Certified Core-Guided MaxSAT Solving

Andy Oertel

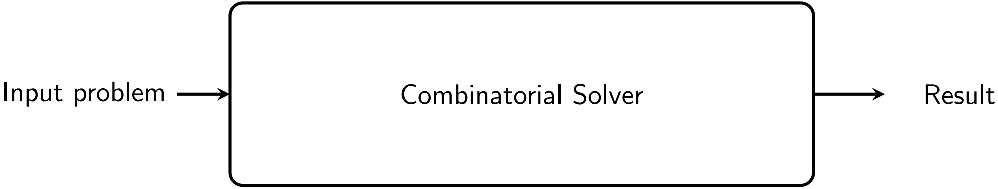
Lund University and
University of Copenhagen

29th International Conference on Automated Deduction

July 2, 2023

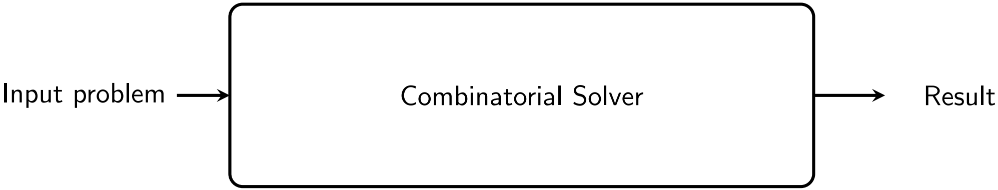
Joint work with Jeremias Berg, Bart Bogaerts, Jakob Nordström and Dieter Vandesande

Combinatorial Solving & Optimization



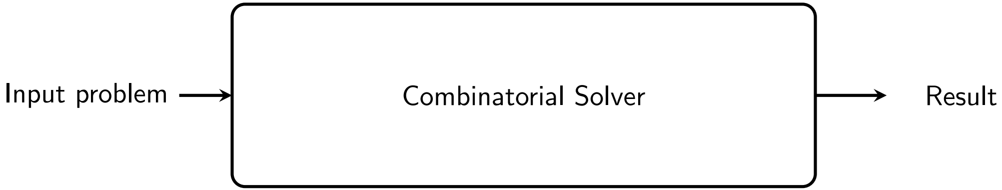
- Problems over discrete variables
- Optimization with objective function
- More or less impossible to solve in theory (NP-hard)

Combinatorial Solving & Optimization



- Problems over discrete variables
- Optimization with objective function
- More or less impossible to solve in theory (NP-hard)
- Often (not always) manageable in practice
- But algorithms quite complex

Combinatorial Solving & Optimization



- Problems over discrete variables
- Optimization with objective function
- More or less impossible to solve in theory (NP-hard)
- Often (not always) manageable in practice
- But algorithms quite complex

How do we know if problem was solved correctly?

Correctness of Combinatorial Solvers

Testing:

- Can only show presence of bugs, not absence
- No guarantees of correctness

Correctness of Combinatorial Solvers

Testing:

- Can only show presence of bugs, not absence
- No guarantees of correctness

Formal verification:

- Provides proof that solver adheres to formal specification
- Modern solvers far too complex for formal verification techniques to scale

Correctness of Combinatorial Solvers

Testing:

- Can only show presence of bugs, not absence
- No guarantees of correctness

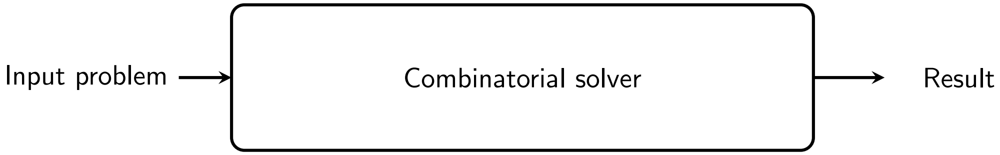
Formal verification:

- Provides proof that solver adheres to formal specification
- Modern solvers far too complex for formal verification techniques to scale

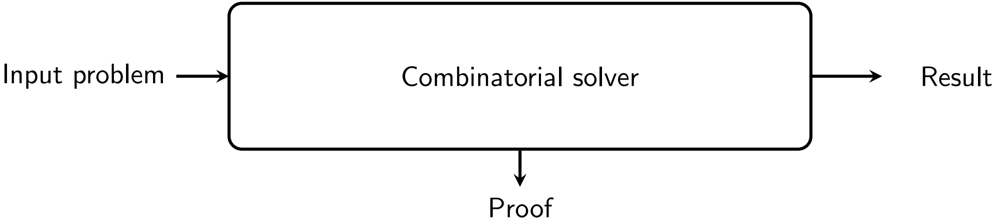
Proof logging (our approach):

- Guarantee that execution was correct
- Moderate overhead for implementing solver

Certifying Results with Proof Logging

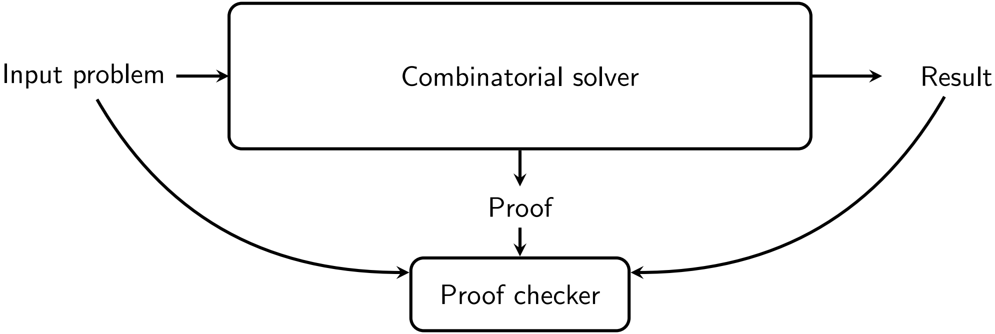


Certifying Results with Proof Logging



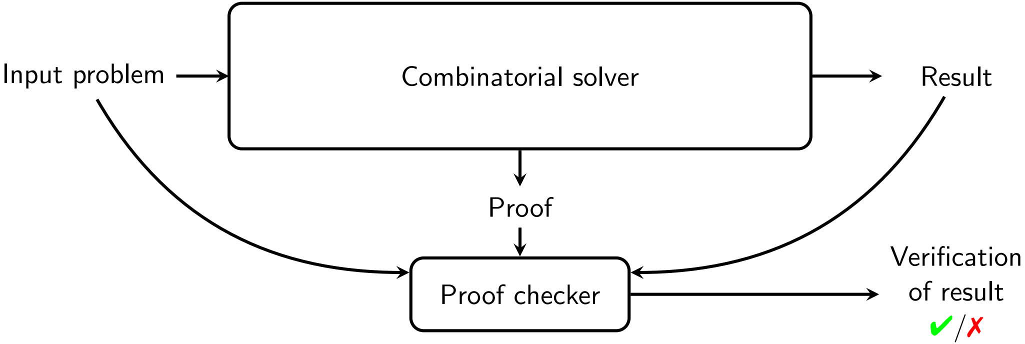
- Solver generates proof/certificate of correctness for result

