Certified Symmetry and Dominance Breaking for Combinatorial Optimisation

Jakob Nordström

University of Copenhagen
and Lund University

Swedish Operations Research Conference
Stockholm, Sweden
October 24, 2022

Joint work with Bart Bogaerts, Stephan Gocht, and Ciaran McCreesh

Combinatorial Solving and Optimisation

- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]*
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]

*See end of slides for all references with bibliographic details

Combinatorial Solving and Optimisation

- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]*
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
- Solve NP problems (or worse) very successfully in practice!

*See end of slides for all references with bibliographic details

Combinatorial Solving and Optimisation

- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]*
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
- Solve NP problems (or worse) very successfully in practice!
- Except solvers are sometimes wrong... (Even best commercial ones) [BLB10, CKSW13, AGJ+18, GSD19, GS19]

*See end of slides for all references with bibliographic details

Combinatorial Solving and Optimisation

- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]*
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
- Solve NP problems (or worse) very successfully in practice!
- Except solvers are sometimes wrong... (Even best commercial ones) [BLB10, CKSW13, AGJ+18, GSD19, GS19]
- Software testing doesn't suffice to resolve this problem

*See end of slides for all references with bibliographic details

Combinatorial Solving and Optimisation

- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]*
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
- Solve NP problems (or worse) very successfully in practice!
- Except solvers are sometimes wrong. . . (Even best commercial ones) [BLB10, CKSW13, AGJ$^+$18, GSD19, GS19]
- Software testing doesn't suffice to resolve this problem
- Formal verification techniques cannot deal with level of complexity of modern solvers

*See end of slides for all references with bibliographic details

Certified Results with Proof Logging

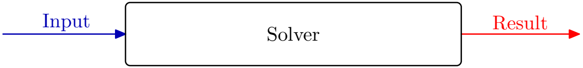
Solution: Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that result is correct

Certified Results with Proof Logging

Solution: Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that result is correct



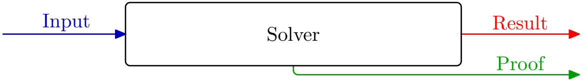
Workflow:

1. Run solver on problem input

Certified Results with Proof Logging

Solution: Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that result is correct



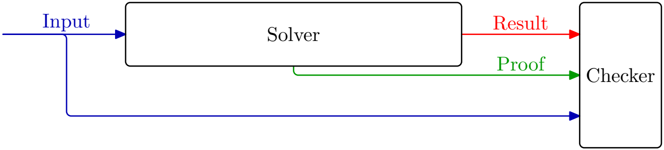
Workflow:

1. Run solver on problem input
2. Get as output not only result but also proof

Certified Results with Proof Logging

Solution: Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that result is correct



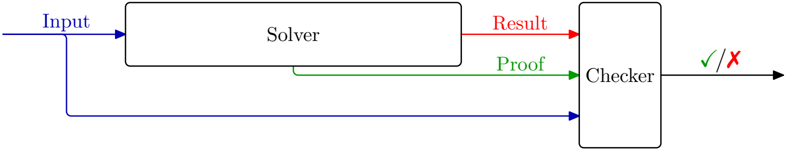
Workflow:

1. Run solver on problem input
2. Get as output not only result but also proof
3. Feed input + result + proof to proof checker

Certified Results with Proof Logging

Solution: Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that result is correct



Workflow:

1. Run solver on problem input
2. Get as output not only result but also proof
3. Feed input + result + proof to proof checker
4. Verify that proof checker says result is correct

Yet Another SAT Success Story

Many proof logging formats for SAT solving using CNF clausal format:

- DRAT [HHW13a, HHW13b, WHH14]
- GRIT [CMS17]
- LRAT [CHH+17]
- ...

Well established — required in main track of SAT competitions
Crucial for unsatisfiable formulas

Yet Another SAT Success Story

Many proof logging formats for SAT solving using CNF clausal format:

- DRAT [HHW13a, HHW13b, WHH14]
- GRIT [CMS17]
- LRAT [CHH+17]
- ...

Well established — required in main track of SAT competitions
Crucial for unsatisfiable formulas

But efficient proof logging has remained out of reach for stronger paradigms

Yet Another SAT Success Story(?)

Many proof logging formats for SAT solving using CNF clausal format:

- DRAT [HHW13a, HHW13b, WHH14]
- GRIT [CMS17]
- LRAT [CHH+17]
- ...

Well established — required in main track of SAT competitions
Crucial for unsatisfiable formulas

But efficient proof logging has remained out of reach for stronger paradigms

And, in fact, even for some advanced SAT solving techniques:

- cardinality reasoning
- Gaussian elimination
- symmetry handling

Our Work: Efficient Proof Logging for Symmetry Breaking

Paper *Certified Symmetry and Dominance Breaking for Combinatorial Optimisation* at *AAAI '22* [BGMN22]:

Implementation in proof checker VeriPB [Ver]

Our Work: Efficient Proof Logging for Symmetry Breaking

Paper *Certified Symmetry and Dominance Breaking for Combinatorial Optimisation* at *AAAI '22* [BGMN22]:

Implementation in proof checker VeriPB [Ver]

- First general & efficient proof logging method for symmetry breaking

Our Work: Efficient Proof Logging for Symmetry Breaking

Paper *Certified Symmetry and Dominance Breaking for Combinatorial Optimisation* at *AAAI '22* [BGMN22]:

