉ D'ARTOIS

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)

Sat4jCP bad without preprocessing

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)

Synt.+Sat4jCP very efficient when specific algorithms are implemented ; best on sequential and two-product encodings

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)

Sem.+Sat4jCP efficient on most encodings ; best on binary, sequential and commander encodings

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)

UNIVERSITÉ D'ARTOIS

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 [VS10] (clasp unable to solve within 24h): solved within a second thanks to semantic preprocessing (AtMost-3 constraints inside)

A Conflict Analysis with Generalized Resolution

Consider the following constraints

$$\chi_1 : \bar{a} + \bar{b} + f \geq 2$$
$$\chi_2 : 3\bar{x} + a + b + d + e \geq 4$$
$$\chi_3 : 4a + 2b + 2c + x \geq 5$$

A Conflict Analysis with Generalized Resolution

Consider the following constraints

$\chi_1 : \bar{a} + \bar{b} + f \geq 2$
$\chi_2 : 3\bar{x} + a + b + d + e \geq 4$
$\chi_3 : 4a + 2b + 2c + x \geq 5$

$f = 0@1$

A Conflict Analysis with Generalized Resolution

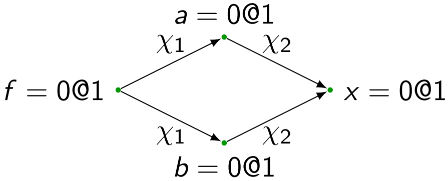
Consider the following constraints

$$\chi_1 : \bar{a} + \bar{b} + f \geq 2$$
$$\chi_2 : 3\bar{x} + a + b + d + e \geq 4$$
$$\chi_3 : 4a + 2b + 2c + x \geq 5$$

$$\begin{tikzcd}
 & a = 0@1 \\
f = 0@1 \arrow[ur, "\chi_1"] \arrow[dr, "\chi_1"'] & \\
 & b = 0@1
\end{tikzcd}$$

Consider the following constraints

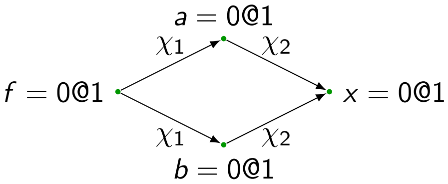
$$\chi_1 : \bar{a} + \bar{b} + f \geq 2$$
$$\chi_2 : 3\bar{x} + a + b + d + e \geq 4$$
$$\chi_3 : 4a + 2b + 2c + x \geq 5$$



A Conflict Analysis with Generalized Resolution

Consider the following constraints

$\chi_1 : \bar{a} + \bar{b} + f \geq 2$
$\chi_2 : 3\bar{x} + a + b + d + e \geq 4$
$\chi_3 : 4a + 2b + 2c + x \geq 5$

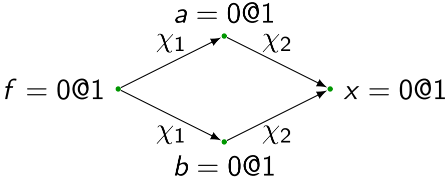


We have falsified χ_3!

A Conflict Analysis with Generalized Resolution

Consider the following constraints

$\chi_1 : \bar{a} + \bar{b} + f \geq 2$
$\chi_2 : 3\bar{x} + a + b + d + e \geq 4$
$\chi_3 : 4a + 2b + 2c + x \geq 5$



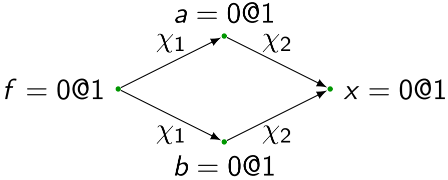
We have falsified χ_3! This conflict is analyzed by resolving χ_3 against χ_2 which is the reason for $\bar{x}$

$$\frac{\chi_3 \qquad \chi_2}{13a + 7b + 6c + d + e \geq 16}$$

A Conflict Analysis with Generalized Resolution

Consider the following constraints

$\chi_1 : \bar{a} + \bar{b} + f \geq 2$
$\chi_2 : 3\bar{x} + a + b + d + e \geq 4$
$\chi_3 : 4a + 2b + 2c + x \geq 5$



We have falsified χ_3! This conflict is analyzed by resolving χ_3 against χ_2 which is the reason for $\bar{x}$

$$\frac{\chi_3 \qquad \chi_2}{13a + 7b + 6c + d + e \geq 16}$$

This constraint is learned because it propagates a to 1 at level 0

A Problem with the Learned Constraint?