Certifying Results with Proof Logging



- Solver generates proof/certificate of correctness for result
- Proof checker checks if reasoning to get result is correct based on the proof

Certifying Results with Proof Logging



- Solver generates proof/certificate of correctness for result
- Proof checker checks if reasoning to get result is correct based on the proof

Our Focus: Maximum Satisfiability (MaxSAT) Solving

$$
\begin{aligned}
\min\ & 2x_1 + x_2 \\
\text{s.t. } & x_1 \lor \overline{z} \\
& z \lor x_2
\end{aligned}
\longrightarrow \boxed{\text{MaxSAT solver}} \longrightarrow
$$

Result:
optimum 1

- Minimize objective subject to satisfying formula in conjunctive normal form (CNF)

Our Focus: Maximum Satisfiability (MaxSAT) Solving

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t.}\ & x_1 \vee \overline{z} \\ & z \vee x_2 \end{aligned} \longrightarrow \boxed{\text{MaxSAT solver}} \longrightarrow \begin{array}{c}\text{Result:} \\ \text{optimum } 1\end{array}$$

- Minimize objective subject to satisfying formula in conjunctive normal form (CNF)
- Equivalently: Maximize satisfied soft clauses subject to satisfying hard clauses

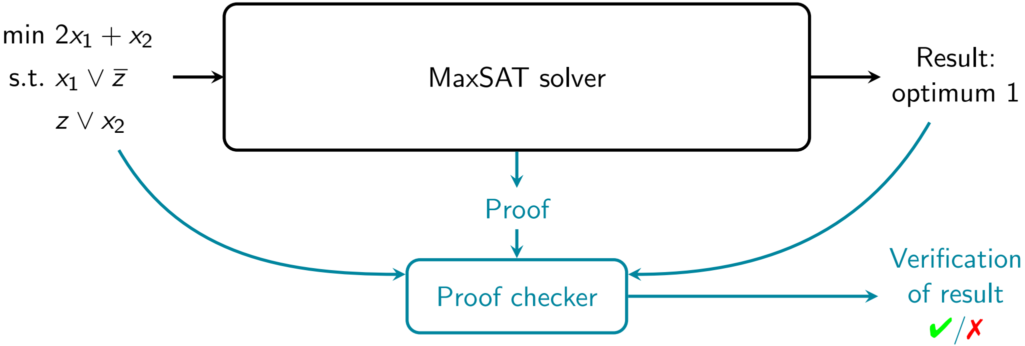
Our Focus: Maximum Satisfiability (MaxSAT) Solving

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t.}\ & x_1 \vee \overline{z} \\ & z \vee x_2 \end{aligned}$$

→ MaxSAT solver → Result: optimum 1

- Minimize objective subject to satisfying formula in conjunctive normal form (CNF)
- Equivalently: Maximize satisfied soft clauses subject to satisfying hard clauses
- Main approaches:
  - Solution-improving or linear SAT-UNSAT search [ES06, LP10, PRB18]
  - Implicit hitting set (IHS) search [DB13a, DB13b]
  - **Core-guided search** [FM06, NB14, ADR15, AG17]

Certified Maximum Satisfiability (MaxSAT) Solving



- **This work:** Certification of state-of-the-art core-guided MaxSAT solving

Rest of This Talk

1. Description of state-of-the-art core-guided MaxSAT solving
2. Our contribution: Adding proof logging to core-guided MaxSAT solving
3. Experimental evaluation
4. Conclusion

Basic Notation

- Boolean variable x: Domain 0 (false) and 1 (true)
- Literal ℓ: x or negation $\overline{x} = 1 - x$
- Pseudo-Boolean (PB) constraint: Integer linear inequality over literals

$$3x_1 + 2x_2 + 5\overline{x}_3 \geq 5$$

Basic Notation

- Boolean variable x: Domain 0 (false) and 1 (true)
- Literal ℓ: x or negation $\overline{x} = 1 - x$
- Pseudo-Boolean (PB) constraint: Integer linear inequality over literals

$$3x_1 + 2x_2 + 5\overline{x}_3 \geq 5$$

- Pseudo-Boolean equality constraint: Syntactic sugar for 2 inequalities

$$3x_1 + 2x_2 + 5\overline{x}_3 = 5 \longrightarrow \begin{array}{l} 3x_1 + 2x_2 + 5\overline{x}_3 \geq 5 \\ 3x_1 + 2x_2 + 5\overline{x}_3 \leq 5 \end{array}$$

Basic Notation

- Boolean variable x: Domain 0 (false) and 1 (true)
- Literal ℓ: x or negation $\overline{x} = 1 - x$
- Pseudo-Boolean (PB) constraint: Integer linear inequality over literals

$$3x_1 + 2x_2 + 5\overline{x}_3 \geq 5$$

- Pseudo-Boolean equality constraint: Syntactic sugar for 2 inequalities

$$3x_1 + 2x_2 + 5\overline{x}_3 = 5 \longrightarrow \begin{matrix} 3x_1 + 2x_2 + 5\overline{x}_3 \geq 5 \\ 3x_1 + 2x_2 + 5\overline{x}_3 \leq 5 \end{matrix}$$

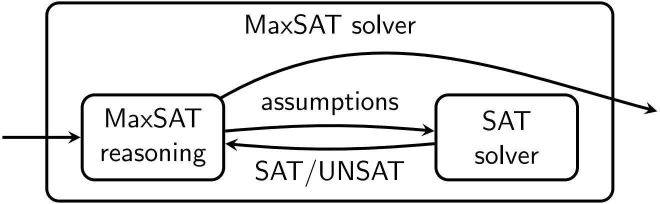
- Clause: Disjunction of literals or at-least-one constraint

$$x_1 \vee \overline{x}_2 \vee \overline{x}_3 \iff x_1 + \overline{x}_2 + \overline{x}_3 \geq 1$$