Implementation in proof checker VeriPB [Ver]

- First general & efficient proof logging method for symmetry breaking
- Supports also pseudo-Boolean reasoning and Gaussian elimination

Our Work: Efficient Proof Logging for Symmetry Breaking

Paper *Certified Symmetry and Dominance Breaking for Combinatorial Optimisation* at *AAAI '22* [BGMN22]:

Implementation in proof checker VeriPB [Ver]

- First general & efficient proof logging method for symmetry breaking
- Supports also pseudo-Boolean reasoning and Gaussian elimination
- Based on 0-1 integer linear constraints instead of clauses

Our Work: Efficient Proof Logging for Symmetry Breaking

Paper *Certified Symmetry and Dominance Breaking for Combinatorial Optimisation* at *AAAI '22* [BGMN22]:

Implementation in proof checker VeriPB [Ver]

- First general & efficient proof logging method for symmetry breaking
- Supports also pseudo-Boolean reasoning and Gaussian elimination
- Based on 0-1 integer linear constraints instead of clauses
- Uses cutting planes method [CCT87] with additional rules

Outline of Presentation

What I hope to cover in the rest of this presentation:

- Basics of proof logging with 0-1 linear constraints
- New rule for symmetry and dominance breaking
- Application to symmetry breaking for SAT solving (also other applications, but focus here on SAT)
- Some future research directions

Outline of Presentation

What I hope to cover in the rest of this presentation:

- Basics of proof logging with 0-1 linear constraints
- New rule for symmetry and dominance breaking
- Application to symmetry breaking for SAT solving (also other applications, but focus here on SAT)
- Some future research directions

Caveat: Only exact problems in this talk but:

- This is already very challenging
- Ideas seem likely to generalize

0-1 Integer Linear (a.k.a. Pseudo-Boolean) Constraints

Pseudo-Boolean (PB) constraints are 0-1 integer linear constraints

$$C \doteq \sum_i a_i \ell_i \geq A$$

- $a_i, A \in \mathbb{Z}$
- literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)
- variables x_i take values $0 = \mathit{false}$ or $1 = \mathit{true}$

0-1 Integer Linear (a.k.a. Pseudo-Boolean) Constraints

Pseudo-Boolean (PB) constraints are 0-1 integer linear constraints

$$C \doteq \sum_i a_i \ell_i \geq A$$

- $a_i, A \in \mathbb{Z}$
- literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)
- variables x_i take values $0 = \mathit{false}$ or $1 = \mathit{true}$

Negation of constraint

$$\neg C \doteq \sum_i a_i \ell_i \leq A - 1$$

0-1 Integer Linear (a.k.a. Pseudo-Boolean) Constraints

Pseudo-Boolean (PB) constraints are 0-1 integer linear constraints

$$C \doteq \sum_i a_i \ell_i \geq A$$

- $a_i, A \in \mathbb{Z}$
- literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)
- variables x_i take values $0 = \mathit{false}$ or $1 = \mathit{true}$

Negation of constraint

$$\neg C \doteq \sum_i a_i \ell_i \leq A - 1$$

Pseudo-Boolean formulas $F \doteq \bigwedge_{i=1}^{m} C_i$ are conjunctions of pseudo-Boolean constraints (a.k.a. 0-1 integer linear programs)

Some Types of Pseudo-Boolean Constraints

1. Clauses

$$x \vee \overline{y} \vee z \quad \Leftrightarrow \quad x + \overline{y} + z \geq 1$$

Some Types of Pseudo-Boolean Constraints

1. Clauses

$$x \vee \overline{y} \vee z \quad \Leftrightarrow \quad x + \overline{y} + z \geq 1$$

2. Cardinality constraints

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

Some Types of Pseudo-Boolean Constraints

1. Clauses

$$x \lor \overline{y} \lor z \quad \Leftrightarrow \quad x + \overline{y} + z \geq 1$$

2. Cardinality constraints

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

3. General pseudo-Boolean constraints

$$x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i\ell_i \geq A \qquad \sum_i b_i\ell_i \geq B}{\sum_i(c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i\ell_i \geq A}{\sum_i a_i\ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i \ell_i \geq A}{\sum_i a_i \ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

Lin comb $$\frac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i \ell_i \geq A}{\sum_i a_i \ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

Lin comb $$\frac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{(2x + 4y + 2z + w) + 2 \cdot (2x + y + w) \geq 5 + 2 \cdot 2}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i) \ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i \ell_i \geq A}{\sum_i a_i \ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