The constraint learned after conflict analysis is

$$13a + 7b + 6c + d + e \geq 16$$

A Problem with the Learned Constraint?

The constraint learned after conflict analysis is

$$13a + 7b + 6c + d + e \geq 16$$

Let us have a close look at this constraint...

A Problem with the Learned Constraint?

The constraint learned after conflict analysis is

$$13a + 7b + 6c + d + e \geq 16$$

Let us have a close look at this constraint...

A Problem with the Learned Constraint?

The constraint learned after conflict analysis is

$$13a + 7b + 6c + d + e \geq 16$$

Let us have a close look at this constraint...
Literals d and e have no effect on the constraint: they are irrelevant!

A Problem with the Learned Constraint?

The constraint learned after conflict analysis is

$$13a + 7b + 6c + d + e \geq 16$$

Let us have a close look at this constraint...
Literals d and e have no effect on the constraint: they are irrelevant!
In particular, this means that removing these literals from the constraint preserves equivalence

$$13a + 7b + 6c \geq 16$$

A Problem with the Learned Constraint?

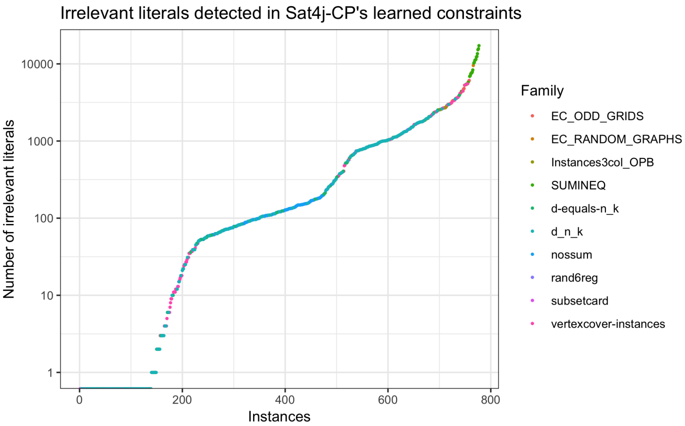
The constraint learned after conflict analysis is

$$13a + 7b + 6c + d + e \geq 16$$

Let us have a close look at this constraint...
Literals d and e have no effect on the constraint: they are irrelevant!
In particular, this means that removing these literals from the constraint preserves equivalence

$$13a + 7b + 6c \geq 14$$

Irrelevant Literals in Practice (in Sat4j)



- Number of irrelevant literals in Sat4j-CP's first 5,000 learned constraints
- Experiments conducted on the 777 decision benchmarks from PB'16
- Sat4j as an example of Generalized-Resolution-based solver

RoundingSat uses a different approach, which mainly consists in using the division rule instead of saturation

$$\frac{\sum_{i=1}^{n} a_i l_i \geq d \qquad \alpha > 0}{\sum_{i=1}^{n} \lceil \frac{a_i}{\alpha} \rceil l_i \geq \lceil \frac{d}{\alpha} \rceil} \text{(division)}$$

A Conflict Analysis in RoundingSat

Consider the following constraints:

$$\chi_1 : 2\bar{c} + 2\bar{d} + b + \bar{e} \geq 4$$
$$\chi_2 : 3a + 3b + c + d + e \geq 4$$
$$\chi_3 : 2\bar{a} + b + e \geq 2$$

A Conflict Analysis in RoundingSat

Consider the following constraints:

$$\chi_1 : 2\bar{c} + 2\bar{d} + b + \bar{e} \geq 4$$
$$\chi_2 : 3a + 3b + c + d + e \geq 4$$
$$\chi_3 : 2\bar{a} + b + e \geq 2$$

$e = 1@1$ •

UNIVERSITÉ D'ARTOIS

A Conflict Analysis in RoundingSat

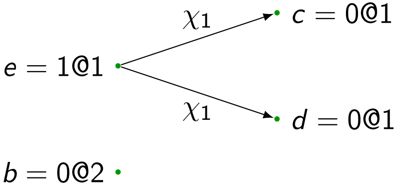
Consider the following constraints:

$\chi_1 : 2\bar{c} + 2\bar{d} + b + \bar{e} \geq 4$
$\chi_2 : 3a + 3b + c + d + e \geq 4$
$\chi_3 : 2\bar{a} + b + e \geq 2$

$$\begin{tikzcd}
 & c = 0@1 \\
e = 1@1 \arrow[ur, "\chi_1"] \arrow[dr, "\chi_1"'] & \\
 & d = 0@1
\end{tikzcd}$$

