Certified MaxSAT Preprocessing

Hannes Ihalainen[1] Andy Oertel[2,3] Yong Kiam Tan[4] Jeremias Berg[1]
Matti Järvisalo[1] Magnus O. Myreen[5] Jakob Nordström[3,2]

[1]University of Helsinki [2]Lund University [3]University of Copenhagen

[4]Institute for Infocomm Research (I^{2}R), A*STAR [5]Chalmers University of Technology

15th Pragmatics of SAT International Workshop
Pune, India
August 20, 2024

Our work on one slide

- Success-story of SAT solving:
  - Solvers are fast
  - **And they produce proofs**
- Certifying SAT-based optimization has remained a challenge
  - Many proposals, e.g., [BLM07, LNOR11, MM11, MIB+19, FMSV20, PCH20, PCH21]
  - Proof logging for state-of-the-art MaxSAT only very recently [VDB22, BBN+23, BBN+24]
  - And only for main solver algorithm after preprocessing
- **Contribution of this work:**
  - Proof logging for standalone MaxSAT preprocessor
  - Proofs for equioptimality (and equisatisfiability) with VeriPB
  - Formally verified checker CakePB for the proofs

Our work on one slide

- Success-story of SAT solving:
  - Solvers are fast
  - **And they produce proofs**
- Certifying SAT-based optimization has remained a challenge
  - Many proposals, e.g., [BLM07, LNOR11, MM11, MIB+19, FMSV20, PCH20, PCH21]
  - Proof logging for state-of-the-art MaxSAT only very recently [VDB22, BBN+23, BBN+24]
  - And only for main solver algorithm after preprocessing
- **Contribution of this work:**
  - Proof logging for standalone MaxSAT preprocessor
  - Proofs for equioptimality (and equisatisfiability) with VeriPB
  - Formally verified checker CakePB for the proofs

Our work on one slide

- Success-story of SAT solving:
  - Solvers are fast
  - **And they produce proofs**
- Certifying SAT-based optimization has remained a challenge
  - Many proposals, e.g., [BLM07, LNOR11, MM11, MIB^{+}19, FMSV20, PCH20, PCH21]
  - Proof logging for state-of-the-art MaxSAT only very recently [VDB22, BBN^{+}23, BBN^{+}24]
  - And only for main solver algorithm after preprocessing
- **Contribution of this work:**
  - Proof logging for standalone MaxSAT preprocessor
  - Proofs for equioptimality (and equisatisfiability) with VERIPB
  - Formally verified checker CAKEPB for the proofs

Our contribution

- Proof logging for standalone MaxSAT preprocessor MaxPre
  - 15+ different preprocessing techniques certified with VeriPB
- Updated VeriPB proof format
  - Support for equioptimality (and equisatisfiability) proofs
- Formal verification with CakePB
  - HOL4 proof assistant
  - CakeML tools
- Experimental evaluation

Bounded Variable Elimination

Unit Propagation

Structure-Based Labeling

Subsumed Label Elimination

Self-Subsuming Resolution

Hidden Literal Elimination

Intrinsic At-most-ones

Subsumed Literal Elimination

TrimMaxSAT

Failed Literals

Equivalent Literal Detection

Binary Core Removal

Label Matching

Hardening

Subsumption Elimination

Blocked Clause Elimination

Hidden Tautology Elimination

Variable Instantiation

Bounded Variable Addition

Backbones

Generalized Subsumed Label Elimination

SAT-based and MaxSAT-specific preprocessing techniques certified

Our contribution

- Proof logging for standalone MaxSAT preprocessor MAXPRE
  - 15+ different preprocessing techniques certified with VERIPB
- Updated VERIPB proof format
  - Support for equioptimality (and equisatisfiability) proofs
- Formal verification with CAKEPB
  - HOL4 proof assistant
  - CAKEML tools
- Experimental evaluation

Bounded Variable Elimination
Unit Propagation
Structure-Based Labeling
Subsumed Label Elimination
Self-Subsuming Resolution
Hidden Literal Elimination
Intrinsic At-most-ones
Subsumed Literal Elimination
TrimMaxSAT
Failed Literals
Equivalent Literal Detection
Binary Core Removal
Hardening
Label Matching
Subsumption Elimination
Blocked Clause Elimination
Hidden Tautology Elimination
Variable Instantiation
Bounded Variable Addition
Backbones
Generalized Subsumed Label Elimination

SAT-based and MaxSAT-specific preprocessing techniques certified

Our contribution

- Proof logging for standalone MaxSAT preprocessor MAXPRE
  - 15+ different preprocessing techniques certified with VERIPB
- Updated VERIPB proof format
  - Support for equioptimality (and equisatisfiability) proofs
- Formal verification with CAKEPB
  - HOL4 proof assistant
  - CAKEML tools
- Experimental evaluation

Bounded Variable Elimination

Unit Propagation

Structure-Based Labeling

Subsumed Label Elimination

Self-Subsuming Resolution

Hidden Literal Elimination

Intrinsic At-most-ones

Subsumed Literal Elimination

TrimMaxSAT

Failed Literals

Equivalent Literal Detection

Binary Core Removal

Hardening

Label Matching

Subsumption Elimination

Blocked Clause Elimination

Hidden Tautology Elimination

Variable Instantiation

Bounded Variable Addition

Backbones

Generalized Subsumed Label Elimination

SAT-based and MaxSAT-specific preprocessing techniques certified

Our contribution

- Proof logging for standalone MaxSAT preprocessor MAXPRE
  - 15+ different preprocessing techniques certified with VERIPB
- Updated VERIPB proof format
  - Support for equioptimality (and equisatisfiability) proofs
- Formal verification with CAKEPB
  - HOL4 proof assistant
  - CAKEML tools
- Experimental evaluation

Bounded Variable Elimination
Unit Propagation
Structure-Based Labeling
Subsumed Label Elimination
Self-Subsuming Resolution
Hidden Literal Elimination
Intrinsic At-most-ones
Subsumed Literal Elimination
TrimMaxSAT
Failed Literals
Equivalent Literal Detection
Binary Core Removal
Hardening
Label Matching
Subsumption Elimination
Blocked Clause Elimination
Hidden Tautology Elimination
Variable Instantiation
Bounded Variable Addition
Backbones
Generalized Subsumed Label Elimination

SAT-based and MaxSAT-specific preprocessing techniques certified

Traditional view of maximum satisfiability (MaxSAT)