Lin comb $$\frac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i \ell_i \geq A}{\sum_i a_i \ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

$$\dfrac{\dfrac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9}\text{Lin comb} \qquad \overline{z} \geq 0}{}\text{Lin comb}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i) \ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i \ell_i \geq A}{\sum_i a_i \ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

$$\dfrac{\text{Lin comb}\ \dfrac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \qquad \overline{z} \geq 0}{6x + 6y + 2z + 3w + 2 \cdot \overline{z} \geq 9 + 2 \cdot 0}\ \text{Lin comb}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i) \ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i \ell_i \geq A}{\sum_i a_i \ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

$$\dfrac{\text{Lin comb}\ \dfrac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \qquad \overline{z} \geq 0}{6x + 6y + 3w + 2 \geq 9}\ \text{Lin comb}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i\ell_i \geq A \qquad \sum_i b_i\ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i\ell_i \geq A}{\sum_i a_i\ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

$$\dfrac{\dfrac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9}\,\text{Lin comb} \qquad \overline{z} \geq 0}{6x + 6y + 3w \geq 7}\,\text{Lin comb}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i \ell_i \geq A}{\sum_i a_i \ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

$$\dfrac{\text{Lin comb}\ \dfrac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \qquad \overline{z} \geq 0}{\text{Div}\ \dfrac{6x + 6y + 3w \geq 7}{2x + 2y + w \geq 2\frac{1}{3}}}\ \text{Lin comb}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms
$$\frac{}{\ell_i \geq 0}$$

Linear combination
$$\frac{\sum_i a_i\ell_i \geq A \qquad \sum_i b_i\ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division
$$\frac{\sum_i c a_i\ell_i \geq A}{\sum_i a_i\ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

$$\text{Div}\ \frac{\text{Lin comb}\ \dfrac{\text{Lin comb}\ \dfrac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \qquad \overline{z} \geq 0}{6x + 6y + 3w \geq 7}}{2x + 2y + w \geq 3}$$

Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i) \ell_i \geq c_A A + c_B B} \quad [c_A, c_B \in \mathbb{N}]$$

Division $$\frac{\sum_i c a_i \ell_i \geq A}{\sum_i a_i \ell_i \geq \lceil A/c \rceil} \quad [c \in \mathbb{N}^+]$$

Toy example:

$$\dfrac{\text{Lin comb}\ \dfrac{\text{Lin comb}\ \dfrac{2x + 4y + 2z + w \geq 5 \qquad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \qquad \overline{z} \geq 0}{6x + 6y + 3w \geq 7}}{2x + 2y + w \geq 3}\ \text{Div}$$

(See [BN21] for more details about cutting planes)

Proof Logging for SAT with Pseudo-Boolean Reasoning

- View clauses as pseudo-Boolean constraints

Proof Logging for SAT with Pseudo-Boolean Reasoning

- View clauses as pseudo-Boolean constraints
- Operate on constraints with cutting planes rules

Proof Logging for SAT with Pseudo-Boolean Reasoning

- View clauses as pseudo-Boolean constraints
- Operate on constraints with cutting planes rules
- Prove unsatisfiability by deriving $0 \geq 1$

Proof Logging for SAT with Pseudo-Boolean Reasoning

- View clauses as pseudo-Boolean constraints
- Operate on constraints with cutting planes rules
- Prove unsatisfiability by deriving $0 \geq 1$
- **Fact:** Fully sufficient for proof logging for so-called conflict-driven clause learning [BS97, MS99, MMZ$^+$01]

Proof Logging for SAT with Pseudo-Boolean Reasoning

- View clauses as pseudo-Boolean constraints
- Operate on constraints with cutting planes rules
- Prove unsatisfiability by deriving $0 \geq 1$
- **Fact:** Fully sufficient for proof logging for so-called conflict-driven clause learning [BS97, MS99, MMZ$^+$01]
- Also need **extension** rule (analogue of RAT [JHB12] used in SAT proof logging) to deal with, e.g., preprocessing/presolving

Extension Rule: Redundance-Based Strengthening

C is redundant with respect to F if F and $F \wedge C$ are equisatisfiable

Want to allow adding redundant constraints

Extension Rule: Redundance-Based Strengthening

C is redundant with respect to F if F and $F \land C$ are equisatisfiable
Want to allow adding redundant constraints

Redundance-based strengthening [BT19, GN21]

C is redundant with respect to F if and only if there is a substitution ω (mapping variables to truth values or literals), called a witness, for which

$$F \land \neg C \models (F \land C)\restriction_{\omega}$$

Extension Rule: Redundance-Based Strengthening

C is redundant with respect to F if F and $F \wedge C$ are equisatisfiable
Want to allow adding redundant constraints

Redundance-based strengthening [BT19, GN21]

C is redundant with respect to F if and only if there is a substitution ω (mapping variables to truth values or literals), called a witness, for which

$$F \wedge \neg C \models (F \wedge C)\restriction_{\omega}$$

- Proof sketch for interesting direction: If α satisfies F but falsifies C, then $\alpha \circ \omega$ satisfies $F \wedge C$

Extension Rule: Redundance-Based Strengthening

C is redundant with respect to F if F and $F \wedge C$ are equisatisfiable
Want to allow adding redundant constraints

Redundance-based strengthening [BT19, GN21]

C is redundant with respect to F if and only if there is a substitution ω (mapping variables to truth values or literals), called a witness, for which

$$F \wedge \neg C \models (F \wedge C)\restriction_{\omega}$$

- Proof sketch for interesting direction: If α satisfies F but falsifies C, then $\alpha \circ \omega$ satisfies $F \wedge C$
- Witness ω should be specified and implication efficiently verifiable by very simple checks (technical details omitted)

The Power of Proof Logging with Extended Cutting Planes

0-1 linear inequalities convenient to capture SAT reasoning (with clauses)

The Power of Proof Logging with Extended Cutting Planes

0-1 linear inequalities convenient to capture SAT reasoning (with clauses)

