

# Certified Symmetry and Dominance Breaking for Combinatorial Optimisation

Jakob Nordström

University of Copenhagen  
and Lund University

*SAT Encodings and Beyond*  
Dagstuhl Seminar 23261  
June 25–30, 2023



Joint AAAI '22 paper 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]
  - Satisfiability modulo theories (SMT) 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<sup>+</sup>18, GSD19, GS19]
- **Software testing** doesn't suffice to resolve this problem
- **Formal verification** techniques cannot deal with level of complexity of modern solvers

# Certified Results with Proof Logging

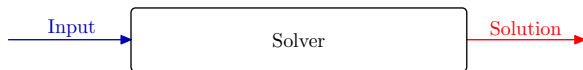
Design **certifying algorithms** [ABM<sup>+</sup>11, MMNS11] that

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

# Certified Results with Proof Logging

Design **certifying algorithms** [ABM<sup>+</sup>11, MMNS11] that

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



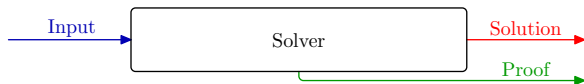
## Workflow:

- 1 Run solver on problem input

# Certified Results with Proof Logging

Design **certifying algorithms** [ABM<sup>+</sup>11, MMNS11] that

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



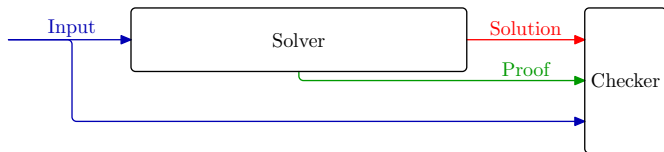
## Workflow:

- 1 Run solver on problem input
- 2 Get as output not only solution but also proof

# Certified Results with Proof Logging

Design **certifying algorithms** [ABM<sup>+</sup>11, MMNS11] that

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



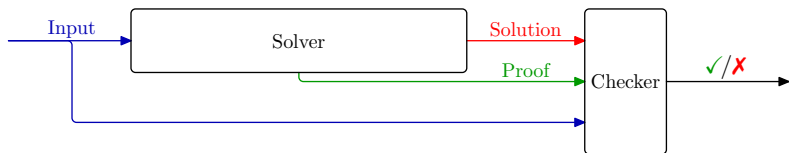
## Workflow:

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

# Certified Results with Proof Logging

Design **certifying algorithms** [ABM<sup>+</sup>11, MMNS11] that

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



## Workflow:

- 1 Run solver on problem input
- 2 Get as output not only solution but also proof
- 3 Feed input + solution + proof to proof checker
- 4 Verify that proof checker says solution 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<sup>+</sup>17]
- ...

Well established — required in main track of SAT competitions



# Yet Another SAT Success Story

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

- DRAT [HHW13a, HHW13b, WHH14]
- GRIT [CMS17]
- LRAT [CHH<sup>+</sup>17]
- ...

Well established — required in main track of SAT competitions

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<sup>+</sup>17]
- ...

Well established — required in main track of SAT competitions

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

# Clausal Proof Logging Approaches

## **Cardinality and pseudo-Boolean reasoning** [SB06, BBH22]

Evaluated on fairly specific crafted benchmarks

More challenging and/or real-world benchmarks would be valuable

# Clausal Proof Logging Approaches

## **Cardinality and pseudo-Boolean reasoning** [SB06, BBH22]

Evaluated on fairly specific crafted benchmarks

More challenging and/or real-world benchmarks would be valuable

## **Gaussian elimination** [PR16, Bry22, SB22]

Problems with proof logging overhead and proof file size

# Clausal Proof Logging Approaches

## **Cardinality and pseudo-Boolean reasoning** [SB06, BBH22]

Evaluated on fairly specific crafted benchmarks

More challenging and/or real-world benchmarks would be valuable

## **Gaussian elimination** [PR16, Bry22, SB22]

Problems with proof logging overhead and proof file size

## **Symmetry handling** [HHW15, TD20]

No fully general method for **symmetry breaking** (i.e., adding constraints to remove symmetric solutions)