- CNF formula can be viewed as a collection of pseudo-Boolean constraints

OLL-Style Core-Guided MaxSAT Solving [MDM14]

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t.}\ & x_1 + \bar{z} \geq 1 \\ & z + x_2 \geq 1 \end{aligned}$$

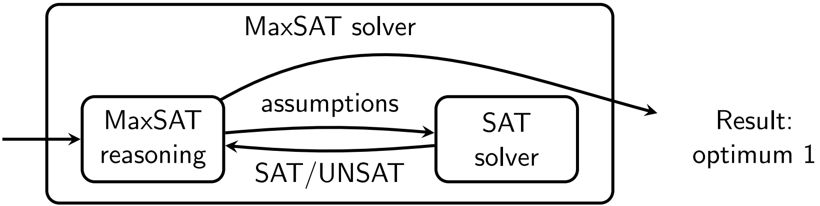


Result:
optimum 1

1. Try best objective value (using optimistic assumptions about the objective)

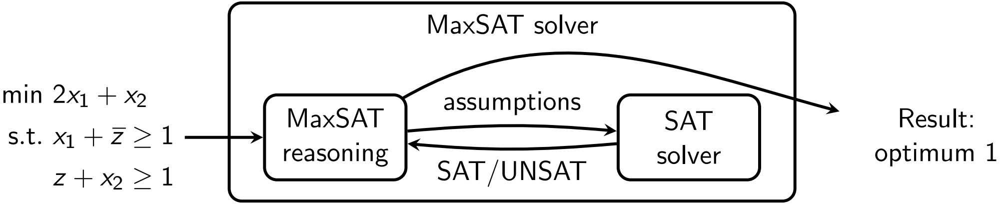
OLL-Style Core-Guided MaxSAT Solving [MDM14]

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t. } & x_1 + \bar{z} \geq 1 \\ & z + x_2 \geq 1 \end{aligned}$$



1. Try best objective value (using optimistic assumptions about the objective)
2. Succeed or find core (clause identifying set of too optimistic assumptions)

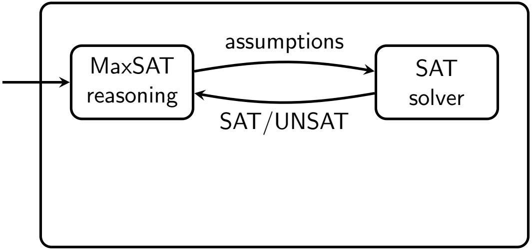
OLL-Style Core-Guided MaxSAT Solving [MDM14]



1. Try best objective value (using optimistic assumptions about the objective)
2. Succeed or find core (clause identifying set of too optimistic assumptions)
3. Reformulate objective and goto 1.

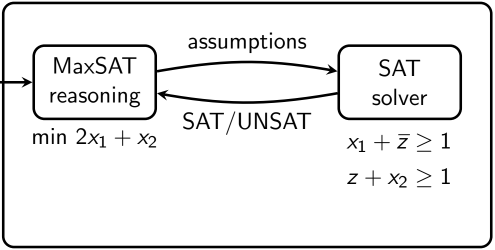
Example: Core-Guided MaxSAT Solving (1/2)

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t. } & x_1 + \bar{z} \geq 1 \\ & z + x_2 \geq 1 \end{aligned}$$

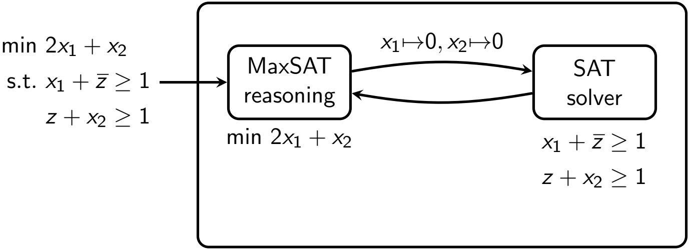


Example: Core-Guided MaxSAT Solving (1/2)

$$\begin{aligned} &\min\ 2x_1 + x_2 \\ &\text{s.t. } x_1 + \bar{z} \geq 1 \\ &\quad\ \ z + x_2 \geq 1 \end{aligned}$$

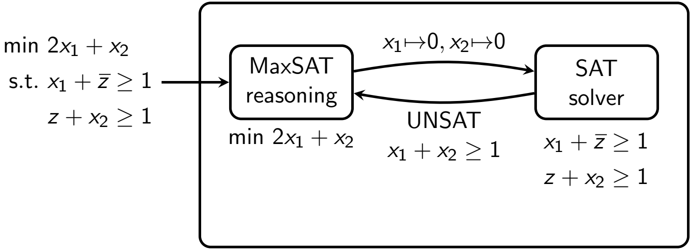


Example: Core-Guided MaxSAT Solving (1/2)



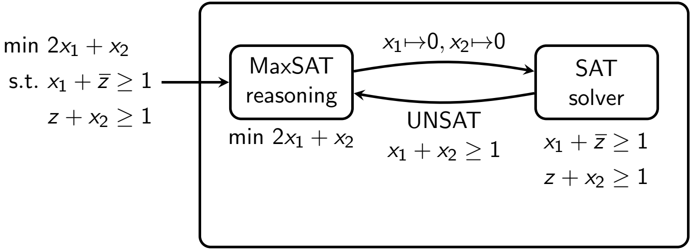
- Call SAT solver with assumptions $x_1 \mapsto 0, x_2 \mapsto 0$

Example: Core-Guided MaxSAT Solving (1/2)



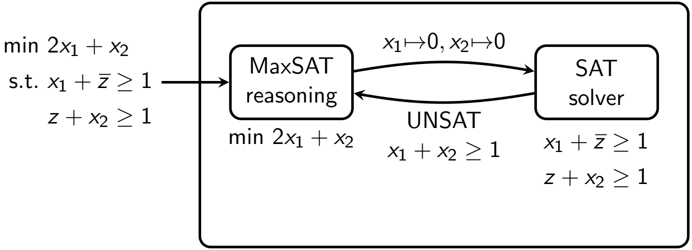
- Call SAT solver with assumptions $x_1 \mapsto 0, x_2 \mapsto 0$
- SAT solver returns UNSAT with core $C : x_1 + x_2 \geq 1$

Example: Core-Guided MaxSAT Solving (1/2)