And yields efficient proof logging for wider range of problems/algorithms:

- Pre- and inprocessing [GN21] (since redundance rule subsumes RAT)
- pseudo-Boolean reasoning (by design)
- Gaussian elimination [GN21]

The Power of Proof Logging with Extended Cutting Planes

0-1 linear inequalities convenient to capture SAT reasoning (with clauses)

And yields efficient proof logging for wider range of problems/algorithms:

- Pre- and inprocessing [GN21] (since redundance rule subsumes RAT)
- pseudo-Boolean reasoning (by design)
- Gaussian elimination [GN21]
- subgraph problems [GMN20, GMM$^+$20]

The Power of Proof Logging with Extended Cutting Planes

0-1 linear inequalities convenient to capture SAT reasoning (with clauses)

And yields efficient proof logging for wider range of problems/algorithms:

- Pre- and inprocessing [GN21] (since redundance rule subsumes RAT)
- pseudo-Boolean reasoning (by design)
- Gaussian elimination [GN21]
- subgraph problems [GMN20, GMM$^+$20]
- solving pseudo-Boolean formulas via translation to CNF [GMNO22]

The Power of Proof Logging with Extended Cutting Planes

0-1 linear inequalities convenient to capture SAT reasoning (with clauses)

And yields efficient proof logging for wider range of problems/algorithms:

- Pre- and inprocessing [GN21] (since redundance rule subsumes RAT)
- pseudo-Boolean reasoning (by design)
- Gaussian elimination [GN21]
- subgraph problems [GMN20, GMM^{+}20]
- solving pseudo-Boolean formulas via translation to CNF [GMNO22]
- (basic) constraint programming [EGMN20, GMN22]

The Power of Proof Logging with Extended Cutting Planes

0-1 linear inequalities convenient to capture SAT reasoning (with clauses)

And yields efficient proof logging for wider range of problems/algorithms:

- Pre- and inprocessing [GN21] (since redundance rule subsumes RAT)
- pseudo-Boolean reasoning (by design)
- Gaussian elimination [GN21]
- subgraph problems [GMN20, GMM$^+$20]
- solving pseudo-Boolean formulas via translation to CNF [GMNO22]
- (basic) constraint programming [EGMN20, GMN22]
- **This talk:** extend to symmetry and dominance breaking [BGMN22]

The Power of Proof Logging with Extended Cutting Planes

0-1 linear inequalities convenient to capture SAT reasoning (with clauses)

And yields efficient proof logging for wider range of problems/algorithms:

- Pre- and inprocessing [GN21] (since redundance rule subsumes RAT)
- pseudo-Boolean reasoning (by design)
- Gaussian elimination [GN21]
- subgraph problems [GMN20, GMM$^+$20]
- solving pseudo-Boolean formulas via translation to CNF [GMNO22]
- (basic) constraint programming [EGMN20, GMN22]
- **This talk:** extend to symmetry and dominance breaking [BGMN22]

Zoom tutorial on all of these developments Mon Nov 28 at 14:00 CET
Combinatorial Solving with Provably Correct Results
See `http://www.jakobnordstrom.se/miao-seminars`

The Challenge of Symmetries

(Syntactic) symmetry: substitution σ preserving F ($F{\restriction}_\sigma \doteq F$)

- Show up in some hard SAT benchmarks
- Can play crucial role in CP and MIP problems [AW13, GSVW14]

The Challenge of Symmetries

(Syntactic) symmetry: substitution σ preserving F ($F{\restriction}_\sigma \doteq F$)

- Show up in some hard SAT benchmarks
- Can play crucial role in CP and MIP problems [AW13, GSVW14]

Symmetry breaking in SAT
Add constraints filtering out symmetric solutions [ASM06, DBBD16]

Symmetric learning in SAT
Allow to add all symmetric versions of learned constraint [DBB17]

The Challenge of Symmetries

(Syntactic) symmetry: substitution σ preserving F ($F\restriction_\sigma \doteq F$)

- Show up in some hard SAT benchmarks
- Can play crucial role in CP and MIP problems [AW13, GSVW14]

Symmetry breaking in SAT
Add constraints filtering out symmetric solutions [ASM06, DBBD16]

Symmetric learning in SAT
Allow to add all symmetric versions of learned constraint [DBB17]

Not supported by standard SAT proof logging!

Optimisation Problems

Deal with symmetry breaking by switching focus to optimisation
(which the title of the talk kind of promised anyway)

Optimisation Problems

Deal with symmetry breaking by switching focus to optimisation
(which the title of the talk kind of promised anyway)

Pseudo-Boolean optimisation

Minimize $f = \sum_i w_i \ell_i$ (for $w_i \in \mathbb{N}^+$) subject to constraints in F

Optimisation Problems

Deal with symmetry breaking by switching focus to optimisation
(which the title of the talk kind of promised anyway)

Pseudo-Boolean optimisation

Minimize $f = \sum_i w_i \ell_i$ (for $w_i \in \mathbb{N}^+$) subject to constraints in F

Proof of optimality:

- F satisfied by α
- $F \wedge \left(\sum_i w_i \ell_i < \sum_i w_i \cdot \alpha(\ell_i)\right)$ is infeasible

Optimisation Problems

Deal with symmetry breaking by switching focus to optimisation (which the title of the talk kind of promised anyway)