Method for **symmetric learning** (i.e., adding symmetric versions of derived constraints) not compatible with SAT preprocessing

# Our Work: Efficient Proof Logging for Symmetry Breaking

Paper *Certified Symmetry and Dominance Breaking for Combinatorial Optimisation* at AAI '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 (and some other problems)
- Some future research directions

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

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

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

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



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

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

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

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

**Pseudo-Boolean formulas**  $F \doteq \bigwedge_{i=1}^m C_i$  are conjunctions of pseudo-Boolean constraints

# Some Types of Pseudo-Boolean Constraints

## 1 Clauses

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

# Some Types of Pseudo-Boolean Constraints

## 1 Clauses

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

## 2 Cardinality constraints

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

# Some Types of Pseudo-Boolean Constraints

## 1 Clauses

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

## 2 Cardinality constraints

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

## 3 General pseudo-Boolean constraints

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

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

**Literal axioms**  $\frac{}{l_i \geq 0}$

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

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

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

**Literal axioms**  $\frac{}{l_i \geq 0}$

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

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

**Toy example:**

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

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

**Literal axioms**  $\frac{}{l_i \geq 0}$

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

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

**Toy example:**

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

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

**Literal axioms**  $\frac{}{l_i \geq 0}$

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

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

**Toy example:**

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



## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

**Literal axioms**  $\frac{}{l_i \geq 0}$

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

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

**Toy example:**

$$\begin{array}{l} \text{Lin comb} \quad \frac{2x + 4y + 2z + w \geq 5 \quad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \quad \bar{z} \geq 0 \\ \text{Lin comb} \quad \frac{}{} \end{array}$$

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

**Literal axioms**  $\frac{}{l_i \geq 0}$

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

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

**Toy example:**

$$\begin{array}{l} \text{Lin comb} \quad \frac{2x + 4y + 2z + w \geq 5 \quad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \quad \bar{z} \geq 0 \\ \text{Lin comb} \quad \frac{6x + 6y + 2z + 3w \geq 9 \quad \bar{z} \geq 0}{6x + 6y + 2z + 3w + 2 \cdot \bar{z} \geq 9 + 2 \cdot 0} \end{array}$$

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

**Literal axioms**  $\frac{}{l_i \geq 0}$

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

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

**Toy example:**

$$\begin{array}{l} \text{Lin comb} \quad \frac{2x + 4y + 2z + w \geq 5 \quad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \quad \bar{z} \geq 0 \\ \text{Lin comb} \quad \frac{6x + 6y + 2z + 3w \geq 9}{6x + 6y + 3w + 2 \geq 9} \end{array}$$

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

**Literal axioms**  $\frac{}{l_i \geq 0}$

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

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

**Toy example:**

$$\begin{array}{l} \text{Lin comb} \quad \frac{2x + 4y + 2z + w \geq 5 \quad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \quad \bar{z} \geq 0 \\ \text{Lin comb} \quad \frac{6x + 6y + 2z + 3w \geq 9}{6x + 6y + 3w \geq 7} \end{array}$$

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

$$\text{Literal axioms} \frac{}{l_i \geq 0}$$

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

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

## Toy example:

$$\begin{array}{l} \text{Lin comb} \frac{2x + 4y + 2z + w \geq 5 \quad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \quad \bar{z} \geq 0 \\ \text{Lin comb} \frac{6x + 6y + 2z + 3w \geq 9}{6x + 6y + 3w \geq 7} \\ \text{Div} \frac{6x + 6y + 3w \geq 7}{2x + 2y + w \geq 2\frac{1}{3}} \end{array}$$

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

$$\text{Literal axioms} \frac{}{l_i \geq 0}$$

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

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

## Toy example:

$$\begin{array}{r} \text{Lin comb} \frac{2x + 4y + 2z + w \geq 5 \quad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \quad \bar{z} \geq 0 \\ \text{Lin comb} \frac{6x + 6y + 2z + 3w \geq 9}{6x + 6y + 3w \geq 7} \\ \text{Div} \frac{6x + 6y + 3w \geq 7}{2x + 2y + w \geq 3} \end{array}$$

