Improving Watched Pseudo-Boolean Propagation with Significant Literals

Mia Müßig, Jan Johannsen

Institut für Informatik, LMU Munich

August 11, 2025

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Pseudo-Boolean Problem

We have variables $x_i \in \{0, 1\}$ and literals l_i representing either x_i or $\overline{x}_i := 1 - x_i$.

$$F = C_1 \wedge \ldots \wedge C_m \quad C_j = \left[\, \sum_i a_i l_i \geq b \,|\, a_i, b \in \mathbb{N}^+ \right]$$

We will assume: $a_1 \geq a_2 \geq \ldots$

Pseudo-Boolean Problem

We have variables $x_i \in \{0, 1\}$ and literals l_i representing either x_i or $\overline{x}_i := 1 - x_i$.

$$F = C_1 \wedge \ldots \wedge C_m \quad C_j = \Big[\sum_i a_i l_i \geq b \,\Big|\, a_i, b \in \mathbb{N}^+ \Big]$$

We will assume: $a_1 \geq a_2 \geq \ldots$

- Special cases: Cardinality constraints ($\forall i : a_i = 1$), Clauses ($\forall i : a_i = 1, b = 1$)
- Efficient encoding of general Pseudo-Boolean constraints $\sum_i a_i \prod_j l_{i,j} \triangleright b$ with $a_i, b \in \mathbb{Q}$ and $\triangleright \in \{=, <, >, \leq, \geq\}$

Unit Literal

An unassigned literal l_j is unit if is the only unassigned literal remaining in an unsatisfied clause.

- A unit literal is forced by the current assignment ρ, afterwards its clause becomes satisfied

Unit Literal

An unassigned literal l_j is unit if is the only unassigned literal remaining in an unsatisfied clause.

- A unit literal is forced by the current assignment ρ, afterwards its clause becomes satisfied
- Efficient detection with two pointers "watching" two distinct unassigned literals
  - When assigning variable only update of its watched occurrences necessary
  - When unassigning variable no updates necessary

Example:

$$5x_1 + 4\overline{x}_2 + 2x_3 + 2x_4 \geq 10$$

Introduction - Unit Literals for PBS

Example:

$$5x_1 + 4\overline{x}_2 + 2x_3 + 2x_4 \geq 10$$

Slack

$$slack(C, \rho) = -b + \sum_{\bar{l}_i \notin \rho} a_i$$

Unit Literal

An unassigned literal l_j is unit if $slack(C, \rho) < a_j$.

- Contrary to SAT, one constraint can contain multiple unit literals and is not necessarily satisfied after their propagation

- Instead of two pointers now set of watched literals $W(C)$

Watchslack

$$wslack(C,\rho) = -b + \sum_{l_i \in W(C)} a_i \leq slack(C,\rho)$$

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

- Instead of two pointers now set of watched literals $W(C)$

Watchslack

$$wslack(C, \rho) = -b + \sum_{l_i \in W(C)} a_i \leq slack(C, \rho)$$

Unit Literal

No unit literals exist if and only if we can find $W(C)$ with $wslack(C, \rho) \geq a_{max}$, where a_{max} is the largest coefficient of the unassigned literals.

Unit Literal

No unit literals exist if and only if we can find $W(C)$ with $wslack(C, \rho) \geq a_{max}$, where a_{max} is the largest coefficient of the unassigned literals.

Dynamic Method:

- a_{max} needs to be updated in all occurrences when assigning and unassigning variables, worse performance than counting method

[1] Jo Devriendt. Watched Propagation of 0-1 Integer Linear Constraints. 2020.

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Unit Literal

No unit literals exist if and only if we can find $W(C)$ with $wslack(C, \rho) \geq a_{max}$, where a_{max} is the largest coefficient of the unassigned literals.

Dynamic Method:

- a_{max} needs to be updated in all occurrences when assigning and unassigning variables, worse performance than counting method

Constant Method:[1]

- Instead choose $wslack(C, \rho) \geq a_1 \geq a_{max}$, so a constant bound
  - Watches more literals than necessary
  - False Positives

[1] Jo Devriendt. Watched Propagation of 0-1 Integer Linear Constraints. 2020.