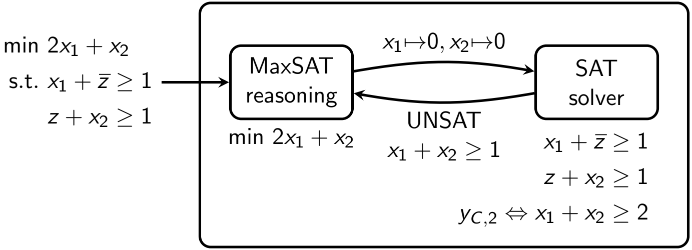
- Call SAT solver with assumptions $x_1 {\mapsto} 0, x_2 {\mapsto} 0$
- SAT solver returns UNSAT with core $C : x_1 + x_2 \geq 1$
- Introduce counter variables $y_{C,1} \Leftrightarrow x_1 + x_2 \geq 1$ and $y_{C,2} \Leftrightarrow x_1 + x_2 \geq 2$

Example: Core-Guided MaxSAT Solving (1/2)



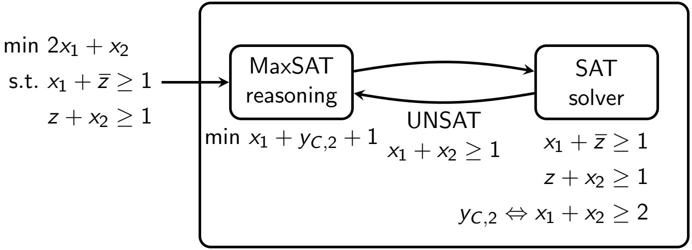
- Call SAT solver with assumptions $x_1 \mapsto 0, x_2 \mapsto 0$
- SAT solver returns UNSAT with core $C : x_1 + x_2 \geq 1$
- Introduce counter variables $y_{C,1}$, fixed to 1, and $y_{C,2} \Leftrightarrow x_1 + x_2 \geq 2$

Example: Core-Guided MaxSAT Solving (1/2)



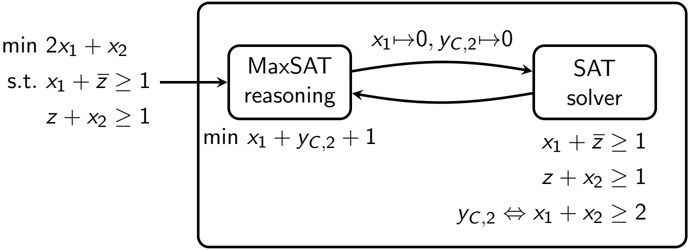
- Call SAT solver with assumptions $x_1 \mapsto 0, x_2 \mapsto 0$
- SAT solver returns UNSAT with core $C : x_1 + x_2 \geq 1$
- Introduce counter variables $y_{C,1}$, fixed to 1, and $y_{C,2} \Leftrightarrow x_1 + x_2 \geq 2$
- Definition of counter variables encoded to CNF using totalizers

Example: Core-Guided MaxSAT Solving (1/2)



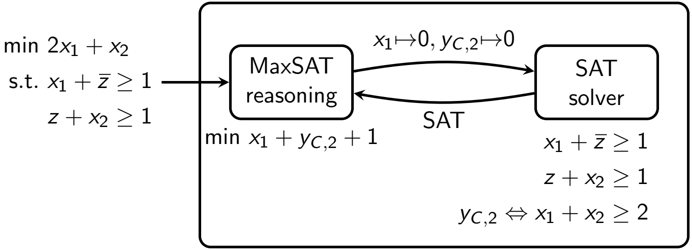
- Call SAT solver with assumptions $x_1 \mapsto 0, x_2 \mapsto 0$
- SAT solver returns UNSAT with core $C : x_1 + x_2 \geq 1$
- Introduce counter variables $y_{C,1}$, fixed to 1, and $y_{C,2} \Leftrightarrow x_1 + x_2 \geq 2$
- Definition of counter variables encoded to CNF using totalizers
- Using $x_1 + x_2 = 1 + y_{C,2}$, reformulate objective from $2x_1 + x_2$ to $x_1 + y_{C,2} + 1$

Example: Core-Guided MaxSAT Solving (2/2)



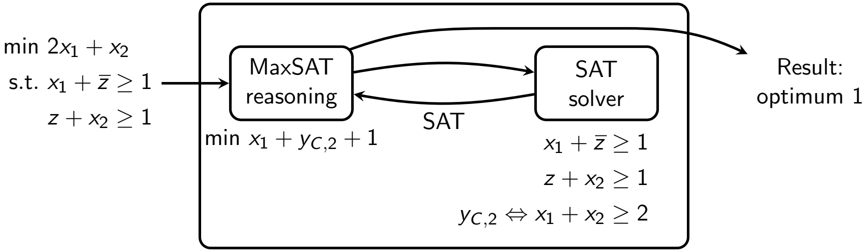
- Call SAT solver with assumptions $x_1 \mapsto 0, y_{C,2} \mapsto 0$

Example: Core-Guided MaxSAT Solving (2/2)



- Call SAT solver with assumptions $x_1 \mapsto 0, y_{C,2} \mapsto 0$
- SAT solver returns SAT with solution $x_1 \mapsto 0, x_2 \mapsto 1, y_{C,2} \mapsto 0, z \mapsto 0$

Example: Core-Guided MaxSAT Solving (2/2)



- Call SAT solver with assumptions $x_1 \mapsto 0, y_{C,2} \mapsto 0$
- SAT solver returns SAT with solution $x_1 \mapsto 0, x_2 \mapsto 1, y_{C,2} \mapsto 0, z \mapsto 0$
- Best possible assumptions about objective satisfy all constraints
- Optimal solution found with value 1

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

$$\frac{}{x \geq 0} \qquad \frac{}{\overline{x} \geq 0}$$

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

$$\overline{ x \geq 0 } \qquad \overline{ \overline{x} \geq 0 }$$

- Addition

$$\text{Addition}\ \frac{x_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3 \qquad \overline{x}_2 + 3x_3 \geq 3}{x_1 + 3\overline{x}_2 + x_3 \geq 4}$$

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

$$\frac{}{x \geq 0} \qquad \frac{}{\overline{x} \geq 0}$$

- Addition

$$\text{Addition}\ \frac{x_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3 \qquad \overline{x}_2 + 3x_3 \geq 3}{x_1 + 3\overline{x}_2 + x_3 \geq 4}$$

- Multiplication

$$\text{Multiply by 2}\ \frac{x_1 + 2\overline{x}_2 \geq 3}{2x_1 + 4\overline{x}_2 \geq 6}$$