## Pseudo-Boolean Reasoning: Cutting Planes [CCT87]

$$\text{Literal axioms} \frac{}{l_i \geq 0}$$

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

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

## Toy example:

$$\begin{array}{l} \text{Lin comb} \frac{2x + 4y + 2z + w \geq 5 \quad 2x + y + w \geq 2}{6x + 6y + 2z + 3w \geq 9} \quad \bar{z} \geq 0 \\ \text{Lin comb} \frac{6x + 6y + 2z + 3w \geq 9}{6x + 6y + 3w \geq 7} \\ \text{Div} \frac{6x + 6y + 3w \geq 7}{2x + 2y + w \geq 3} \end{array}$$

(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  $0 \geq 1$
- Generalize **reverse unit propagation (RUP)** rule [GN03, Van08] to PB constraints — just convenient shorthand for derivation

# 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  $0 \geq 1$
- Generalize **reverse unit propagation (RUP)** rule [GN03, Van08] to PB constraints — just convenient shorthand for derivation
- Also need **extension** rule (analogue of RAT [JHB12]) to deal with, e.g., preprocessing

## 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  $F \wedge C$  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**  $\omega$  (mapping variables to truth values or literals), called a **witness**, for which

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

## 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**  $\omega$  (mapping variables to truth values or literals), called a **witness**, for which

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

- Proof sketch for interesting direction: If  $\alpha$  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**  $\omega$  (mapping variables to truth values or literals), called a **witness**, for which

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

- Proof sketch for interesting direction: If  $\alpha$  satisfies  $F$  but falsifies  $C$ , then  $\alpha \circ \omega$  satisfies  $F \wedge C$
- Implication should be **efficiently verifiable** — every  $D \in (F \wedge C) \upharpoonright_{\omega}$  should follow from  $F \wedge \neg C$  by, e.g.,
  - 1 weakening (addition of literal axioms  $\ell_i \geq 0$ )
  - 2 reverse unit propagation (RUP)
  - 3 explicit derivation presented in proof log



# 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 redundancy rule subsumes RAT)
- Pseudo-Boolean reasoning (by design)
- Gaussian elimination [GN21]
- Subgraph problems [GMN20, GMM<sup>+</sup>20]
- Pseudo-Boolean solving via translation to CNF [GMNO22]
- Core-guided MaxSAT solving [VDB22, BBN<sup>+</sup>23]
- Constraint programming [EGMN20, GMN22]
- **This talk:** 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 redundancy rule subsumes RAT)
- Pseudo-Boolean reasoning (by design)
- Gaussian elimination [GN21]
- Subgraph problems [GMN20, GMM<sup>+</sup>20]
- Pseudo-Boolean solving via translation to CNF [GMNO22]
- Core-guided MaxSAT solving [VDB22, BBN<sup>+</sup>23]
- Constraint programming [EGMN20, GMN22]
- **This talk:** Symmetry and dominance breaking [BGMN22]

More info in tutorial [Combinatorial solving with provably correct results](https://youtu.be/s_5BIi4I22w) ([https://youtu.be/s\\_5BIi4I22w](https://youtu.be/s_5BIi4I22w)) on pseudo-Boolean proof logging

# The Challenge of Symmetries

(Syntactic) symmetry: substitution  $\sigma$  preserving  $F$  ( $F \upharpoonright_{\sigma} \doteq F$ )

- 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  $\sigma$  preserving  $F$  ( $F \upharpoonright_{\sigma} \doteq F$ )

- 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  $\sigma$  preserving  $F$  ( $F \upharpoonright_{\sigma} \doteq F$ )

- 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  $f = \sum_i w_i l_i$  (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  $f = \sum_i w_i l_i$  (for  $w_i \in \mathbb{N}$ ) subject to constraints in  $F$

### Proof of optimality:

- $F$  satisfied by  $\alpha$
- $F \wedge (\sum_i w_i l_i < \sum_i w_i \cdot \alpha(l_i))$  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  $f = \sum_i w_i l_i$  (for  $w_i \in \mathbb{N}$ ) subject to constraints in  $F$

### Proof of optimality:

- $F$  satisfied by  $\alpha$
- $F \wedge (\sum_i w_i l_i < \sum_i w_i \cdot \alpha(l_i))$  is infeasible

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

# Proof Logging for Optimisation Problems

How does proof system change?