Significant Literals - Concept

- Choose watched literals with $wslack(C, \rho) \geq a_{smax}$, where a_{smax} is only updated for "significant" literals for the constraint
- Aim is to allow any constant criterion $\text{isSig}(C, a_i)$ to be used to determine if a literal is significant for a constraint

- Choose watched literals with $wslack(C, \rho) \geq a_{smax}$, where a_{smax} is only updated for "significant" literals for the constraint
- Aim is to allow any constant criterion isSig(C, a_i) to be used to determine if a literal is significant for a constraint
- If isSig(C, a_i) = true, $a_{smax} = a_{max}$ and we obtain the Dynamic method
- If isSig(C, a_i) = false, $a_{smax} = a_1$ and we obtain the Constant method

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

$$C : 100y + \sum_{i=1}^{100} x_i \geq 10$$

- After $\rho = \{\overline{y}\}$ we have $slack(C, \rho) = 90$, $a_1 = 100$ and $a_{max} = 1$

Significant Literals - Example

$$C : 100y + \sum_{i=1}^{100} x_i \geq 10$$

- After $\rho = \{\overline{y}\}$ we have $slack(C, \rho) = 90$, $a_1 = 100$ and $a_{max} = 1$
- Until next restart the Constant method always needs to watch all remaining x_i literals, while the Dynamic method only needs to watch 11 literals
- Goal is to identify variables like y with our definition of isSig(C, a_i)

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

- **Absolute Size** with cut-off $c \in \mathbb{N}$

$$\text{isSig}(C, a_i) := (a_i > c)$$

- **Absolute Max Size** with cut-off $c \in \mathbb{N}$

$$\text{isSig}(C, a_i) := (a_1 > c)$$

- **Relative Size** with cut-offs $s \in \mathbb{R}^+$ and $n \in \mathbb{N}$

$$\text{isSig}(C, a_i) := \left(a_1 > s \sum_{j=2}^{n} a_j \right)$$

- If we don't allow for significant literals in conflict constraints, we add "C input" to the definition

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

- Modification of the state of the art PB Solver ROUNDINGSAT[2], "Constant" and "Counting" data is obtained from the unmodified solver
- Evaluation on the linear instances of the Pseudo-Boolean Competition 2024 with a timeout of 3600s