Consider the following constraints:

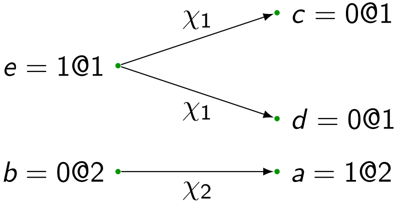
$$\chi_1 : 2\bar{c} + 2\bar{d} + b + \bar{e} \geq 4$$
$$\chi_2 : 3a + 3b + c + d + e \geq 4$$
$$\chi_3 : 2\bar{a} + b + e \geq 2$$



A Conflict Analysis in RoundingSat

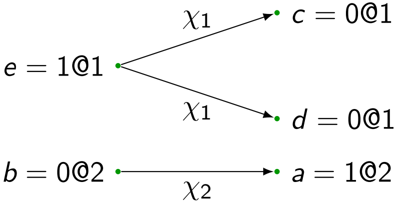
Consider the following constraints:

$$\chi_1 : 2\bar{c} + 2\bar{d} + b + \bar{e} \geq 4$$
$$\chi_2 : 3a + 3b + c + d + e \geq 4$$
$$\chi_3 : 2\bar{a} + b + e \geq 2$$



Consider the following constraints:

$\chi_1 : 2\bar{c} + 2\bar{d} + b + \bar{e} \geq 4$
$\chi_2 : 3a + 3b + c + d + e \geq 4$
$\chi_3 : 2\bar{a} + b + e \geq 2$



We have falsified χ_3!

Consider the following constraints:

$$\chi_1 : 2\bar{c} + 2\bar{d} + b + \bar{e} \geq 4$$
$$\chi_2 : 3a + 3b + c + d + e \geq 4$$
$$\chi_3 : 2\bar{a} + b + e \geq 2$$

$$\begin{tikzcd}
 & c = 0@1 \\
e = 1@1 \arrow[ur, "\chi_1"] \arrow[dr, "\chi_1"'] & \\
 & d = 0@1 \\
b = 0@2 \arrow[r, "\chi_2"'] & a = 1@2
\end{tikzcd}$$

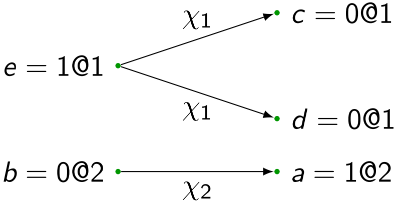
We have falsified χ_3! Before applying clashing addition, χ_2 is weakened on e and divided by 3

$$\dfrac{\dfrac{\chi_2}{3a + 3b + c + d \geq 3}}{a + b + c + d \geq 1}$$

A Conflict Analysis in RoundingSat

Consider the following constraints:

$$\chi_1 : 2\bar{c} + 2\bar{d} + b + \bar{e} \geq 4$$
$$\chi_2 : 3a + 3b + c + d + e \geq 4$$
$$\chi_3 : 2\bar{a} + b + e \geq 2$$



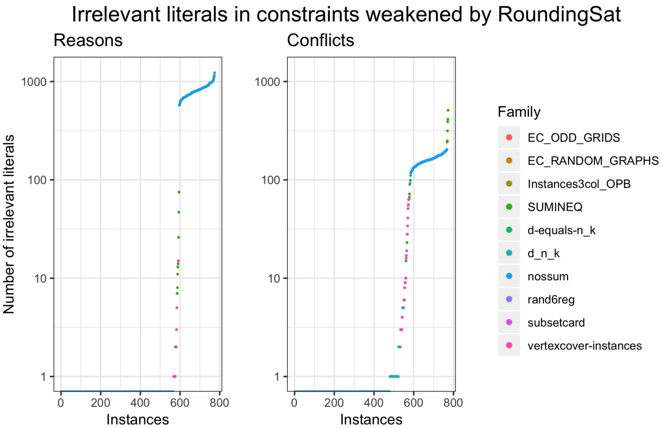
We have falsified χ_3! Before applying clashing addition, χ_2 is weakened on e and divided by 3

$$\cfrac{\cfrac{\chi_2}{3a + 3b + c + d \geq 3}}{a + b + c + d \geq 1}$$

Observe how c and d become irrelevant, and then relevant again, and how they prevent the inference of the stronger constraint

$$a + b \geq 1$$

Irrelevant Literals in Practice (in RoundingSat)