# 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  $\alpha$  has been found, allow constraint  $\sum_i w_i l_i < \sum_i w_i \cdot \alpha(l_i)$  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  $\alpha$  has been found, allow constraint  $\sum_i w_i l_i < \sum_i w_i \cdot \alpha(l_i)$  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  $\alpha$  has been found, allow constraint  $\sum_i w_i l_i < \sum_i w_i \cdot \alpha(l_i)$  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  $\omega$  such that

$$F \wedge \neg C \models (F \wedge C) \upharpoonright_{\omega} \wedge f \upharpoonright_{\omega} \leq f$$



# Redundance and Dominance Rules

Redundance-based strengthening, optimisation version

Add constraint  $C$  to formula  $F$  if exists witness substitution  $\omega$  such that

$$F \wedge \neg C \models (F \wedge C)\downarrow_{\omega} \wedge f\downarrow_{\omega} \leq f$$

# Redundance and Dominance Rules

Redundance-based strengthening, optimisation version

Add constraint  $C$  to formula  $F$  if exists witness substitution  $\omega$  such that

$$F \wedge \neg C \models (F \wedge C) \upharpoonright_{\omega} \wedge f \upharpoonright_{\omega} \leq f$$

Can be more aggressive if witness  $\omega$  **strictly improves** solution

# Redundance and Dominance Rules

## Redundance-based strengthening, optimisation version

Add constraint  $C$  to formula  $F$  if exists witness substitution  $\omega$  such that

$$F \wedge \neg C \models (F \wedge C)\downarrow_{\omega} \wedge f\downarrow_{\omega} \leq f$$

Can be more aggressive if witness  $\omega$  **strictly improves** solution

## Dominance-based strengthening (simplified)

Add constraint  $C$  to formula  $F$  if exists witness substitution  $\omega$  such that

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

# Redundance and Dominance Rules

## Redundance-based strengthening, optimisation version

Add constraint  $C$  to formula  $F$  if exists witness substitution  $\omega$  such that

$$F \wedge \neg C \models (F \wedge C)\downarrow_{\omega} \wedge f\downarrow_{\omega} \leq f$$

Can be more aggressive if witness  $\omega$  **strictly improves** solution

## Dominance-based strengthening (simplified)

Add constraint  $C$  to formula  $F$  if exists witness substitution  $\omega$  such that

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

- Applying  $\omega$  should **strictly decrease**  $f$
- If so, don't need to show that  $C\downarrow_{\omega}$  implied!

# Soundness of Dominance Rule

## Dominance-based strengthening (simplified)

Add constraint  $C$  to formula  $F$  if exists witness substitution  $\omega$  such that

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

Why is this sound?

# Soundness of Dominance Rule

## Dominance-based strengthening (simplified)

Add constraint  $C$  to formula  $F$  if exists witness substitution  $\omega$  such that

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

Why is this sound?

- 1 Suppose  $\alpha$  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  $\omega$  such that

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

Why is this sound?

- 1 Suppose  $\alpha$  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  $\omega$  such that

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

Why is this sound?

- 1 Suppose  $\alpha$  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  $\omega$  such that

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

Why is this sound?

- ① Suppose  $\alpha$  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  $\omega$  such that

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

Why is this sound?

- 1 Suppose  $\alpha$  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  $\omega$  such that

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

Why is this sound?

- 1 Suppose  $\alpha$  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  $\omega$  such that

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

Why is this sound?

- 1 Suppose  $\alpha$  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  $\omega$  such that

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

Why is this sound?