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

$$\frac{}{x \geq 0} \qquad \frac{}{\overline{x} \geq 0}$$

- Addition

$$\text{Addition}\ \frac{x_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3 \qquad \overline{x}_2 + 3x_3 \geq 3}{x_1 + 3\overline{x}_2 + x_3 \geq 4}$$

- Multiplication

$$\text{Multiply by 2}\ \frac{x_1 + 2\overline{x}_2 \geq 3}{2x_1 + 4\overline{x}_2 \geq 6}$$

- Division (and rounding up)

$$\text{Divide by 2}\ \frac{2x_1 + 2\overline{x}_2 + 4x_3 \geq 5}{x_1 + \overline{x}_2 + 2x_3 \geq 2.5}$$

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

$$\frac{}{x \geq 0} \qquad \frac{}{\overline{x} \geq 0}$$

- Addition

$$\text{Addition}\ \frac{x_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3 \qquad \overline{x}_2 + 3x_3 \geq 3}{x_1 + 3\overline{x}_2 + x_3 \geq 4}$$

- Multiplication

$$\text{Multiply by 2}\ \frac{x_1 + 2\overline{x}_2 \geq 3}{2x_1 + 4\overline{x}_2 \geq 6}$$

- Division (and rounding up)

$$\text{Divide by 2}\ \frac{2x_1 + 2\overline{x}_2 + 4x_3 \geq 5}{x_1 + \overline{x}_2 + 2x_3 \geq 3}$$

Extended Cutting Planes: Reification

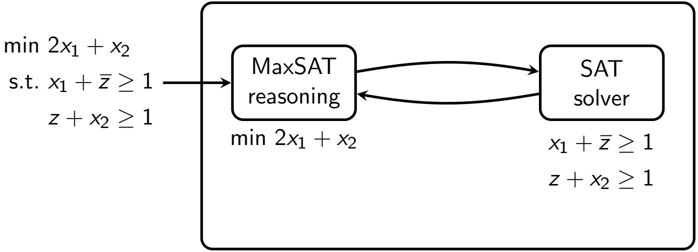
- Constraint C and variable a fresh (not used in proof so far)
- Reification $a \Leftrightarrow C$ (special case of redundance rule in [GN21, BGMN22])

Extended Cutting Planes: Reification

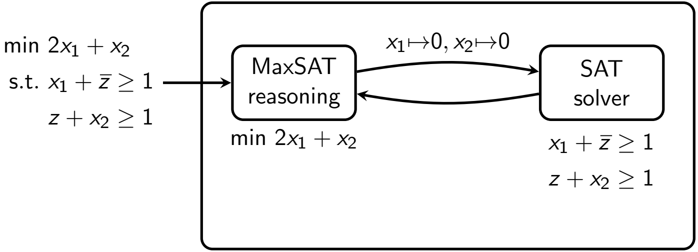
- Constraint C and variable a fresh (not used in proof so far)
- Reification $a \Leftrightarrow C$ (special case of redundance rule in [GN21, BGMN22])

$$a \Leftrightarrow x_1 + \overline{x}_2 + 2x_3 \geq 2 \longrightarrow \begin{array}{ll} 2\overline{a} + x_1 + \overline{x}_2 + 2x_3 \geq 2 & (a \Rightarrow x_1 + \overline{x}_2 + 2x_3 \geq 2) \\ 3a + \overline{x}_1 + x_2 + 2\overline{x}_3 \geq 3 & (a \Leftarrow x_1 + \overline{x}_2 + 2x_3 \geq 2) \end{array}$$

Example: Proof Logging for Core-Guided MaxSAT Solving (1/2)

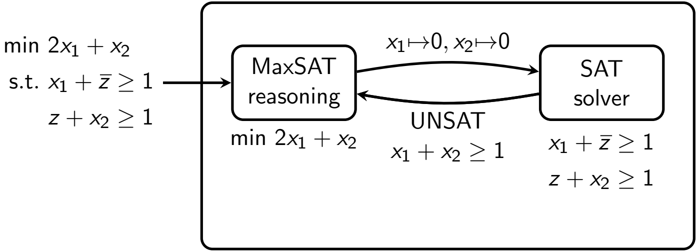


Example: Proof Logging for Core-Guided MaxSAT Solving (1/2)



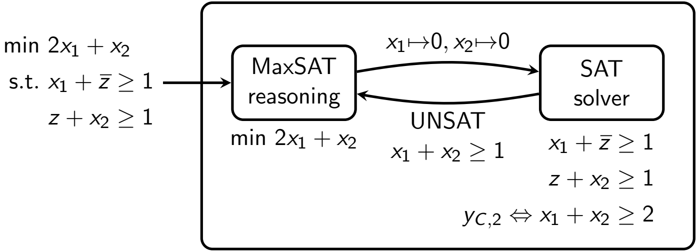
- For SAT solving, translate DRAT proofs [HHW13a, HHW13b, WHH14] to PB

Example: Proof Logging for Core-Guided MaxSAT Solving (1/2)



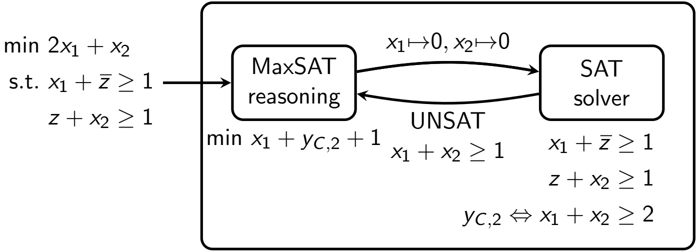
- For SAT solving, translate DRAT proofs [HHW13a, HHW13b, WHH14] to PB
- Core $C : x_1 + x_2 \geq 1$ is implied trivially

Example: Proof Logging for Core-Guided MaxSAT Solving (1/2)



- For SAT solving, translate DRAT proofs [HHW13a, HHW13b, WHH14] to PB
- Core $C : x_1 + x_2 \geq 1$ is implied trivially
- Introduce counter variables definition by reification
- Provide proof logging for totalizers leveraging [GMNO22, VDB22]

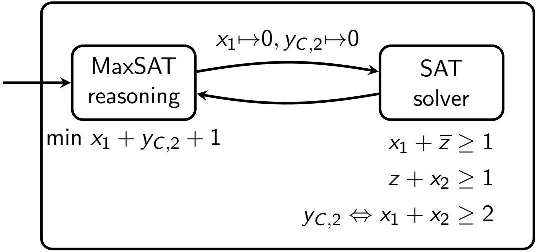
Example: Proof Logging for Core-Guided MaxSAT Solving (1/2)