Weighted CNF (WCNF) representation

- Hard clauses
- Weighted soft clauses
  - $\langle(\bar{x}_2 \vee \bar{x}_3), 2\rangle$: "incur cost 2 if $(\bar{x}_2 \vee \bar{x}_3)$ is falsified"

Traditional view of maximum satisfiability (MaxSAT)

Weighted CNF (WCNF) representation

- Hard clauses
- Weighted soft clauses
  - $\langle(\bar{x}_2 \vee \bar{x}_3), 2\rangle$: "incur cost 2 if $(\bar{x}_2 \vee \bar{x}_3)$ is falsified"

Example of WCNF instance

$\mathcal{F} = (F_H, F_S)$
$F_H = \{(x_1 \vee \bar{x}_2), (x_2 \vee \bar{x}_3), (x_3 \vee \bar{x}_1)\}$
$F_S = \{\langle(x_1), 1\rangle, \langle(\bar{x}_2 \vee \bar{x}_3), 2\rangle\}$

Traditional view of maximum satisfiability (MaxSAT)

Weighted CNF (WCNF) representation

- Hard clauses
- Weighted soft clauses
  - $\langle(\bar{x}_2 \vee \bar{x}_3), 2\rangle$: "incur cost 2 if $(\bar{x}_2 \vee \bar{x}_3)$ is falsified"

Example of WCNF instance

$\mathcal{F} = (F_H, F_S)$
$F_H = \{(x_1 \vee \bar{x}_2), (x_2 \vee \bar{x}_3), (x_3 \vee \bar{x}_1)\}$
$F_S = \{\langle(x_1), 1\rangle, \langle(\bar{x}_2 \vee \bar{x}_3), 2\rangle\}$

Conversion using blocking variables

$\mathcal{F}^b = (F_H^b, F_S^b)$
$F_H^b = \{(x_1 \vee \bar{x}_2), (x_2 \vee \bar{x}_3), (x_3 \vee \bar{x}_1), (\bar{x}_2 \vee \bar{x}_3 \vee x_4)\}$
$F_S^b = \{\langle(x_1), 1\rangle, \langle(\bar{x}_4), 2\rangle\}$

Traditional view of maximum satisfiability (MaxSAT)

Weighted CNF (WCNF) representation

- Hard clauses
- Weighted soft clauses
  - $\langle(\bar{x}_2 \vee \bar{x}_3), 2\rangle$: "incur cost 2 if $(\bar{x}_2 \vee \bar{x}_3)$ is falsified"

Example of WCNF instance

$\mathcal{F} = (F_H, F_S)$
$F_H = \{(x_1 \vee \bar{x}_2), (x_2 \vee \bar{x}_3), (x_3 \vee \bar{x}_1)\}$
$F_S = \{\langle(x_1), 1\rangle, \langle(\bar{x}_2 \vee \bar{x}_3), 2\rangle\}$

Conversion using blocking variables

$\mathcal{F}^b = (F_H^b, F_S^b)$
$F_H^b = \{(x_1 \vee \bar{x}_2), (x_2 \vee \bar{x}_3), (x_3 \vee \bar{x}_1), (\bar{x}_2 \vee \bar{x}_3 \vee x_4)\}$
$F_S^b = \{\langle(x_1), 1\rangle, \langle(\bar{x}_4), 2\rangle\}$

Same as satisfying F_H^b while minimizing objective $O = \bar{x}_1 + 2x_4$

Modern objective-centric view of MaxSAT

- Optimization variant of SAT
  - CNF formula F
  - Linear objective function O
- Minimize value of O subject to F,

Example:

$F = \{(x_1 \vee \bar{x}_2), (x_2 \vee \bar{x}_3), (x_3 \vee \bar{x}_1), (\bar{x}_2 \vee \bar{x}_3 \vee x_4)\}$
$O = \bar{x}_1 + 2x_4$

- Applications in:
  - Planning
  - Scheduling
  - Configuration
  - Artificial intelligence
  - Combinatorial problems
  - Verification and security
  - Bioinformatics
  - $\ldots$

Modern objective-centric view of MaxSAT

- Optimization variant of SAT
  - CNF formula F
  - Linear objective function O
- Minimize value of O subject to F,

Example:

$F = \{(x_1 \vee \bar{x}_2), (x_2 \vee \bar{x}_3), (x_3 \vee \bar{x}_1), (\bar{x}_2 \vee \bar{x}_3 \vee x_4)\}$

$O = \bar{x}_1 + 2x_4$

There are three solutions:

$\tau_1 = \{x_1 \to 1, x_2 \to 1, x_3 \to 1, x_4 \to 1\}$

$O(\tau_1) = 2$,

- Applications in:
  - Planning
  - Scheduling
  - Configuration
  - Artificial intelligence
  - Combinatorial problems
  - Verification and security
  - Bioinformatics
  - ...

Modern objective-centric view of MaxSAT

- Optimization variant of SAT
  - CNF formula F
  - Linear objective function O
- Minimize value of O subject to F,

Example:

$F = \{(x_1 \lor \bar{x}_2), (x_2 \lor \bar{x}_3), (x_3 \lor \bar{x}_1), (\bar{x}_2 \lor \bar{x}_3 \lor x_4)\}$
$O = \bar{x}_1 + 2\bar{x}_4$

There are three solutions:

$\tau_1 = \{x_1 \to 1, x_2 \to 1, x_3 \to 1, x_4 \to 1\}$
$\tau_2 = \{x_1 \to 0, x_2 \to 0, x_3 \to 0, x_4 \to 1\}$

$O(\tau_1) = 2$, $O(\tau_2) = 3$,

- Applications in:
  - Planning
  - Scheduling
  - Configuration
  - Artificial intelligence
  - Combinatorial problems
  - Verification and security
  - Bioinformatics
  - ...