- 1 Suppose  $\alpha$  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  $\alpha'$  satisfying  $F \wedge C$

# Strength of Dominance Rule

## Dominance-based strengthening (stronger, still simplified)

If  $C_1, C_2, \dots, C_{m-1}$  have been derived from  $F$  (maybe using dominance), then can derive also  $C_m$  if exists witness substitution  $\omega$  such that

$$F \wedge \bigwedge_{i=1}^{m-1} C_i \wedge \neg C_m \models F \upharpoonright_{\omega} \wedge f \upharpoonright_{\omega} < f$$

Only consider original formula — no need to show that any  $C_i \upharpoonright_{\omega}$  implied!

# Strength of Dominance Rule

## Dominance-based strengthening (stronger, still simplified)

If  $C_1, C_2, \dots, C_{m-1}$  have been derived from  $F$  (maybe using dominance), then can derive also  $C_m$  if exists witness substitution  $\omega$  such that

$$F \wedge \bigwedge_{i=1}^{m-1} C_i \wedge \neg C_m \models F|_{\omega} \wedge f|_{\omega} < f$$

Only consider original formula — no need to show that any  $C_i|_{\omega}$  implied!

Now why is **this** sound?

- Same inductive proof as before, but nested
- Or pick  $\alpha$  satisfying  $F$  and minimizing  $f$  and argue by contradiction

# Strength of Dominance Rule

## Dominance-based strengthening (stronger, still simplified)

If  $C_1, C_2, \dots, C_{m-1}$  have been derived from  $F$  (maybe using dominance), then can derive also  $C_m$  if exists witness substitution  $\omega$  such that

$$F \wedge \bigwedge_{i=1}^{m-1} C_i \wedge \neg C_m \models F \upharpoonright_{\omega} \wedge f \upharpoonright_{\omega} < f$$

Only consider original formula — no need to show that any  $C_i \upharpoonright_{\omega}$  implied!

Now why is **this** sound?

- Same inductive proof as before, but nested
- Or pick  $\alpha$  satisfying  $F$  and minimizing  $f$  and argue by contradiction

Further extensions:

- Define dominance rule w.r.t. order independent of objective function
- Switch between different orders in same proof
- See [BGMN22] for details



# 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**

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

# 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**

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

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

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

# Breaking Symmetries With the Dominance Rule (1/2)

## Theorem

$C_\sigma \doteq f \leq f|_\sigma$  can be derived from  $F$  using dominance with witness  $\sigma$

$$F \wedge \neg C_\sigma \models F|_\sigma \wedge f|_\sigma < f$$

# Breaking Symmetries With the Dominance Rule (1/2)

## Theorem

$C_\sigma \doteq f \leq f|_\sigma$  can be derived from  $F$  using dominance with witness  $\sigma$

$$F \wedge \neg C_\sigma \models F|_\sigma \wedge f|_\sigma < f$$

## Comparison to DRAT-style proofs

Redundance-based strengthening can be used analogously to [HHW15]

# Breaking Symmetries With the Dominance Rule (1/2)

## Theorem

$C_\sigma \doteq f \leq f|_\sigma$  can be derived from  $F$  using dominance with witness  $\sigma$

$$F \wedge \neg C_\sigma \models F|_\sigma \wedge f|_\sigma < f$$

## Comparison to DRAT-style proofs

Redundance-based strengthening can be used analogously to [HHW15]

- but only guaranteed to work for breaking **single symmetry**  $\sigma$

# Breaking Symmetries With the Dominance Rule (1/2)

## Theorem

$C_\sigma \doteq f \leq f|_\sigma$  can be derived from  $F$  using dominance with witness  $\sigma$

$$F \wedge \neg C_\sigma \models F|_\sigma \wedge f|_\sigma < f$$

## Comparison to DRAT-style proofs

Redundance-based strengthening can be used analogously to [HHW15]

- but only guaranteed to work for breaking **single symmetry**  $\sigma$
- if  $\sigma$  is **involution** (i.e., its own inverse)

# Breaking Symmetries With the Dominance Rule (1/2)

## Theorem

$C_\sigma \doteq f \leq f|_\sigma$  can be derived from  $F$  using dominance with witness  $\sigma$

$$F \wedge \neg C_\sigma \models F|_\sigma \wedge f|_\sigma < f$$

## Comparison to DRAT-style proofs

Redundance-based strengthening can be used analogously to [HHW15]