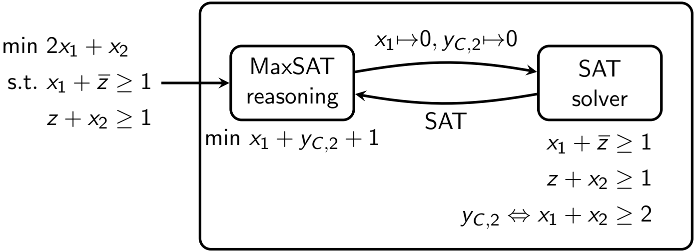
- For SAT solving, translate DRAT proofs [HHW13a, HHW13b, WHH14] to PB
- Core $C : x_1 + x_2 \geq 1$ is implied trivially
- Introduce counter variables definition by reification
- Provide proof logging for totalizers leveraging [GMNO22, VDB22]
- Maintain invariant original objective equal to reformulated objective in proof
- This is $x_1 + x_2 = 1 + y_{C,2}$

Example: Proof Logging for Core-Guided MaxSAT Solving (2/2)

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t. }\ & x_1 + \bar{z} \geq 1 \\ & z + x_2 \geq 1 \end{aligned}$$

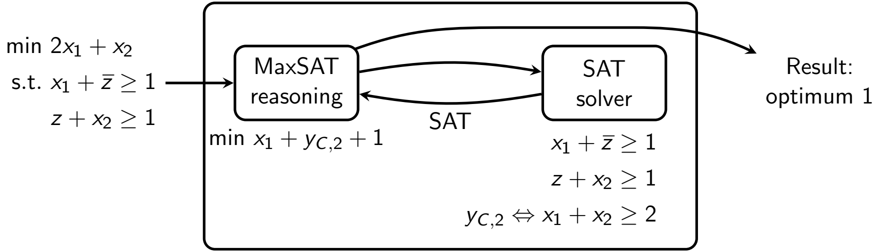


Example: Proof Logging for Core-Guided MaxSAT Solving (2/2)



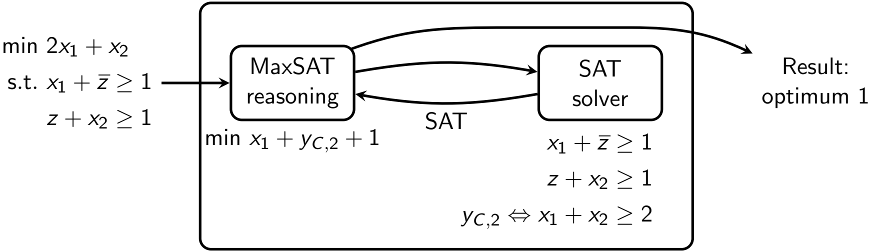
- Solution $(x_1 \mapsto 0, x_2 \mapsto 1, y_{C,2} \mapsto 0, z \mapsto 0)$ is logged in proof

Example: Proof Logging for Core-Guided MaxSAT Solving (2/2)



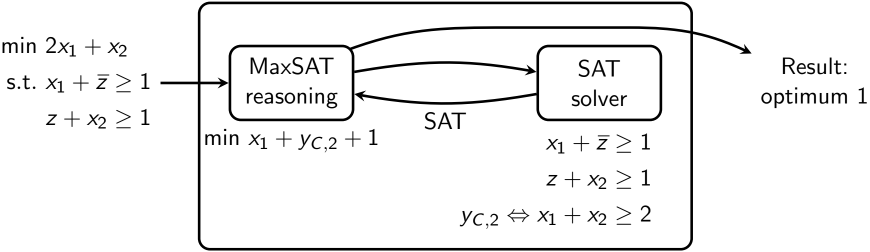
- Solution $(x_1{\mapsto}0, x_2{\mapsto}1, y_{C,2}{\mapsto}0, z{\mapsto}0)$ is logged in proof
- Prove optimality by assuming better solution exists, i.e., $2x_1 + x_2 \leq 0$
- Goal: Derive contradiction from this assumption

Example: Proof Logging for Core-Guided MaxSAT Solving (2/2)



- Solution $(x_1 \mapsto 0, x_2 \mapsto 1, y_{C,2} \mapsto 0, z \mapsto 0)$ is logged in proof
- Prove optimality by assuming better solution exists, i.e., $2x_1 + x_2 \leq 0$
- Goal: Derive contradiction from this assumption
- Add invariant $x_1 + x_2 = 1 + y_{C,2}$ and objective-improving constraint $2x_1 + x_2 \leq 0$

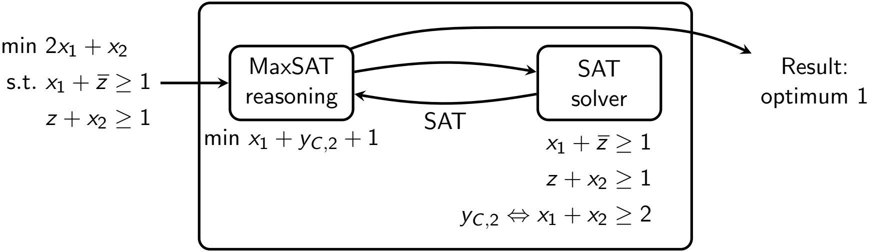
Example: Proof Logging for Core-Guided MaxSAT Solving (2/2)



- Solution $(x_1 \mapsto 0, x_2 \mapsto 1, y_{C,2} \mapsto 0, z \mapsto 0)$ is logged in proof
- Prove optimality by assuming better solution exists, i.e., $2x_1 + x_2 \leq 0$
- Goal: Derive contradiction from this assumption
- Add invariant $x_1 + x_2 = 1 + y_{C,2}$ and objective-improving constraint $2x_1 + x_2 \leq 0$

$$\text{Addition}\ \frac{x_1 + x_2 \geq 1 + y_{C,2} \qquad 0 \geq 2x_1 + x_2}{}$$

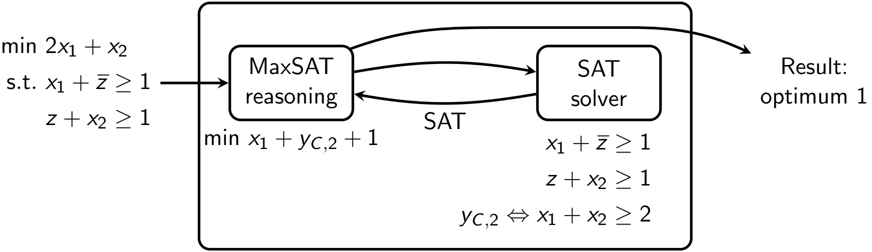
Example: Proof Logging for Core-Guided MaxSAT Solving (2/2)