Modern objective-centric view of MaxSAT

- Optimization variant of SAT
  - CNF formula F
  - Linear objective function O
- Minimize value of O subject to F,

Example:

$F = \{(x_1 \lor \bar{x}_2), (x_2 \lor \bar{x}_3), (x_3 \lor \bar{x}_1), (\bar{x}_2 \lor \bar{x}_3 \lor x_4)\}$

$O = \bar{x}_1 + 2x_4$

There are three solutions:

$\tau_1 = \{x_1 \rightarrow 1, x_2 \rightarrow 1, x_3 \rightarrow 1, x_4 \rightarrow 1\}$

$\tau_2 = \{x_1 \rightarrow 0, x_2 \rightarrow 0, x_3 \rightarrow 0, x_4 \rightarrow 1\}$

$\tau_3 = \{x_1 \rightarrow 0, x_2 \rightarrow 0, x_3 \rightarrow 0, x_4 \rightarrow 0\}$

$O(\tau_1) = 2$, $O(\tau_2) = 3$, $O(\tau_3) = 1$

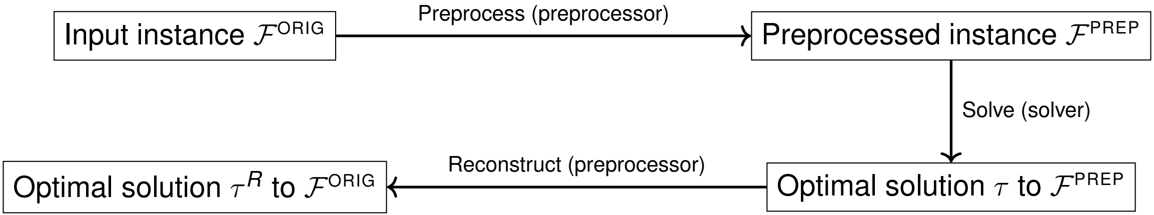
- Applications in:
  - Planning
  - Scheduling
  - Configuration
  - Artificial intelligence
  - Combinatorial problems
  - Verification and security
  - Bioinformatics
  - ...

MaxSAT preprocessing

- Simplify instance before solving
  - Remove clauses
  - Introduce new clauses
  - Change the objective function
- In MaxSAT solvers or by a **standalone preprocessor**

MaxSAT preprocessing

- Simplify instance before solving
  - Remove clauses
  - Introduce new clauses
  - Change the objective function
- In MaxSAT solvers or by a **standalone preprocessor**



- Certified preprocessing: Verify equioptimality of $\mathcal{F}^{\text{ORIG}}$ and $\mathcal{F}^{\text{PREP}}$

MaxSAT preprocessing

- Simplify instance before solving
  - Remove clauses
  - Introduce new clauses
  - Change the objective function
- In MaxSAT solvers or by a **standalone preprocessor**

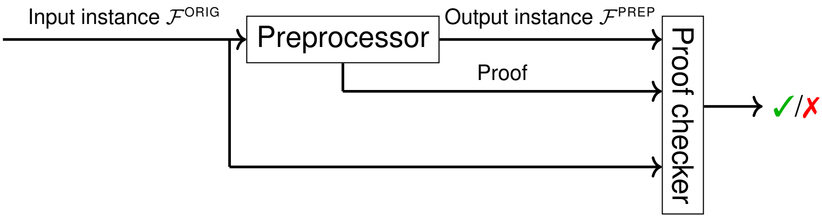
$$\begin{tikzcd}
\boxed{\text{Input instance } \mathcal{F}^{\mathrm{ORIG}}} \arrow[r, "\text{Preprocess (preprocessor)}"] & \boxed{\text{Preprocessed instance } \mathcal{F}^{\mathrm{PREP}}} \arrow[d, "\text{Solve (solver)}"] \\
\boxed{\text{Optimal solution } \tau^{R} \text{ to } \mathcal{F}^{\mathrm{ORIG}}} & \boxed{\text{Optimal solution } \tau \text{ to } \mathcal{F}^{\mathrm{PREP}}} \arrow[l, "\text{Reconstruct (preprocessor)}"']
\end{tikzcd}$$

- Certified preprocessing: Verify equioptimality of $\mathcal{F}^{\mathrm{ORIG}}$ and $\mathcal{F}^{\mathrm{PREP}}$

Concrete workflow of MaxPre MaxSAT preprocessor

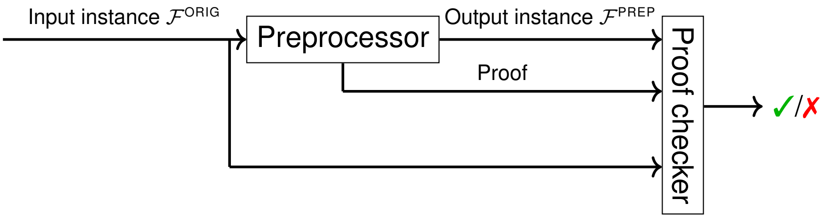
1. Reading of input
2. Preprocessing on WCNF
3. Conversion to objective-centric
4. Preprocessing on objective-centric
5. Removing constant from the objective function + renaming variables
6. Writing of output

How to verify equioptimality



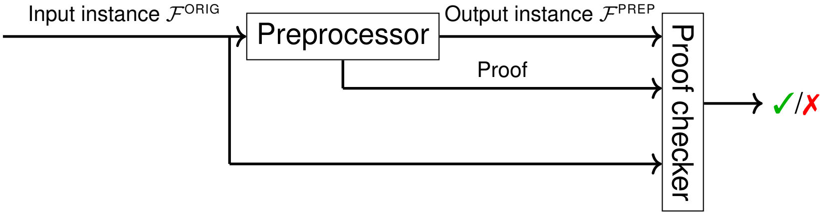
- Proof shows how to derive $\mathcal{F}^{\text{PREP}}$ from $\mathcal{F}^{\text{ORIG}}$
  - Add constraints
  - Remove constraints
  - Change the objective function
- Proof checker verifies that
  - All steps are sound (optimal cost is preserved)
  - In the end, the database is identical to the output instance

How to verify equioptimality



- Proof shows how to derive $\mathcal{F}^{\text{PREP}}$ from $\mathcal{F}^{\text{ORIG}}$
  - Add constraints
  - Remove constraints
  - Change the objective function
- Proof checker verifies that
  - All steps are sound (optimal cost is preserved)
  - In the end, the database is identical to the output instance

How to verify equioptimality



- Proof shows how to derive $\mathcal{F}^{\mathrm{PREP}}$ from $\mathcal{F}^{\mathrm{ORIG}}$
  - Add constraints
  - Remove constraints
  - Change the objective function
- Proof checker verifies that
  - All steps are sound (optimal cost is preserved)
  - In the end, the database is identical to the output instance

Which proof system to use for MaxSAT?

- MaxSAT: Formula in CNF
  - Use SAT proof logging techniques (DRAT)?
    - We also have the objective function
  - Extend SAT proof systems to MaxSAT?
    - Difficult for actual solvers to produce proofs
- Objective-function essentially pseudo-Boolean
  - Use pseudo-Boolean proof system
  - PB not too far from MaxSAT
- VeriPB simple but powerful, recent successes:
  - Advanced SAT techniques [GN21, GMNO22, BGMN23]
  - **MaxSAT solving** [VDB22, BBN+23, BBN+24]
  - Constraint programming [EGMN20, GMN22, MM23, MMN24]
  - Subgraph solving [GMM+20, GMN20, GMM+24]
  - Presolving for 0-1 integer linear programming [HOGN24]
  - Dynamic programming and decision diagrams [DMM+24]

Which proof system to use for MaxSAT?