- but only guaranteed to work for breaking **single symmetry**  $\sigma$
- if  $\sigma$  is **involution** (i.e., its own inverse)
- not known how to deal with symmetries that are complex or interact



# Breaking Symmetries With the Dominance Rule (2/2)

Breaking symmetries with the dominance rule

- Surprisingly **simple**

# Breaking Symmetries With the Dominance Rule (2/2)

Breaking symmetries with the dominance rule

- Surprisingly **simple**
- **Generalizes well** (compared to redundancy-based symmetry breaking)

# Breaking Symmetries With the Dominance Rule (2/2)

Breaking symmetries with the dominance rule

- Surprisingly **simple**
- **Generalizes well** (compared to redundancy-based symmetry breaking)
  - Works for **arbitrary symmetries**

# Breaking Symmetries With the Dominance Rule (2/2)

Breaking symmetries with the dominance rule

- Surprisingly **simple**
- **Generalizes well** (compared to redundancy-based symmetry breaking)
  - Works for **arbitrary symmetries**
  - Works for **multiple symmetries** (ignore previously derived constraints)

$$F \wedge C_\sigma \wedge \neg C_\tau \models F|_\tau \wedge f|_\tau < f$$

# Breaking Symmetries With the Dominance Rule (2/2)

Breaking symmetries with the dominance rule

- Surprisingly **simple**
- **Generalizes well** (compared to redundancy-based symmetry breaking)
  - Works for **arbitrary symmetries**
  - Works for **multiple symmetries** (ignore previously derived constraints)

$$F \wedge C_\sigma \wedge \neg C_\tau \models F \upharpoonright_\tau \wedge f \upharpoonright_\tau < f$$

Why does it work?

- Witness need not satisfy all derived constraints
- Sufficient to just produce “better” assignment

# Applied Symmetry Breaking for SAT Solving

- 1 Break symmetries with `BREAKID` and get proofs for symmetry breaking clauses
- 2 Concatenate with CDCL solver proof (DRAT rewritten in `VERIPB` format)

# Applied Symmetry Breaking for SAT Solving

- 1 Break symmetries with `BREAKID` and get proofs for symmetry breaking clauses
- 2 Concatenate with CDCL solver proof (DRAT rewritten in `VERIPB` format)

## Short dictionary for DRAT-to-VeriPB translations

DRAT	VERIPB
1	x1
-2	$\sim x2$
1 -2 3 0	1 x1 1 $\sim x2$ 1 x3 $\geq 1$ ;
1 -2 3 0 is RUP	rup 1 x1 1 $\sim x2$ 1 x3 $\geq 1$ ;
1 -2 3 0 is RAT	red 1 x1 1 $\sim x2$ 1 x3 $\geq 1$ ; x1 $\rightarrow$ 1

# Applied Symmetry Breaking for SAT Solving

- 1 Break symmetries with `BREAKID` and get proofs for symmetry breaking clauses
- 2 Concatenate with CDCL solver proof (DRAT rewritten in `VERIPB` format)

## Short dictionary for DRAT-to-VeriPB translations

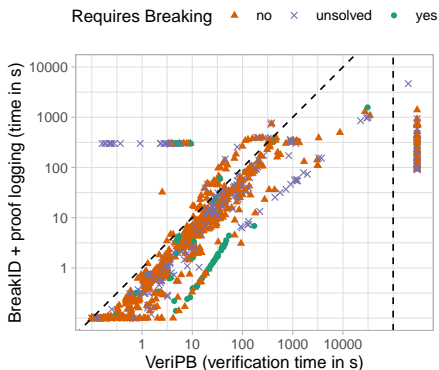
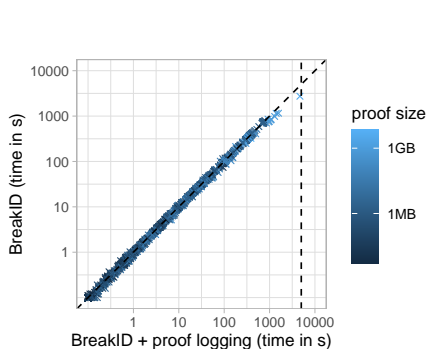
DRAT	VERIPB
1	x1
-2	$\sim x2$
1 -2 3 0	1 x1 1 $\sim x2$ 1 x3 $\geq 1$ ;
1 -2 3 0 is RUP	rup 1 x1 1 $\sim x2$ 1 x3 $\geq 1$ ;
1 -2 3 0 is RAT	red 1 x1 1 $\sim x2$ 1 x3 $\geq 1$ ; x1 $\rightarrow 1$