[2] `https://gitlab.com/MIAOresearch/software/roundingsat`, Commit d34b6bed

Experimental Evaluation

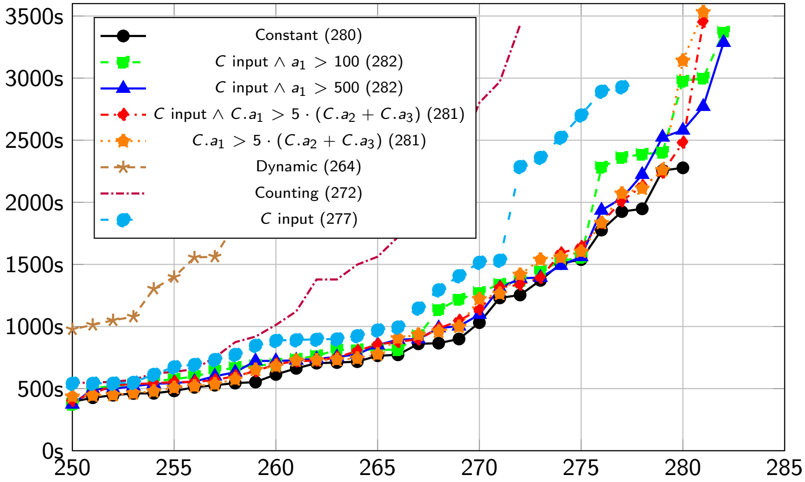

Figure: Runtime for 398 instances of the DEC-LIN track

Experimental Evaluation

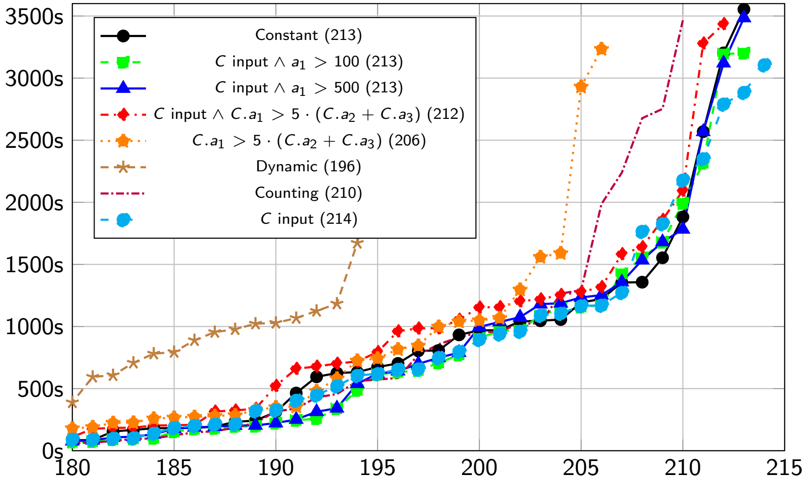

Figure: Runtime for 487 instances of the OPT-LIN track

- Significant literals leave cardinality constraints and clauses completely unaffected, which represent 96.3% of all constraints in DEC-LIN instances and 88.0% in OPT-LIN instances
- Only some optimizations developed by Devriendt for the Constant scheme are still valid for a non-constant a_{smax}

Experimental Evaluation - Coefficient Distribution

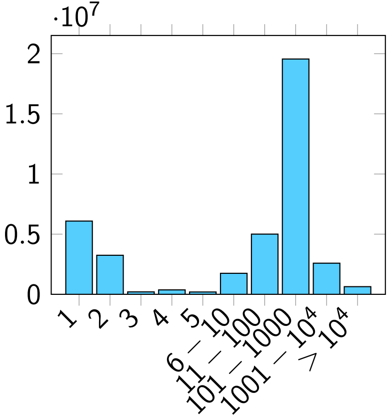

Figure: Input constraints

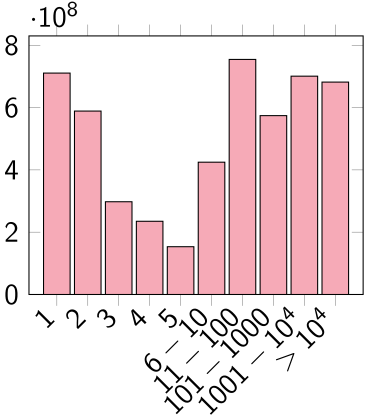

Figure: Learned constraints

Experimental Evaluation - Knapsack

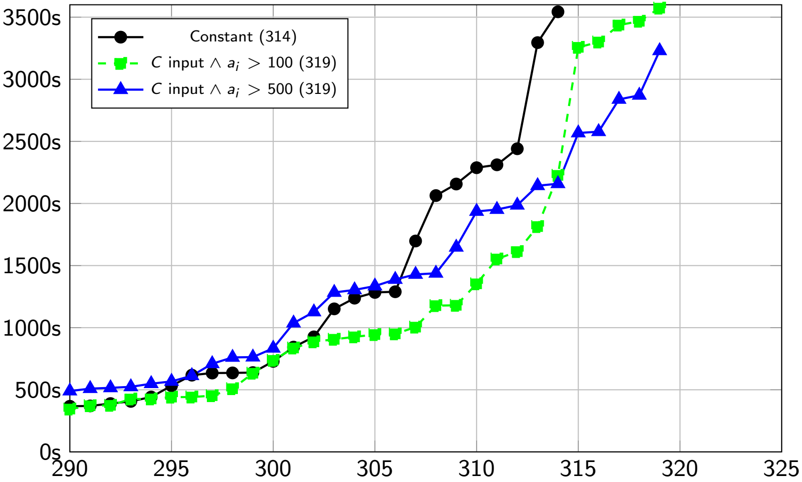

Figure: Runtime for 783 instances of the Knapsack dataset from the Pseudo-Boolean Competition

Future Work

- Experimentation with more complicated criteria for significance
- Adaptation of the logging method[3] to reliable identify small performance improvements
- Choosing cut-off values per instance during the preprocessing step

[3] Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Rui Zhao. Speeding up Pseudo-Boolean Propagation. 2024.