- MaxSAT: Formula in CNF
  - Use SAT proof logging techniques (DRAT)?
    - We also have the objective function
  - Extend SAT proof systems to MaxSAT?
    - Difficult for actual solvers to produce proofs
- Objective-function essentially pseudo-Boolean
  - Use pseudo-Boolean proof system
  - PB not too far from MaxSAT
- VeriPB simple but powerful, recent successes:
  - Advanced SAT techniques [GN21, GMNO22, BGMN23]
  - **MaxSAT solving** [VDB22, BBN+23, BBN+24]
  - Constraint programming [EGMN20, GMN22, MM23, MMN24]
  - Subgraph solving [GMM+20, GMN20, GMM+24]
  - Presolving for 0-1 integer linear programming [HOGN24]
  - Dynamic programming and decision diagrams [DMM+24]

Which proof system to use for MaxSAT?

- MaxSAT: Formula in CNF
  - Use SAT proof logging techniques (DRAT)?
    - We also have the objective function
  - Extend SAT proof systems to MaxSAT?
    - Difficult for actual solvers to produce proofs
- Objective-function essentially pseudo-Boolean
  - Use pseudo-Boolean proof system
  - PB not too far from MaxSAT
- VeriPB simple but powerful, recent successes:
  - Advanced SAT techniques [GN21, GMNO22, BGMN23]
  - **MaxSAT solving** [VDB22, BBN+23, BBN+24]
  - Constraint programming [EGMN20, GMN22, MM23, MMN24]
  - Subgraph solving [GMM+20, GMN20, GMM+24]
  - Presolving for 0-1 integer linear programming [HOGN24]
  - Dynamic programming and decision diagrams [DMM+24]

VeriPB proof system [BGMN23, GN21]

Cutting planes method [CCT87]

Addition

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3 \qquad x_3 + \bar{x}_4 \geq 1}{3x_1 + 2\bar{x}_2 + 2x_3 + \bar{x}_4 \geq 4}$$

Divide (here by 2)

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3}{\left\lceil \frac{3}{2} \right\rceil x_1 + \bar{x}_2 + \left\lceil \frac{1}{2} \right\rceil x_3 \geq \left\lceil \frac{3}{2} \right\rceil}$$

etc.

VeriPB proof system [BGMN23, GN21]

Cutting planes method [CCT87]

Addition

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3 \qquad x_3 + \bar{x}_4 \geq 1}{3x_1 + 2\bar{x}_2 + 2x_3 + \bar{x}_4 \geq 4}$$

Divide (here by 2)

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3}{\left\lceil \frac{3}{2} \right\rceil x_1 + \bar{x}_2 + \left\lceil \frac{1}{2} \right\rceil x_3 \geq \left\lceil \frac{3}{2} \right\rceil}$$

etc.

Reverse unit propagation (RUP)

$$F = \{x + y \geq 1, x + \bar{y} \geq 1\}$$

Introduce $x \geq 1$,
$\bar{x} \geq 1$ unit propagates to a conflict

VeriPB proof system [BGMN23, GN21]

Cutting planes method [CCT87]

Addition

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3 \qquad x_3 + \bar{x}_4 \geq 1}{3x_1 + 2\bar{x}_2 + 2x_3 + \bar{x}_4 \geq 4}$$

Divide (here by 2)

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3}{\left\lceil \frac{3}{2} \right\rceil x_1 + \bar{x}_2 + \left\lceil \frac{1}{2} \right\rceil x_3 \geq \left\lceil \frac{3}{2} \right\rceil}$$

etc.

Reverse unit propagation (RUP)

$F = \{x + y \geq 1, x + \bar{y} \geq 1\}$

Introduce $x \geq 1$,
$\bar{x} \geq 1$ unit propagates to a conflict

Redundance-based strengthening

$F = \{\bar{x}_1 + \bar{x}_2 \geq 1\}$

Introduce $x_1 + x_2 + x_3 \geq 2$
witness $\omega = \{x_1 \to \bar{x}_2, x_3 \to 1\}$

VeriPB proof system [BGMN23, GN21]

Cutting planes method [CCT87]

Addition

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3 \qquad x_3 + \bar{x}_4 \geq 1}{3x_1 + 2\bar{x}_2 + 2x_3 + \bar{x}_4 \geq 4}$$

Divide (here by 2)

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3}{\left\lceil \frac{3}{2} \right\rceil x_1 + \bar{x}_2 + \left\lceil \frac{1}{2} \right\rceil x_3 \geq \left\lceil \frac{3}{2} \right\rceil}$$

etc.

Reverse unit propagation (RUP)

$F = \{x + y \geq 1, x + \bar{y} \geq 1\}$

Introduce $x \geq 1$,
$\bar{x} \geq 1$ unit propagates to a conflict

Redundance-based strengthening

$F = \{\bar{x}_1 + \bar{x}_2 \geq 1\}$

Introduce $x_1 + x_2 + x_3 \geq 2$
witness $\omega = \{x_1 \rightarrow \bar{x}_2, x_3 \rightarrow 1\}$

(Checked) deletion

Delete a constraint only if we can rederive it

VeriPB proof system [BGMN23, GN21]

Cutting planes method [CCT87]

ADDITION

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3 \qquad x_3 + \bar{x}_4 \geq 1}{3x_1 + 2\bar{x}_2 + 2x_3 + \bar{x}_4 \geq 4}$$

DIVIDE (HERE BY 2)

$$\frac{3x_1 + 2\bar{x}_2 + x_3 \geq 3}{\left\lceil \frac{3}{2} \right\rceil x_1 + \bar{x}_2 + \left\lceil \frac{1}{2} \right\rceil x_3 \geq \left\lceil \frac{3}{2} \right\rceil}$$

etc.