Pseudo-Boolean optimisation

Minimize $f = \sum_i w_i \ell_i$ (for $w_i \in \mathbb{N}^+$) subject to constraints in F

Proof of optimality:

- F satisfied by α
- $F \wedge \left(\sum_i w_i \ell_i < \sum_i w_i \cdot \alpha(\ell_i)\right)$ is infeasible

[Note that $\sum_i w_i \ell_i < \sum_i w_i \cdot \alpha(\ell_i)$ means $\sum_i w_i \ell_i \leq -1 + \sum_i w_i \cdot \alpha(\ell_i)$]

Optimisation Problems

Deal with symmetry breaking by switching focus to optimisation (which the title of the talk kind of promised anyway)

Pseudo-Boolean optimisation

Minimize $f = \sum_i w_i \ell_i$ (for $w_i \in \mathbb{N}^+$) subject to constraints in F

Proof of optimality:

- F satisfied by α
- $F \wedge \left(\sum_i w_i \ell_i < \sum_i w_i \cdot \alpha(\ell_i)\right)$ is infeasible

[Note that $\sum_i w_i \ell_i < \sum_i w_i \cdot \alpha(\ell_i)$ means $\sum_i w_i \ell_i \leq -1 + \sum_i w_i \cdot \alpha(\ell_i)$]

Spoiler alert:

For decision problem, nothing stops us from inventing objective function (like $\sum_{i=1}^{n} 2^{n-i} \cdot x_i$ minimized by lexicographic order)

Proof Logging for Optimisation Problems

How does proof system change?

Rules must preserve (at least one) optimal solution

Proof Logging for Optimisation Problems

How does proof system change?
Rules must preserve (at least one) optimal solution

1. Standard cutting planes rules OK — derive constraints that must hold for any satisfying assignment

Proof Logging for Optimisation Problems

How does proof system change?
Rules must preserve (at least one) optimal solution

1. Standard cutting planes rules OK — derive constraints that must hold for any satisfying assignment

2. Once solution α has been found, allow constraint $\sum_i w_i \ell_i < \sum_i w_i \cdot \alpha(\ell_i)$ to force search for better solutions

Proof Logging for Optimisation Problems

How does proof system change?
Rules must preserve (at least one) optimal solution

1. Standard cutting planes rules OK — derive constraints that must hold for any satisfying assignment
2. Once solution α has been found, allow constraint $\sum_i w_i \ell_i < \sum_i w_i \cdot \alpha(\ell_i)$ to force search for better solutions
3. Redundance rule must not destroy good solutions

Proof Logging for Optimisation Problems

How does proof system change?
Rules must preserve (at least one) optimal solution

1. Standard cutting planes rules OK — derive constraints that must hold for any satisfying assignment

2. Once solution α has been found, allow constraint $\sum_i w_i \ell_i < \sum_i w_i \cdot \alpha(\ell_i)$ to force search for better solutions

3. Redundance rule must not destroy good solutions

Redundance-based strengthening, optimisation version

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models (F \wedge C)\restriction_\omega \ \wedge \ f\restriction_\omega \leq f$$

Redundance and Dominance Rules

Redundance-based strengthening, optimisation version

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models (F \wedge C)\restriction_\omega \;\wedge\; f\restriction_\omega \leq f$$

Redundance and Dominance Rules

Redundance-based strengthening, optimisation version

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models (F \wedge C)\restriction_\omega \wedge f\restriction_\omega \leq f$$

Can be more aggressive if witness ω strictly improves solution

Redundance and Dominance Rules

Redundance-based strengthening, optimisation version

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models (F \wedge C)\restriction_\omega \ \wedge \ f\restriction_\omega \leq f$$

Can be more aggressive if witness ω strictly improves solution

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F\restriction_\omega \ \wedge \ f\restriction_\omega < f$$

Redundance and Dominance Rules

Redundance-based strengthening, optimisation version

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models (F \wedge C)\restriction_\omega \ \wedge \ f\restriction_\omega \leq f$$

Can be more aggressive if witness ω strictly improves solution

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F\restriction_\omega \ \wedge \ f\restriction_\omega < f$$

- Applying ω should strictly decrease f
- If so, don't need to show that $C\restriction_\omega$ implied!

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F\restriction_\omega \wedge f\restriction_\omega < f$$

Why is this sound?

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F\!\restriction_\omega \;\wedge\; f\!\restriction_\omega < f$$

Why is this sound?

1. Suppose α satisfies F but falsifies C (i.e., satisfies $\neg C$)

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F\restriction_{\omega} \wedge f\restriction_{\omega} < f$$

Why is this sound?

1. Suppose α satisfies F but falsifies C (i.e., satisfies $\neg C$)
2. Then $\alpha \circ \omega$ satisfies F and $f(\alpha \circ \omega) < f(\alpha)$

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \land \neg C \models F\restriction_\omega \ \land \ f\restriction_\omega < f$$

Why is this sound?

1. Suppose α satisfies F but falsifies C (i.e., satisfies $\neg C$)
2. Then $\alpha \circ \omega$ satisfies F and $f(\alpha \circ \omega) < f(\alpha)$
3. If $\alpha \circ \omega$ satisfies C, we're done

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F\restriction_\omega \wedge f\restriction_\omega < f$$

Why is this sound?