(Or use `KISSAT` fork with `VERIPB` proof logging at [https://gitlab.com/MIA0research/tools-and-utilities/kissat\\_fork](https://gitlab.com/MIA0research/tools-and-utilities/kissat_fork))



# Experimental Evaluation

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



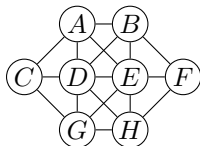
- proof logging overhead negligible
- verification at most 20 times slower than solving for 95% of instances

# Symmetry Breaking for Constraint Programming

## Crystal Maze puzzle

Place numbers 1 to 8 without repetition

Adjacent circles mustn't have consecutive numbers

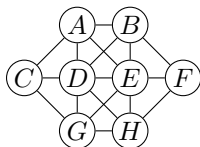


# Symmetry Breaking for Constraint Programming

## Crystal Maze puzzle

Place numbers 1 to 8 without repetition

Adjacent circles mustn't have consecutive numbers



Without loss of generality:

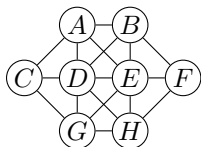
- $A < G$  (horizontal mirror symmetry)
- $A < B$  (vertical mirror symmetry)
- $A \leq 4$  (value symmetry)

# Symmetry Breaking for Constraint Programming

## Crystal Maze puzzle

Place numbers 1 to 8 without repetition

Adjacent circles mustn't have consecutive numbers



Without loss of generality:

- $A < G$  (horizontal mirror symmetry)
- $A < B$  (vertical mirror symmetry)
- $A \leq 4$  (value symmetry)

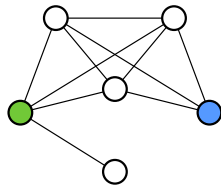
*Technical challenge:* integer-valued variables

See [GMN22] for more detailed discussion

# Dominance Breaking for Maximum Clique Solving

## Maximum clique solving

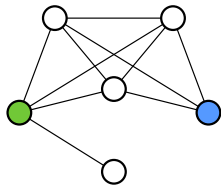
Find largest fully connected component



# Dominance Breaking for Maximum Clique Solving

## Maximum clique solving

Find largest fully connected component



## Lazy global domination [MP16]

Only consider green and not blue vertex

(since every neighbour of blue is also neighbour of green)

*Technical challenge:* vertex domination detected only lazily during search  
Dominance rule (rather than redundancy rule) really helpful here

# 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 [BMM<sup>+</sup> 23]*)

# 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 [BMM<sup>+</sup> 23]*)

## Proof logging for other combinatorial problems and techniques

- Symmetric learning and recycling (substitution) of subproofs
- MaxSAT solving and PB optimization (*work in progress [VDB22, BBN<sup>+</sup> 23]*)
- Mixed integer linear programming (*work on SCIP in [CGS17, EG21]*)
- Satisfiability modulo theories (SMT) solving (*work by Bjørner and others*)
- Planning (*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 [BMM<sup>+</sup> 23]*)

## Proof logging for other combinatorial problems and techniques

- Symmetric learning and recycling (substitution) of subproofs
- MaxSAT solving and PB optimization (*work in progress [VDB22, BBN<sup>+</sup> 23]*)
- Mixed integer linear programming (*work on SCIP in [CGS17, EG21]*)
- Satisfiability modulo theories (SMT) solving (*work by Bjørner and others*)
- Planning (*work in progress*)