Reverse unit propagation (RUP)

$F = \{x + y \geq 1, x + \bar{y} \geq 1\}$

Introduce $x \geq 1$,
$\bar{x} \geq 1$ unit propagates to a conflict

Redundance-based strengthening

$F = \{\bar{x}_1 + \bar{x}_2 \geq 1\}$

Introduce $x_1 + x_2 + x_3 \geq 2$
witness $\omega = \{x_1 \to \bar{x}_2, x_3 \to 1\}$

(Checked) deletion

Delete a constraint only if we can rederive it

Update the objective function

If we can prove $O^{\text{OLD}} = O^{\text{NEW}}$

Practical example:

Input instance (MaxSAT):

$(x_1 \vee x_2 \vee x_3)$
$(\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)$
$(x_2 \vee x_4)$
$(x_3 \vee \bar{x}_5)$
$(x_1 \vee \bar{x}_5)$
$(x_3 \vee \bar{x}_4)$

$O = x_1 + 2x_3 + \bar{x}_5$

In proof (pseudo-Boolean optimization):

$x_1 + x_2 + x_3 \geq 1$
$\bar{x}_2 + \bar{x}_4 + \bar{x}_5 \geq 1$
$x_2 + x_4 \geq 1$
$x_3 + \bar{x}_5 \geq 1$
$x_1 + \bar{x}_5 \geq 1$
$x_3 + \bar{x}_4 \geq 1$

$O = x_1 + 2x_3 + \bar{x}_5$

Practical example: Autarky detection / Subsumed literal elimination

- Consider the MaxSAT instance:
  - $F = \{(x_1 \lor x_2 \lor x_3)^1, (\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)^2, (x_2 \lor x_4)^3, (x_3 \lor \bar{x}_5)^4, (x_1 \lor \bar{x}_5)^5, (x_3 \lor \bar{x}_4)^6\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- Consider literals x_2 and $\bar{x}_4$
- Remove all clauses containing x_2 or x_4 (by fixing $x_2 = 1$, $x_4 = 0$)

Practical example: Autarky detection / Subsumed literal elimination

- Consider the MaxSAT instance:
  - $F = \{(x_1 \vee x_2 \vee x_3)^1, (\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)^2, (x_2 \vee x_4)^3, (x_3 \vee \bar{x}_5)^4, (x_1 \vee \bar{x}_5)^5, (x_3 \vee \bar{x}_4)^6\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- Consider literals x_2 and $\bar{x}_4$
- Remove all clauses containing x_2 or x_4 (by fixing $x_2 = 1$, $x_4 = 0$)

Practical example: Autarky detection / Subsumed literal elimination

- Consider the MaxSAT instance:
  - $F = \{(x_1 \vee x_2 \vee x_3)^1, (\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)^2, (x_2 \vee x_4)^3, (x_3 \vee \bar{x}_5)^4, (x_1 \vee \bar{x}_5)^5, (x_3 \vee \bar{x}_4)^6\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- Consider literals x_2 and $\bar{x}_4$
- Remove all clauses containing x_2 or x_4 (by fixing $x_2 = 1$, $x_4 = 0$)

Practical example: Autarky detection / Subsumed literal elimination

- Consider the MaxSAT instance:
  - $F = \{(x_1 \lor x_2 \lor x_3)^1, (\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)^2, (x_2 \lor x_4)^3, (x_3 \lor \bar{x}_5)^4, (x_1 \lor \bar{x}_5)^5, (x_3 \lor \bar{x}_4)^6\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- Consider literals x_2 and $\bar{x}_4$
- Remove all clauses containing x_2 or x_4 (by fixing $x_2 = 1$, $x_4 = 0$)

Proof

- Introduce $x_2 \geq 1, \omega = \{x_4 \to 0, x_2 \to 1\}$
- Introduce $\bar{x}_4 \geq 1, \omega = \{x_4 \to 0, x_2 \to 1\}$
- Delete clauses where x_4 or x_2 appear (RUP)
- Delete $x_2 \geq 1, \omega = \{x_2 \to 1\}$
- Delete $\bar{x}_4 \geq 1, \omega = \{x_4 \to 0\}$

```
red +1 x2 >= 1 ; x4 -> 0 x2 -> 1
core id 7
red +1 ~x4 >= 1 ; x4 -> 0 x2 -> 1
core id 8
```

Practical example: Autarky detection / Subsumed literal elimination

- Consider the MaxSAT instance:
  - $F = \{ \quad (x_3 \vee \bar{x}_5)^4, (x_1 \vee \bar{x}_5)^5 \quad \}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- Consider literals x_2 and $\bar{x}_4$
- Remove all clauses containing x_2 or x_4 (by fixing $x_2 = 1$, $x_4 = 0$)

Proof

- Introduce $x_2 \geq 1, \omega = \{x_4 \to 0, x_2 \to 1\}$
- Introduce $\bar{x}_4 \geq 1, \omega = \{x_4 \to 0, x_2 \to 1\}$
- Delete clauses where x_4 or x_2 appear (RUP)
- Delete $x_2 \geq 1, \omega = \{x_2 \to 1\}$
- Delete $\bar{x}_4 \geq 1, \omega = \{x_4 \to 0\}$

```
red +1 x2 >= 1 ; x4 -> 0 x2 -> 1
core id 7
red +1 ~x4 >= 1 ; x4 -> 0 x2 -> 1
core id 8


del id 1
del id 2
del id 3
del id 6
```

Practical example: Autarky detection / Subsumed literal elimination

- Consider the MaxSAT instance:
  - $F = \{ \qquad (x_3 \vee \bar{x}_5)^4, (x_1 \vee \bar{x}_5)^5 \qquad \}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- Consider literals x_2 and $\bar{x}_4$
- Remove all clauses containing x_2 or x_4 (by fixing $x_2 = 1$, $x_4 = 0$)

Proof

- Introduce $x_2 \geq 1, \omega = \{x_4 \to 0, x_2 \to 1\}$
- Introduce $\bar{x}_4 \geq 1, \omega = \{x_4 \to 0, x_2 \to 1\}$
- Delete clauses where x_4 or x_2 appear (RUP)
- Delete $x_2 \geq 1, \omega = \{x_2 \to 1\}$
- Delete $\bar{x}_4 \geq 1, \omega = \{x_4 \to 0\}$

```
red +1 x2 >= 1 ; x4 -> 0 x2 -> 1
core id 7
red +1 ~x4 >= 1 ; x4 -> 0 x2 -> 1
core id 8


del id 1
del id 2
del id 3
del id 6


del id 7 ; x2 -> 1
del id 8 ; x4 -> 0
```

Practical example: Hardening

- Continue with the formula
  - $F = \{(x_3 \vee \bar{x}_5)^4, (x_1 \vee \bar{x}_5)^5\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- There is a solution $\tau = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$, $O(\tau) = 1$
  - Definitely no optimal solution sets $x_3 = 1$
  - We can fix $x_3 = 0$