- Solution $(x_1 \mapsto 0, x_2 \mapsto 1, y_{C,2} \mapsto 0, z \mapsto 0)$ is logged in proof
- Prove optimality by assuming better solution exists, i.e., $2x_1 + x_2 \leq 0$
- Goal: Derive contradiction from this assumption
- Add invariant $x_1 + x_2 = 1 + y_{C,2}$ and objective-improving constraint $2x_1 + x_2 \leq 0$

$$\text{Addition}\ \frac{x_1 + x_2 \geq 1 + y_{C,2} \qquad 0 \geq 2x_1 + x_2}{0 \geq 1 + y_{C,2} + x_1}$$

Example: Proof Logging for Core-Guided MaxSAT Solving (2/2)



- Solution $(x_1 \mapsto 0, x_2 \mapsto 1, y_{C,2} \mapsto 0, z \mapsto 0)$ is logged in proof
- Prove optimality by assuming better solution exists, i.e., $2x_1 + x_2 \leq 0$
- Goal: Derive contradiction from this assumption
- Add invariant $x_1 + x_2 = 1 + y_{C,2}$ and objective-improving constraint $2x_1 + x_2 \leq 0$

$$\text{Addition} \ \frac{x_1 + x_2 \geq 1 + y_{C,2} \qquad 0 \geq 2x_1 + x_2}{0 \geq 1 + y_{C,2} + x_1}$$

- Contradicts assumption
- Solution must be optimal

General Approach: Proof Logging for Core-Guided MaxSAT Solving

1. Record solution in proof to establish upper bound UB
2. Establish lower bound LB

General Approach: Proof Logging for Core-Guided MaxSAT Solving

1. Record solution in proof to establish upper bound UB
2. Establish lower bound LB
   - Prove objective reformulation using invariant $O_{orig} = O_{reform}$ (which contains LB)
     - Show that core clauses are valid
     - Introduce clauses defining counter variables

General Approach: Proof Logging for Core-Guided MaxSAT Solving

1. Record solution in proof to establish upper bound UB
2. Establish lower bound LB
   - Prove objective reformulation using invariant $O_{orig} = O_{reform}$ (which contains LB)
     - Show that core clauses are valid
     - Introduce clauses defining counter variables
3. Obtain proof of optimality if $LB = UB$

General Approach: Proof Logging for Core-Guided MaxSAT Solving

1. Record solution in proof to establish upper bound UB
2. Establish lower bound LB
   - Prove objective reformulation using invariant $O_{orig} = O_{reform}$ (which contains LB)
     - Show that core clauses are valid
     - Introduce clauses defining counter variables
3. Obtain proof of optimality if $LB = UB$

- For anytime solving: Guarantee on lower and upper bound without step 3

Advanced Techniques for Core-Guided MaxSAT

- Important to deal with all state-of-the-art solver techniques
- Additional techniques that are skipped in this talk
  - Intrinsic at-most-one constraints [IMM19]
  - Hardening [ABGL12]
  - Lazy counter variables [MJML14]
- Proof logging also required for these techniques

Advanced Techniques for Core-Guided MaxSAT

- Important to deal with all state-of-the-art solver techniques
- Additional techniques that are skipped in this talk
  - Intrinsic at-most-one constraints [IMM19]
  - Hardening [ABGL12]
  - Lazy counter variables [MJML14]
- Proof logging also required for these techniques
- Very convenient to do in our proof format $\rightarrow$ see our paper

Experimental Evaluation

- Implemented certifying version of state-of-the-art solver CGSS[1] [IBJ21]
- Proof checked with proof checker VeriPB[2]
- Benchmarks from MaxSAT Evaluation 2022[3]
  - 607 unweighted instances and 594 weighted instances

[1] https://gitlab.com/MIAOresearch/software/certified-cgss
[2] https://gitlab.com/MIAOresearch/software/VeriPB
[3] https://maxsat-evaluations.github.io/2022/

Experimental Evaluation

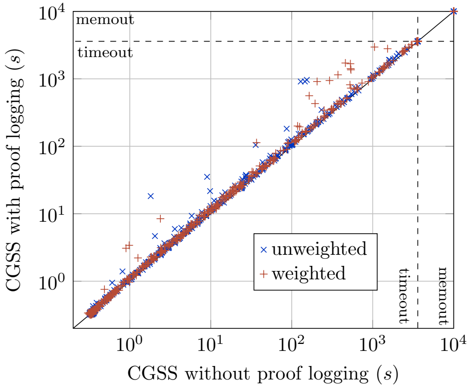
- Implemented certifying version of state-of-the-art solver CGSS[1] [IBJ21]
- Proof checked with proof checker VeriPB[2]
- Benchmarks from MaxSAT Evaluation 2022[3]
  - 607 unweighted instances and 594 weighted instances

First result:

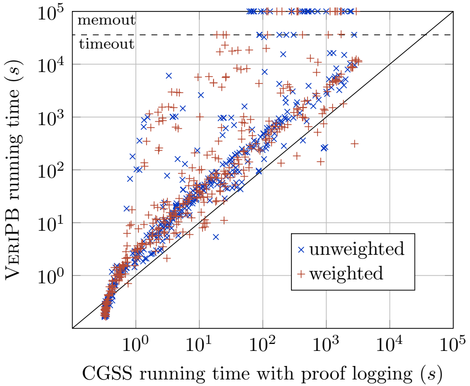
- Discovered bugs in CGSS (and also RC2, on which CGSS is based)
  - All claimed optimal solutions correct for our benchmarks set
  - But solver reasoning sometimes wrong
  - Solver bug could lead to erroneous claims of optimality for other benchmarks

[1] https://gitlab.com/MIAOresearch/software/certified-cgss
[2] https://gitlab.com/MIAOresearch/software/VeriPB
[3] https://maxsat-evaluations.github.io/2022/

Experimental Results



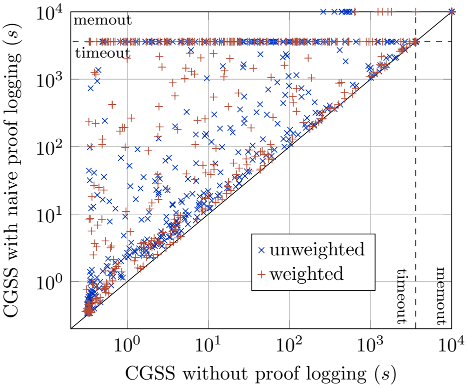
(a) Overhead for proof logging.