1. Suppose α satisfies F but falsifies C (i.e., satisfies $\neg C$)
2. Then $\alpha \circ \omega$ satisfies F and $f(\alpha \circ \omega) < f(\alpha)$
3. If $\alpha \circ \omega$ satisfies C, we're done
4. Otherwise $(\alpha \circ \omega) \circ \omega$ satisfies F and $f((\alpha \circ \omega) \circ \omega) < f(\alpha \circ \omega)$

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F\restriction_\omega \;\wedge\; f\restriction_\omega < f$$

Why is this sound?

1. Suppose α satisfies F but falsifies C (i.e., satisfies $\neg C$)
2. Then $\alpha \circ \omega$ satisfies F and $f(\alpha \circ \omega) < f(\alpha)$
3. If $\alpha \circ \omega$ satisfies C, we're done
4. Otherwise $(\alpha \circ \omega) \circ \omega$ satisfies F and $f((\alpha \circ \omega) \circ \omega) < f(\alpha \circ \omega)$
5. If $(\alpha \circ \omega) \circ \omega$ satisfies C, we're done

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F\restriction_\omega \;\wedge\; f\restriction_\omega < f$$

Why is this sound?

1. Suppose α satisfies F but falsifies C (i.e., satisfies $\neg C$)
2. Then $\alpha \circ \omega$ satisfies F and $f(\alpha \circ \omega) < f(\alpha)$
3. If $\alpha \circ \omega$ satisfies C, we're done
4. Otherwise $(\alpha \circ \omega) \circ \omega$ satisfies F and $f((\alpha \circ \omega) \circ \omega) < f(\alpha \circ \omega)$
5. If $(\alpha \circ \omega) \circ \omega$ satisfies C, we're done
6. Otherwise $((\alpha \circ \omega) \circ \omega) \circ \omega$ satisfies F and $f(((\alpha \circ \omega) \circ \omega) \circ \omega) < f((\alpha \circ \omega) \circ \omega)$

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \land \neg C \models F\restriction_\omega \land f\restriction_\omega < f$$

Why is this sound?

1. Suppose α satisfies F but falsifies C (i.e., satisfies $\neg C$)
2. Then $\alpha \circ \omega$ satisfies F and $f(\alpha \circ \omega) < f(\alpha)$
3. If $\alpha \circ \omega$ satisfies C, we're done
4. Otherwise $(\alpha \circ \omega) \circ \omega$ satisfies F and $f((\alpha \circ \omega) \circ \omega) < f(\alpha \circ \omega)$
5. If $(\alpha \circ \omega) \circ \omega$ satisfies C, we're done
6. Otherwise $((\alpha \circ \omega) \circ \omega) \circ \omega$ satisfies F and $f(((\alpha \circ \omega) \circ \omega) \circ \omega) < f((\alpha \circ \omega) \circ \omega)$
7. ...

Soundness of Dominance Rule

Dominance-based strengthening (simplified)

Add constraint C to formula F if exists witness substitution ω such that

$$F \wedge \neg C \models F{\restriction_\omega} \ \wedge \ f{\restriction_\omega} < f$$

Why is this sound?

1. Suppose α satisfies F but falsifies C (i.e., satisfies $\neg C$)
2. Then $\alpha \circ \omega$ satisfies F and $f(\alpha \circ \omega) < f(\alpha)$
3. If $\alpha \circ \omega$ satisfies C, we're done
4. Otherwise $(\alpha \circ \omega) \circ \omega$ satisfies F and $f((\alpha \circ \omega) \circ \omega) < f(\alpha \circ \omega)$
5. If $(\alpha \circ \omega) \circ \omega$ satisfies C, we're done
6. Otherwise $((\alpha \circ \omega) \circ \omega) \circ \omega$ satisfies F and $f(((\alpha \circ \omega) \circ \omega) \circ \omega) < f((\alpha \circ \omega) \circ \omega)$
7. ...
8. Can't go on forever, so finally reach α' satisfying $F \wedge C$

Strategy for SAT Symmetry Breaking

1. Pretend to solve optimisation problem minimizing $f \doteq \sum_{i=1}^{n} 2^{n-i} \cdot x_i$ (searching lexicographically smallest assignment satisfying formula)

Strategy for SAT Symmetry Breaking

1. Pretend to solve optimisation problem minimizing $f \doteq \sum_{i=1}^{n} 2^{n-i} \cdot x_i$ (searching lexicographically smallest assignment satisfying formula)
2. Derive pseudo-Boolean lex-leader constraint

$$\begin{aligned} C_\sigma &\doteq f \leq f\restriction_\sigma \\ &\doteq \sum_{i=1}^{n} 2^{n-i} \cdot (\sigma(x_i) - x_i) \geq 0 \end{aligned}$$

Strategy for SAT Symmetry Breaking

1. Pretend to solve optimisation problem minimizing $f \doteq \sum_{i=1}^{n} 2^{n-i} \cdot x_i$ (searching lexicographically smallest assignment satisfying formula)
2. Derive pseudo-Boolean lex-leader constraint