Practical example: Hardening

- Continue with the formula
  - $F = \{(x_3 \vee \bar{x}_5)^4, (x_1 \vee \bar{x}_5)^5\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- There is a solution $\tau = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$, $O(\tau) = 1$
  - Definitely no optimal solution sets $x_3 = 1$
  - We can fix $x_3 = 0$

Practical example: Hardening

- Continue with the formula
  - $F = \{(x_3 \vee \bar{x}_5)^4, (x_1 \vee \bar{x}_5)^5\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- There is a solution $\tau = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$, $O(\tau) = 1$
  - Definitely no optimal solution sets $x_3 = 1$
  - We can fix $x_3 = 0$

Practical example: Hardening

- Continue with the formula
  - $F = \{(x_3 \lor \bar{x}_5)^4, (x_1 \lor \bar{x}_5)^5\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- There is a solution $\tau = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$, $O(\tau) = 1$
  - Definitely no optimal solution sets $x_3 = 1$
  - We can fix $x_3 = 0$

Proof

- Introduce $\bar{x}_3 \geq 1$, $\omega = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$
- Remove x_3 from the objective function
- Remove x_3 from the clauses
  - Introduce $\bar{x}_5 \geq 1$ (RUP)
  - Delete $(x_3 \lor \bar{x}_5)$
- Delete $\bar{x}_3 \geq 1$, $\omega = \{x_3 \to 0\}$

```
red +1 ~x3 >= 1 ; x1 -> 0 x3 -> 0 x5 -> 0
core id 9
```

Practical example: Hardening

- Continue with the formula
  - $F = \{(x_3 \vee \bar{x}_5)^4, (x_1 \vee \bar{x}_5)^5\}$
  - $O = x_1 + 2x_3 + \bar{x}_5$
- There is a solution $\tau = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$, $O(\tau) = 1$
  - Definitely no optimal solution sets $x_3 = 1$
  - We can fix $x_3 = 0$

Proof

- Introduce $\bar{x}_3 \geq 1$, $\omega = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$
- Remove x_3 from the objective function
- Remove x_3 from the clauses
  - Introduce $\bar{x}_5 \geq 1$ (RUP)
  - Delete $(x_3 \vee \bar{x}_5)$
- Delete $\bar{x}_3 \geq 1$, $\omega = \{x_3 \to 0\}$

```
red +1 ~x3 >= 1 ; x1 -> 0 x3 -> 0 x5 -> 0
core id 9

obju diff -2 x3
```

Practical example: Hardening

- Continue with the formula
  - $F = \{(x_3 \lor \bar{x}_5)^4, (x_1 \lor \bar{x}_5)^5, (\bar{x}_5)^{10}\}$
  - $O = x_1 \qquad\qquad + \bar{x}_5$
- There is a solution $\tau = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$, $O(\tau) = 1$
  - Definitely no optimal solution sets $x_3 = 1$
  - We can fix $x_3 = 0$

Proof

- Introduce $\bar{x}_3 \geq 1$, $\omega = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$
- Remove x_3 from the objective function
- Remove x_3 from the clauses
  - Introduce $\bar{x}_5 \geq 1$ (RUP)
  - Delete $(x_3 \lor \bar{x}_5)$
- Delete $\bar{x}_3 \geq 1$, $\omega = \{x_3 \to 0\}$

```
red +1 ~x3 >= 1 ; x1 -> 0 x3 -> 0 x5 -> 0
core id 9

obju diff -2 x3

rup 1 ~x5 >= 1 ;
core id 10
del id 4
```

Practical example: Hardening

- Continue with the formula
  - $F = \{ \quad (x_1 \vee \bar{x}_5)^5, (\bar{x}_5)^{10}\}$
  - $O = x_1 \quad + \bar{x}_5$
- There is a solution $\tau = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$, $O(\tau) = 1$
  - Definitely no optimal solution sets $x_3 = 1$
  - We can fix $x_3 = 0$

Proof

- Introduce $\bar{x}_3 \geq 1$, $\omega = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$
- Remove x_3 from the objective function
- Remove x_3 from the clauses
  - Introduce $\bar{x}_5 \geq 1$ (RUP)
  - Delete $(x_3 \vee \bar{x}_5)$
- Delete $\bar{x}_3 \geq 1$, $\omega = \{x_3 \to 0\}$

```
red +1 ~x3 >= 1 ; x1 -> 0 x3 -> 0 x5 -> 0
core id 9

obju diff -2 x3

rup 1 ~x5 >= 1 ;
core id 10
del id 4

del id 9 ; x3 -> 0
```

Practical example: Hardening

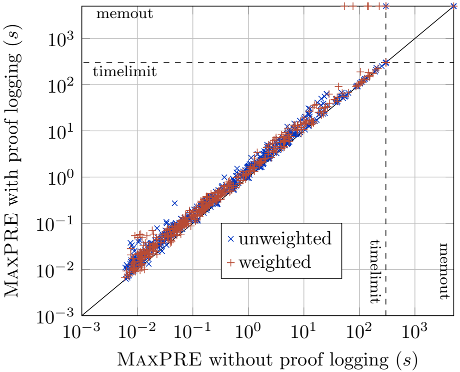
- Continue with the formula
  - $F = \{ \qquad\qquad (x_1 \vee \bar{x}_5)^5, (\bar{x}_5)^{10}\}$
  - $O = x_1 \qquad + \bar{x}_5$
- There is a solution $\tau = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$, $O(\tau) = 1$
  - Definitely no optimal solution sets $x_3 = 1$
  - We can fix $x_3 = 0$

Proof

- Introduce $\bar{x}_3 \geq 1$, $\omega = \{x_1 \to 0, x_3 \to 0, x_5 \to 0\}$
- Remove x_3 from the objective function
- Remove x_3 from the clauses
  - Introduce $\bar{x}_5 \geq 1$ (RUP)
  - Delete $(x_3 \vee \bar{x}_5)$
- Delete $\bar{x}_3 \geq 1$, $\omega = \{x_3 \to 0\}$

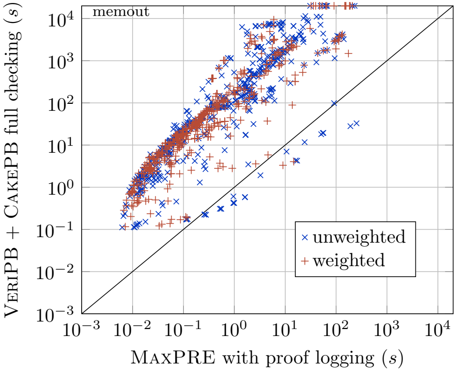
```
red +1 ~x3 >= 1 ; x1 -> 0 x3 -> 0 x5 -> 0
core id 9

obju diff -2 x3

rup 1 ~x5 >= 1 ;
core id 10
del id 4

del id 9 ; x3 -> 0
```