- Number of irrelevant literals in RoudingSat's first 100,000 weakened constraints
- Experiments conducted on the 777 decision benchmarks from PB'16

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$17a + 10b + 10c + d + e \geq 17$$

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$17a + 10b + 10c + d + e \geq 17 \equiv \quad 17a + 10b + 10c \geq 15$$

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$\begin{aligned} 17a + 10b + 10c + d + e \geq 17 \equiv &\quad 17a + 10b + 10c \geq 15 \\ \equiv &\quad 15a + 10b + 10c \geq 15 \end{aligned}$$

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$\begin{aligned} 17a + 10b + 10c + d + e \geq 17 &\equiv \quad 17a + 10b + 10c \geq 15 \\ &\equiv \quad 15a + 10b + 10c \geq 15 \\ &\equiv \quad 3a + 2b + 2c \geq 3 \end{aligned}$$

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$\begin{aligned} 17a + 10b + 10c + d + e \geq 17 &\equiv \quad 17a + 10b + 10c \geq 15 \\ &\equiv \quad 15a + 10b + 10c \geq 15 \\ &\equiv \quad 3a + 2b + 2c \geq 3 \end{aligned}$$

Applying generalized resolution is harder when coefficients are big due to the need of arbitrary precision

UNIVERSITÉ D'ARTOIS

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$
\begin{aligned}
17a + 10b + 10c + d + e \geq 17 &\equiv \quad 17a + 10b + 10c \geq 15 \\
&\equiv \quad 15a + 10b + 10c \geq 15 \\
&\equiv \quad 3a + 2b + 2c \geq 3
\end{aligned}
$$

Applying generalized resolution is harder when coefficients are big due to the need of arbitrary precision

Irrelevant literals hide cardinality constraints:

$$3a + 3b + 3c + 3d + e + f \geq 6$$

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$\begin{aligned} 17a + 10b + 10c + d + e \geq 17 &\equiv \quad 17a + 10b + 10c \geq 15 \\ &\equiv \quad 15a + 10b + 10c \geq 15 \\ &\equiv \quad 3a + 2b + 2c \geq 3 \end{aligned}$$

Applying generalized resolution is harder when coefficients are big due to the need of arbitrary precision

Irrelevant literals hide cardinality constraints:

$$3a + 3b + 3c + 3d + e + f \geq 6 \equiv \quad 3a + 3b + 3c + 3d \geq 4$$

UNIVERSITÉ D'ARTOIS

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$
\begin{aligned}
17a + 10b + 10c + d + e \geq 17 &\equiv \quad 17a + 10b + 10c \geq 15 \\
&\equiv \quad 15a + 10b + 10c \geq 15 \\
&\equiv \quad 3a + 2b + 2c \geq 3
\end{aligned}
$$

Applying generalized resolution is harder when coefficients are big due to the need of arbitrary precision

Irrelevant literals hide cardinality constraints:

$$
\begin{aligned}
3a + 3b + 3c + 3d + e + f \geq 6 &\equiv \quad 3a + 3b + 3c + 3d \geq 4 \\
&\equiv \quad a + b + c + d \geq 2
\end{aligned}
$$

Why are Irrelevant Literals an Issue?

Irrelevant literals make coefficients bigger than necessary:

$$\begin{aligned} 17a + 10b + 10c + d + e \geq 17 &\equiv \quad 17a + 10b + 10c \geq 15 \\ &\equiv \quad 15a + 10b + 10c \geq 15 \\ &\equiv \quad 3a + 2b + 2c \geq 3 \end{aligned}$$

Applying generalized resolution is harder when coefficients are big due to the need of arbitrary precision

Irrelevant literals hide cardinality constraints:

$$\begin{aligned} 3a + 3b + 3c + 3d + e + f \geq 6 &\equiv \quad 3a + 3b + 3c + 3d \geq 4 \\ &\equiv \quad a + b + c + d \geq 2 \end{aligned}$$

Efficient data structures implemented in PB solvers cannot be used when cardinality constraints are hidden

Conclusion

- PB constraint represent concisely some Boolean functions
- It is possible to introduce some kind of cutting planes reasoning in CDCL solvers, driven by conflict analysis
- Solves PHP instances expressed by cardinalities (not CNF)
- Semantic cardinality detection can help when input is CNF
- But in practice learning LPB often slows down the solver
- Last decade focussed on encoding those constraints into CNF
- Recent work toward new proof systems, cardinality detection (Jakob Nordstrom's group)
- None of existing rules prevent irrelevant literals production

Albert Atserias, Johannes Klaus Fichte, and Marc Thurley.
Clause-learning algorithms with many restarts and bounded-width resolution.
J. Artif. Intell. Res. (JAIR), 40:353–373, 2011.