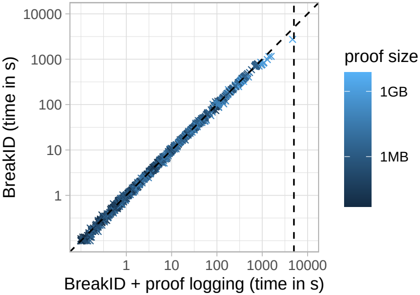
$$\begin{aligned} C_\sigma &\doteq f \leq f\restriction_\sigma \\ &\doteq \sum_{i=1}^{n} 2^{n-i} \cdot (\sigma(x_i) - x_i) \geq 0 \end{aligned}$$

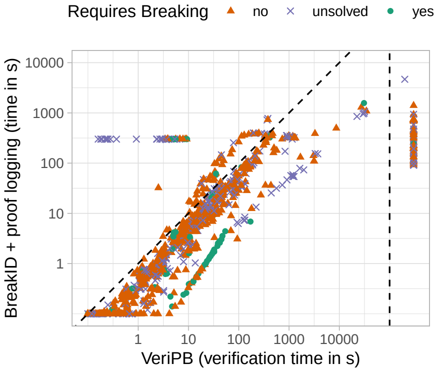
3. Derive CNF encoding of lex-leader constraints from PB constraint (in same spirit as PB-to-CNF translation in [GMNO22])

$$\begin{array}{ll} y_0 & \overline{y}_j \vee \overline{\sigma(x_j)} \vee x_j \\ \overline{y}_{j-1} \vee \overline{x}_j \vee \sigma(x_j) & y_j \vee \overline{y}_{j-1} \vee \overline{x}_j \\ \overline{y}_j \vee y_{j-1} & y_j \vee \overline{y}_{j-1} \vee \sigma(x_j) \end{array}$$

Experimental Evaluation

- Evaluated on SAT competition benchmarks
- BREAKID [DBBD16, Bre] used to find and break symmetries





- Proof logging overhead negligible
- Verification at most 20 times slower than solving for 95% of instances

Future Research Directions

Performance and reliability of pseudo-Boolean proof logging

- Trim proof while verifying (as in DRAT-TRIM [HHW13a])
- Compress proof file using binary format
- Design formally verified proof checker *(work in progress)*

Future Research Directions

Performance and reliability of pseudo-Boolean proof logging

- Trim proof while verifying (as in DRAT-trim [HHW13a])
- Compress proof file using binary format
- Design formally verified proof checker *(work in progress)*

Proof logging for combinatorial optimization

- Symmetric learning and recycling (substitution) of subproofs
- Maximum satisfiability (MaxSAT) solving *(work in progress [VDB22])*
- Pseudo-Boolean optimization
- Mixed integer linear programming *(work on SCIP in [CGS17, EG21])*

Future Research Directions

Performance and reliability of pseudo-Boolean proof logging

- Trim proof while verifying (as in DRAT-TRIM [HHW13a])
- Compress proof file using binary format
- Design formally verified proof checker *(work in progress)*

Proof logging for combinatorial optimization

- Symmetric learning and recycling (substitution) of subproofs
- Maximum satisfiability (MaxSAT) solving *(work in progress [VDB22])*
- Pseudo-Boolean optimization
- Mixed integer linear programming *(work on SCIP in [CGS17, EG21])*

And more. . .

- Lots of challenging problems and interesting ideas

Future Research Directions

Performance and reliability of pseudo-Boolean proof logging

- Trim proof while verifying (as in DRAT-TRIM [HHW13a])
- Compress proof file using binary format
- Design formally verified proof checker *(work in progress)*

Proof logging for combinatorial optimization

- Symmetric learning and recycling (substitution) of subproofs
- Maximum satisfiability (MaxSAT) solving *(work in progress [VDB22])*
- Pseudo-Boolean optimization
- Mixed integer linear programming *(work on SCIP in [CGS17, EG21])*

And more...

- Lots of challenging problems and interesting ideas
- We're hiring! Talk to me to join the proof logging revolution! ☺

Summing up

- Combinatorial solving and optimization is a true success story
- But ensuring correctness is a crucial, and not yet satisfactorily addressed, concern
- Certifying solvers producing machine-verifiable proofs of correctness seems like most promising approach
- Cutting planes reasoning with pseudo-Boolean constraints seems to hit a sweet spot between simplicity and expressivity
- **This work:** Efficient proof logging for symmetry and dominance breaking using cutting planes proof system with extensions

Summing up

- Combinatorial solving and optimization is a true success story
- But ensuring correctness is a crucial, and not yet satisfactorily addressed, concern
- Certifying solvers producing machine-verifiable proofs of correctness seems like most promising approach
- Cutting planes reasoning with pseudo-Boolean constraints seems to hit a sweet spot between simplicity and expressivity
- **This work:** Efficient proof logging for symmetry and dominance breaking using cutting planes proof system with extensions

Thank you for your attention!

References I

[ABM+11] Eyad Alkassar, Sascha Böhme, Kurt Mehlhorn, Christine Rizkallah, and Pascal Schweitzer. An introduction to certifying algorithms. *it - Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 53(6):287–293, December 2011.

[AGJ+18] Özgür Akgün, Ian P. Gent, Christopher Jefferson, Ian Miguel, and Peter Nightingale. Metamorphic testing of constraint solvers. In *Proceedings of the 24th International Conference on Principles and Practice of Constraint Programming (CP '18)*, volume 11008 of *Lecture Notes in Computer Science*, pages 727–736. Springer, August 2018.