## 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 [BMM<sup>+</sup> 23]*)

## Proof logging for other combinatorial problems and techniques

- Symmetric learning and recycling (substitution) of subproofs
- MaxSAT solving and PB optimization (*work in progress [VDB22, BBN<sup>+</sup> 23]*)
- Mixed integer linear programming (*work on SCIP in [CGS17, EG21]*)
- Satisfiability modulo theories (SMT) solving (*work by Bjørner and others*)
- Planning (*work in progress*)

## 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 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 with extensions

*Thank you for your attention!*

# References I

- [ABM<sup>+</sup>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<sup>+</sup>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

- [BBH22] Randal E. Bryant, Armin Biere, and Marijn J. H. Heule. Clausal proofs for pseudo-Boolean reasoning. In *Proceedings of the 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '22)*, volume 13243 of *Lecture Notes in Computer Science*, pages 443–461. Springer, April 2022.
- [BBN<sup>+</sup>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)*, July 2023. To appear.
- [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.

## References III

- [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.
- [BMM<sup>+</sup>23] Bart Bogaerts, Ciaran McCreesh, Magnus O. Myreen, Jakob Nordström, Andy Oertel, and Yong Kiam Tan. Documentation of VeriPB and CakePB for the SAT competition 2023. Available at <https://satcompetition.github.io/2023/checkers.html>, March 2023.
- [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>.
- [Bry22] Randal E. Bryant. TBUDDY: a proof-generating BDD package. EasyChair Preprint 8471, July 2022. Available at <https://easychair.org/publications/preprint/DbRN>.

# References IV

- [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<sup>+</sup>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.
- [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.



# References V

- [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.
- [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.

# References VI

- [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<sup>+</sup>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.
- [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.

## References VII

- [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.
- [GN03] Evgueni Goldberg and Yakov Novikov. Verification of proofs of unsatisfiability for CNF formulas. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '03)*, pages 886–891, March 2003.
- [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](https://www.kth.se/polopoly_fs/1.879851.1550484700!/CertifiedCP.pdf), February 2019.

## References VIII

- [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 IX

- [HHW15] Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Expressing symmetry breaking in DRAT proofs. In *Proceedings of the 25th International Conference on Automated Deduction (CADE-25)*, volume 9195 of *Lecture Notes in Computer Science*, pages 591–606. Springer, August 2015.
- [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.
- [MP16] Ciaran McCreesh and Patrick Prosser. Finding maximum  $k$ -cliques faster using lazy global domination. In *Proceedings of the 9th Annual Symposium on Combinatorial Search (SOCS '16)*, pages 72–80, July 2016.
- [PR16] Tobias Philipp and Adrián Rebola-Pardo. DRAT proofs for XOR reasoning. In *Proceedings of the 15th European Conference on Logics in Artificial Intelligence (JELIA '16)*, volume 10021 of *Lecture Notes in Computer Science*, pages 415–429. Springer, November 2016.

# References X

- [RvBW06] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.
- [SB06] Carsten Sinz and Armin Biere. Extended resolution proofs for conjoining BDDs. In *Proceedings of the 1st International Computer Science Symposium in Russia (CSR '06)*, volume 3967 of *Lecture Notes in Computer Science*, pages 600–611. Springer, June 2006.
- [SB22] Mate Soos and Randal E. Bryant. Combining CDCL, Gauss-Jordan elimination, and proof generation. *EasyChair Preprint 8497*, July 2022. Available at <https://easychair.org/publications/preprint/4rGK>.
- [TD20] Rodrigue Konan Tchinda and Clémentin Tayou Djamégni. On certifying the UNSAT result of dynamic symmetry-handling-based SAT solvers. *Constraints*, 25(3–4):251–279, December 2020.
- [Van08] Allen Van Gelder. Verifying RUP proofs of propositional unsatisfiability. In *10th International Symposium on Artificial Intelligence and Mathematics (ISAIM '08)*, 2008. Available at <http://isaim2008.unl.edu/index.php?page=proceedings>.

# References XI

- [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/MIA0research/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.