Overhead of proof logging



Overhead of proof logging 46% (geometric mean)

Overhead of formally verified proof checking



Formally verified checking 113 times slower than preprocessing (geometric mean)

Discussion of performance

Putting things in perspective

- Preprocessing only small part of solving time
- So even pretty bad overhead can be negligible in the grand scheme of things
- Still, it is an important challenge to make proof checking faster

Potential for improvements

- Lots of software engineering-level things to improve
- Faster unit propagation
- Binary proof format
- But this is not the full story

Discussion of performance

Putting things in perspective

- Preprocessing only small part of solving time
- So even pretty bad overhead can be negligible in the grand scheme of things
- Still, it is an important challenge to make proof checking faster

Potential for improvements

- Lots of software engineering-level things to improve
- Faster unit propagation
- Binary proof format
- But this is not the full story

Difference between SAT and MaxSAT preprocessing?

SAT preprocessing techniques

- Are fast for the preprocessor
- But generate lots of proofs, which take time to write
- These proofs then take lots of time to check

MaxSAT preprocessing techniques

- Generate proofs at much slower rate
- And so cause less overhead for proof checking

Difference between SAT and MaxSAT preprocessing?

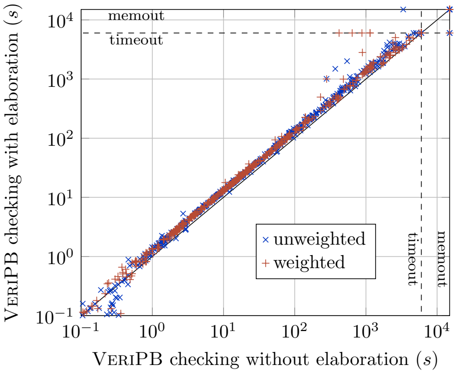
SAT preprocessing techniques

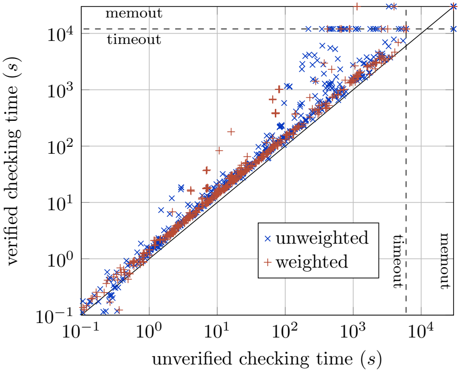
- Are fast for the preprocessor
- But generate lots of proofs, which take time to write
- These proofs then take lots of time to check

MaxSAT preprocessing techniques

- Generate proofs at much slower rate
- And so cause less overhead for proof checking

Taking a closer look at the proof checking overhead...





Formal verification is not the bottleneck

- Elaborating proof to formally verified format causes no major slowdown for VeriPB
- Nor is CakePB verified checking massively slower than VeriPB unverified checking

So why is proof checking for MaxSAT preprocessing slow?

Unverified proof checking with VeriPB seems like the main bottleneck

- Again, lots of engineering could (and should) be done here
- Focus has been on expanding the reach of proof logging to completely new domains

Checked deletion steps take **a lot** of time

- Perhaps not entirely unreasonable
- Equioptimality and equisatisfiability are strong guarantees that don't come for free

But the final variable renaming at the end can take half of the total time!?

- Using redundance & checked deletion for this is a very heavy hammer
- Maybe have dedicated rule stating the obvious that variable names don't matter?
- Related earlier observation for objective update: Stating not the new objective O^{NEW} but the **difference** $O^{\mathrm{NEW}} - O^{\mathrm{OLD}}$ speeds up proof logging dramatically

So why is proof checking for MaxSAT preprocessing slow?

Unverified proof checking with VeriPB seems like the main bottleneck

- Again, lots of engineering could (and should) be done here
- Focus has been on expanding the reach of proof logging to completely new domains

Checked deletion steps take **a lot** of time

- Perhaps not entirely unreasonable
- Equioptimality and equisatisfiability are strong guarantees that don't come for free

But the final variable renaming at the end can take half of the total time!?

- Using redundance & checked deletion for this is a very heavy hammer
- Maybe have dedicated rule stating the obvious that variable names don't matter?
- Related earlier observation for objective update: Stating not the new objective O^{NEW} but the **difference** $O^{\mathrm{NEW}} - O^{\mathrm{OLD}}$ speeds up proof logging dramatically

So why is proof checking for MaxSAT preprocessing slow?

Unverified proof checking with VeriPB seems like the main bottleneck

- Again, lots of engineering could (and should) be done here
- Focus has been on expanding the reach of proof logging to completely new domains

Checked deletion steps take **a lot** of time

- Perhaps not entirely unreasonable
- Equioptimality and equisatisfiability are strong guarantees that don't come for free

But the final variable renaming at the end can take half of the total time!?

- Using redundance & checked deletion for this is a very heavy hammer
- Maybe have dedicated rule stating the obvious that variable names don't matter?
- Related earlier observation for objective update: Stating not the new objective O^{NEW} but the **difference** $O^{\mathrm{NEW}} - O^{\mathrm{OLD}}$ speeds up proof logging dramatically

So why is proof checking for MaxSAT preprocessing slow?

Unverified proof checking with VeriPB seems like the main bottleneck

- Again, lots of engineering could (and should) be done here
- Focus has been on expanding the reach of proof logging to completely new domains

Checked deletion steps take **a lot** of time

- Perhaps not entirely unreasonable
- Equioptimality and equisatisfiability are strong guarantees that don't come for free

But the final variable renaming at the end can take half of the total time!?

- Using redundance & checked deletion for this is a very heavy hammer
- Maybe have dedicated rule stating the obvious that variable names don't matter?
- Related earlier observation for objective update: Stating not the new objective O^{NEW} but the **difference** $O^{\mathrm{NEW}} - O^{\mathrm{OLD}}$ speeds up proof logging dramatically

Conclusion

Our contribution:

- VeriPB proof logging for standalone MaxSAT preprocessor
  - 15+ preprocessing techniques implemented in MaxPre
- Proofs of equioptimality
  - First practical tool for even verifying equisatisfiability
- Formally verified end-to-end proof checking with CakePB

Future work:

- Implement more features
  - solution reconstruction
  - proof trimming
  - proof composition
- Optimize proof checker
- Provide efficient proof logging to even more combinatorial optimization paradigms!