[ASM06] Fadi A. Aloul, Karem A. Sakallah, and Igor L. Markov. Efficient symmetry breaking for Boolean satisfiability. *IEEE Transactions on Computers*, 55(5):549–558, May 2006. Preliminary version in *IJCAI '03*.

[AW13] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*, pages 449–481. Springer, 2013.

References II

[BGMN22] Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified symmetry and dominance breaking for combinatorial optimisation. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI '22)*, pages 3698–3707, February 2022.

[BHvMW21] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2nd edition, February 2021.

[BLB10] Robert Brummayer, Florian Lonsing, and Armin Biere. Automated testing and debugging of SAT and QBF solvers. In *Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT '10)*, volume 6175 of *Lecture Notes in Computer Science*, pages 44–57. Springer, July 2010.

[BN21] Samuel R. Buss and Jakob Nordström. Proof complexity and SAT solving. In Biere et al. [BHvMW21], chapter 7, pages 233–350.

[BR07] Robert Bixby and Edward Rothberg. Progress in computational mixed integer programming—A look back from the other side of the tipping point. *Annals of Operations Research*, 149(1):37–41, February 2007.

[Bre] Breakid. `https://bitbucket.org/krr/breakid`.

References III

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[BT19] Samuel R. Buss and Neil Thapen. DRAT proofs, propagation redundancy, and extended resolution. 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 71–89. Springer, July 2019.

[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.

[CGS17] Kevin K. H. Cheung, Ambros M. Gleixner, and Daniel E. Steffy. Verifying integer programming results. In *Proceedings of the 19th International Conference on Integer Programming and Combinatorial Optimization (IPCO '17)*, volume 10328 of *Lecture Notes in Computer Science*, pages 148–160. Springer, June 2017.

[CHH+17] Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt Jr., Matt Kaufmann, and Peter Schneider-Kamp. Efficient certified RAT verification. In *Proceedings of the 26th International Conference on Automated Deduction (CADE-26)*, volume 10395 of *Lecture Notes in Computer Science*, pages 220–236. Springer, August 2017.

References IV

[CKSW13] William Cook, Thorsten Koch, Daniel E. Steffy, and Kati Wolter. A hybrid branch-and-bound approach for exact rational mixed-integer programming. *Mathematical Programming Computation*, 5(3):305–344, September 2013.

[CMS17] Luís Cruz-Filipe, João P. Marques-Silva, and Peter Schneider-Kamp. Efficient certified resolution proof checking. In *Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '17)*, volume 10205 of *Lecture Notes in Computer Science*, pages 118–135. Springer, April 2017.

[DBB17] Jo Devriendt, Bart Bogaerts, and Maurice Bruynooghe. Symmetric explanation learning: Effective dynamic symmetry handling for SAT. In *Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing (SAT '17)*, volume 10491 of *Lecture Notes in Computer Science*, pages 83–100. Springer, August 2017.

[DBBD16] Jo Devriendt, Bart Bogaerts, Maurice Bruynooghe, and Marc Denecker. Improved static symmetry breaking for SAT. In *Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT '16)*, volume 9710 of *Lecture Notes in Computer Science*, pages 104–122. Springer, July 2016.

References V

[EG21] Leon Eifler and Ambros Gleixner. A computational status update for exact rational mixed integer programming. In *Proceedings of the 22nd International Conference on Integer Programming and Combinatorial Optimization (IPCO '21)*, volume 12707 of *Lecture Notes in Computer Science*, pages 163–177. Springer, May 2021.

[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.

[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.

[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.

References VI

[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.

[GS19] Graeme Gange and Peter Stuckey. Certifying optimality in constraint programming. Presentation at KTH Royal Institute of Technology. Slides available at `https://www.kth.se/polopoly_fs/1.879851.1550484700!/CertifiedCP.pdf`, February 2019.

References VII

[GSD19] Xavier Gillard, Pierre Schaus, and Yves Deville. SolverCheck: Declarative testing of constraints. In *Proceedings of the 25th International Conference on Principles and Practice of Constraint Programming (CP '19)*, volume 11802 of *Lecture Notes in Computer Science*, pages 565–582. Springer, October 2019.

[GSVW14] Maria Garcia de la Banda, Peter J. Stuckey, Pascal Van Hentenryck, and Mark Wallace. The future of optimization technology. *Constraints*, 19(2):126–138, April 2014.

[HHW13a] Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Trimming while checking clausal proofs. In *Proceedings of the 13th International Conference on Formal Methods in Computer-Aided Design (FMCAD '13)*, pages 181–188, October 2013.

[HHW13b] Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In *Proceedings of the 24th International Conference on Automated Deduction (CADE-24)*, volume 7898 of *Lecture Notes in Computer Science*, pages 345–359. Springer, June 2013.

References VIII

[JHB12] Matti Järvisalo, Marijn J. H. Heule, and Armin Biere. Inprocessing rules. In *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR '12)*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, June 2012.

[MMNS11] Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, May 2011.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[MS99] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

[RvBW06] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.

References IX

[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.

[Ver] VeriPB: Verifier for pseudo-Boolean proofs. `https://gitlab.com/MIAOresearch/software/VeriPB`.

[WHH14] Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, July 2014.