Carlos Ansótegui, Jose Larrubia, Chu Min Li, and Felip Manyà.

Exploiting multivalued knowledge in variable selection heuristics for sat solvers.
Ann. Math. Artif. Intell., 49(1-4):191–205, 2007.

Josep Argelich, Inês Lynce, and João P. Marques Silva.
On solving boolean multilevel optimization problem.
In *Proc. of IJCAI'09*, pages 393–398, 2009.

Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell.
Cardinality networks and their applications.

In Oliver Kullmann, editor, *SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2009.

Carlos José Ansótegui.
Complete SAT solvers for Many-Valued CNF Formulas.
PhD thesis, University of Lleida, 2004.

F. Aloul, A. Ramani, I. Markov, and K. Sakallah.
Generic ILP versus Specialized 0-1 ILP: an update.
In *Proceedings of ICCAD'02*, pages 450–457, 2002.

Peter Barth.
A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization.
Technical Report MPI–I–95–2–003, Max-Plank-Institut fur Informatik, Saarbrücken, 1995.

W. Cook, C.R. Coullard, and Gy. Turán.
On the complexity of cutting-plane proofs.

UNIVERSITÉ D'ARTOIS

Discrete Applied Mathematics, 18(1):25 – 38, 1987.

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

Donald Chai and Andreas Kuehlmann.
A fast pseudo-boolean constraint solver.
In *ACM/IEEE Design Automation Conference (DAC'03)*, pages 830–835, Anaheim, CA, 2003.

Jan Elffers and Jakob Nordström.
Divide and conquer: Towards faster pseudo-boolean solving.
In *Proc. of IJCAI'18*, pages 1291–1299, 2018.

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

UNIVERSITÉ D'ARTOIS

Alan M. Frisch and Paul A. Giannaros.
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)*, 2010.

Heidi E. Dixon Matthew L. Ginsberg.
Inference methods for a pseudo-boolean satisfiability solver.
In *Proceedings of The Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 635–640, 2002.

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

J. N. Hooker.

Generalized resolution and cutting planes.
Ann. Oper. Res., 12(1-4):217–239, 1988.

Vasco M. Manquinho, Paulo F. Flores, João P. Marques Silva, and Arlindo L. Oliveira.
Prime implicant computation using satisfiability algorithms.
In *ICTAI*, pages 232–239, 1997.

Heidi E. Dixon Matthew L. Ginsberg Andrew J. Parkes.
Generalizing boolean satisfiability i: Background and survey of existing work.
In *Journal of Artificial Intelligence Research 21*, 2004.

Knot Pipatsrisawat and Adnan Darwiche.
On the power of clause-learning sat solvers as resolution engines.
Artif. Intell., 175(2):512–525, 2011.

S. Prestwich.

Randomised backtracking for linear pseudo-boolean constraint problems.
In *Proceedings of Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR'2002)*, pages 7–20, 2002.

S. Prestwich.
Incomplete dynamic backtracking for linear pseudo-boolean problems: Hybrid optimization techniques.
Annals of Operations Research, 130(1-4):57–73, August 2004.

Olivier Roussel and Vasco M. Manquinho.
Pseudo-boolean and cardinality constraints.
In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 695–733. IOS Press, 2009.

Fadi A. Aloul Arathi Ramani Igor L. Markov Karem A. Sakallah.
Symmetry-breaking for pseudo-boolean formulas.
In *International Workshop on Symmetry on Constraint Satisfaction Problems (SymCon)*, pages 1–12, County Cork, Ireland, 2003.

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

Hossein M. Sheini and Karem A. Sakallah.
Pueblo: A Hybrid Pseudo-Boolean SAT Solver.
Journal on Satisfiability, Boolean Modeling and Computation (JSAT), 2:165–182, 2006.

Allen Van Gelder and Ivor Spence.
Zero-one designs produce small hard sat instances.
In Ofer Strichman and Stefan Szeider, editors, *SAT*, volume 6175 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 2010.

J. P. Walser.
Solving Linear Pseudo-Boolean Constraint Problems with Local Search.
In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 269–274, 1997.

Jesse Whittemore, Joonyoung Kim, and Karem A. Sakallah.
Satire: A new incremental satisfiability engine.
In *DAC*, pages 542–545. ACM, 2001.