Conclusion

Our contribution:

- VeriPB proof logging for standalone MaxSAT preprocessor
  - 15+ preprocessing techniques implemented in MaxPre
- Proofs of equioptimality
  - First practical tool for even verifying equisatisfiability
- Formally verified end-to-end proof checking with CakePB

Future work:

- Implement more features
  - solution reconstruction
  - proof trimming
  - proof composition
- Optimize proof checker
- Provide efficient proof logging to even more combinatorial optimization paradigms!

Conclusion

Our contribution:

- VeriPB proof logging for standalone MaxSAT preprocessor
  - 15+ preprocessing techniques implemented in MaxPre
- Proofs of equioptimality
  - First practical tool for even verifying equisatisfiability
- Formally verified end-to-end proof checking with CakePB

Future work:

- Implement more features
  - solution reconstruction
  - proof trimming
  - proof composition
- Optimize proof checker
- Provide efficient proof logging to even more combinatorial optimization paradigms!

Thank you for your attention!

Bibliography I

[BBN+23] Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, and Dieter Vandesande. Certified core-guided MaxSAT solving. In *Proceedings of the 29th International Conference on Automated Deduction (CADE-29)*, volume 14132 of *Lecture Notes in Computer Science*, pages 1–22. Springer, July 2023.

[BBN+24] Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, Tobias Paxian, and Dieter Vandesande. Certifying without loss of generality reasoning in solution-improving maximum satisfiability. In *Proceedings of the 30th International Conference on Principles and Practice of Constraint Programming (CP '24)*, September 2024. To appear.

[BGMN23] Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified dominance and symmetry breaking for combinatorial optimisation. *Journal of Artificial Intelligence Research*, 77:1539–1589, August 2023. Preliminary version in *AAAI '22*.

[BLM07] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8-9):606–618, 2007.

[CCT87] William Cook, Collette Rene Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987.

[DMM+24] Emir Demirović, Ciaran McCreesh, Matthew McIlree, Jakob Nordström, Andy Oertel, and Konstantin Sidorov. Pseudo-Boolean reasoning about states and transitions to certify dynamic programming and decision diagram algorithms. In *Proceedings of the 30th International Conference on Principles and Practice of Constraint Programming (CP '24)*, September 2024. To appear.

Bibliography II

[EGMN20] Jan Elffers, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Justifying all differences using pseudo-Boolean reasoning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*, pages 1486–1494, February 2020.

[FMSV20] Yuval Filmus, Meena Mahajan, Gaurav Sood, and Marc Vinyals. MaxSAT resolution and subcube sums. In *Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing (SAT '20)*, volume 12178 of *Lecture Notes in Computer Science*, pages 295–311. Springer, July 2020.

[GMM+20] Stephan Gocht, Ross McBride, Ciaran McCreesh, Jakob Nordström, Patrick Prosser, and James Trimble. Certifying solvers for clique and maximum common (connected) subgraph problems. In *Proceedings of the 26th International Conference on Principles and Practice of Constraint Programming (CP '20)*, volume 12333 of *Lecture Notes in Computer Science*, pages 338–357. Springer, September 2020.

[GMM+24] Stephan Gocht, Ciaran McCreesh, Magnus O. Myreen, Jakob Nordström, Andy Oertel, and Yong Kiam Tan. End-to-end verification for subgraph solving. In *Proceedings of the 368h AAAI Conference on Artificial Intelligence (AAAI '24)*, pages 8038–8047, February 2024.

[GMN20] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Subgraph isomorphism meets cutting planes: Solving with certified solutions. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*, pages 1134–1140, July 2020.

Bibliography III

[GMN22] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. An auditable constraint programming solver. In *Proceedings of the 28th International Conference on Principles and Practice of Constraint Programming (CP '22)*, volume 235 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:18, August 2022.

[GMNO22] Stephan Gocht, Ruben Martins, Jakob Nordström, and Andy Oertel. Certified CNF translations for pseudo-Boolean solving. In *Proceedings of the 25th International Conference on Theory and Applications of Satisfiability Testing (SAT '22)*, volume 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:25, August 2022.

[GN21] Stephan Gocht and Jakob Nordström. Certifying parity reasoning efficiently using pseudo-Boolean proofs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*, pages 3768–3777, February 2021.

[HOGN24] Alexander Hoen, Andy Oertel, Ambros Gleixner, and Jakob Nordström. Certifying MIP-based presolve reductions for 0–1 integer linear programs. In *Proceedings of the 21st International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR '24)*, volume 14742 of *Lecture Notes in Computer Science*, pages 310–328. Springer, May 2024.

[LNOR11] Javier Larrosa, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. A framework for certified Boolean branch-and-bound optimization. *Journal of Automated Reasoning*, 46(1):81–102, 2011.

Bibliography IV

[MIB+19] António Morgado, Alexey Ignatiev, María Luisa Bonet, João P. Marques-Silva, and Samuel R. Buss. DRMaxSAT with MaxHS: First contact. In *Proceedings of the 22nd International Conference on Theory and Applications of Satisfiability Testing (SAT '19)*, volume 11628 of *Lecture Notes in Computer Science*, pages 239–249. Springer, July 2019.

[MM11] António Morgado and João Marques-Silva. On validating Boolean optimizers. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence, (ICTAI '11)*, pages 924–926, 2011.

[MM23] Matthew McIlree and Ciaran McCreesh. Proof logging for smart extensional constraints. In *Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming (CP '23)*, volume 280 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:17, August 2023.

[MMN24] Matthew McIlree, Ciaran McCreesh, and Jakob Nordström. Proof logging for the circuit constraint. In *Proceedings of the 21st International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR '24)*, volume 14743 of *Lecture Notes in Computer Science*, pages 38–55. Springer, May 2024.

[PCH20] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Towards bridging the gap between SAT and Max-SAT refutations. In *Proceedings of the 32nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI '20)*, pages 137–144, November 2020.

Bibliography V

[PCH21] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. A proof builder for Max-SAT. In *Proceedings of the 24th International Conference on Theory and Applications of Satisfiability Testing (SAT '21)*, volume 12831 of *Lecture Notes in Computer Science*, pages 488–498. Springer, July 2021.

[VDB22] Dieter Vandesande, Wolf De Wulf, and Bart Bogaerts. QMaxSATpb: A certified MaxSAT solver. In *Proceedings of the 16th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR '22)*, volume 13416 of *Lecture Notes in Computer Science*, pages 429–442. Springer, September 2022.