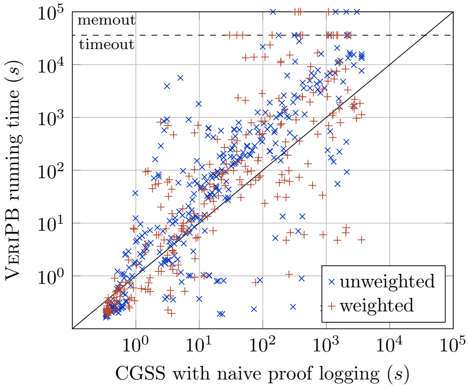
(b) Solving versus checking.

- Low proof logging overhead (8.8% median)
- Checking time could be improved (VeriPB not optimized for SAT solver proofs)

How about Using a SAT Solver to Certify Result?



(a) Overhead for proof logging.



(b) Solving versus checking.

- Encode objective-improving constraint to CNF and solve with SAT solver (Kissat)

Future Work

Further proof logging:

- State-of-the-art linear SAT-UNSAT search solver (like Pacose)
- Implicit hitting set MaxSAT solver
  - Fundamental challenge: proof logging for MIP solver
- Pseudo-Boolean optimization

Future Work

Further proof logging:

- State-of-the-art linear SAT-UNSAT search solver (like Pacose)
- Implicit hitting set MaxSAT solver
  - Fundamental challenge: proof logging for MIP solver
- Pseudo-Boolean optimization

Improving performance and reliability:

- Optimize VeriPB for SAT solver proofs
- Backwards checking/trimming for verification (as in DRAT-trim [HHW13a])
- Formally verified proof checker [BMM+23]

The Sales Pitch For Proof Logging

1. Certifies correctness of computed results
2. Detects errors even if due to compiler bugs, hardware failures, or cosmic rays
3. Debugging support during development [EG21, GMM+20, KM21, BBN+23]
4. Facilitates performance analysis
5. Helps identify potential for further improvements
6. Enables auditability
7. Serves as stepping stone towards explainability

Conclusion

- MaxSAT: successful optimization paradigm, but without proof logging
- Pseudo-Boolean reasoning supports MaxSAT proof logging
- This work: Proof logging for state-of-the-art core-guided MaxSAT solving
- Hopefully step towards general adoption of proof logging for MaxSAT

Conclusion

- MaxSAT: successful optimization paradigm, but without proof logging
- Pseudo-Boolean reasoning supports MaxSAT proof logging
- This work: Proof logging for state-of-the-art core-guided MaxSAT solving
- Hopefully step towards general adoption of proof logging for MaxSAT

Pseudo-Boolean reasoning provides unified proof logging method for:

- SAT solving (including advanced techniques) [GN21, BGMN22]
- Constraint programming [EGMN20, GMN22]
- Graph problems [GMN20, GMM+20]
- SAT-based pseudo-Boolean solving [GMNO22]
- This work: Core-guided MaxSAT solving

Conclusion

- MaxSAT: successful optimization paradigm, but without proof logging
- Pseudo-Boolean reasoning supports MaxSAT proof logging
- This work: Proof logging for state-of-the-art core-guided MaxSAT solving
- Hopefully step towards general adoption of proof logging for MaxSAT

Pseudo-Boolean reasoning provides unified proof logging method for:

- SAT solving (including advanced techniques) [GN21, BGMN22]
- Constraint programming [EGMN20, GMN22]
- Graph problems [GMN20, GMM+20]
- SAT-based pseudo-Boolean solving [GMNO22]
- This work: Core-guided MaxSAT solving
- And soon(?): MaxSAT solving in general
- Further on: MIP solving, planning, other combinatorial problems

Conclusion

- MaxSAT: successful optimization paradigm, but without proof logging
- Pseudo-Boolean reasoning supports MaxSAT proof logging
- This work: Proof logging for state-of-the-art core-guided MaxSAT solving
- Hopefully step towards general adoption of proof logging for MaxSAT

Pseudo-Boolean reasoning provides unified proof logging method for:

- SAT solving (including advanced techniques) [GN21, BGMN22]
- Constraint programming [EGMN20, GMN22]
- Graph problems [GMN20, GMM+20]
- SAT-based pseudo-Boolean solving [GMNO22]
- This work: Core-guided MaxSAT solving
- And soon(?): MaxSAT solving in general
- Further on: MIP solving, planning, other combinatorial problems

Thank you for your attention!

References I

[ABGL12] Carlos Ansótegui, María Luisa Bonet, Joel Gabàs, and Jordi Levy.
Improving SAT-based weighted MaxSAT solvers.
In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP '12)*, volume 7514 of *Lecture Notes in Computer Science*, pages 86–101. Springer, October 2012.

[ADR15] Mario Alviano, Carmine Dodaro, and Francesco Ricca.
A maxsat algorithm using cardinality constraints of bounded size.
In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2677–2683. AAAI Press, 2015.

[AG17] Carlos Ansótegui and Joel Gabàs.
WPM3: an (in)complete algorithm for weighted partial maxsat.
Artif. Intell., 250:37–57, 2017.

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

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.

[BMM+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.

[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.

[DB13a] Jessica Davies and Fahiem Bacchus.
Exploiting the power of MIP solvers in MAXSAT.
In *Proceedings of the 16th International Conference on Theory and Applications of Satisfiability Testing (SAT '13)*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer, July 2013.

References III

[DB13b] Jessica Davies and Fahiem Bacchus.
Postponing optimization to speed up MAXSAT solving.
In *CP*, volume 8124 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2013.

[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.

[ES06] Niklas Eén and Niklas Sörensson.
Translating pseudo-Boolean constraints into SAT.
Journal on Satisfiability, Boolean Modeling and Computation, 2(1-4):1–26, March 2006.

References IV

[FM06] Zhaohui Fu and Sharad Malik.
On solving the partial MAX-SAT problem.
In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT '06)*, volume 4121 of *Lecture Notes in Computer Science*, pages 252–265. Springer, August 2006.

[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 V

[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.

References VI

[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.

[IBJ21] Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo.
Refined core relaxation for core-guided maxsat solving.
In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, volume 210 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[IMM19] Alexey Ignatiev, António Morgado, and João P. Marques-Silva.
RC2: an efficient MaxSAT solver.
Journal on Satisfiability, Boolean Modeling and Computation, 11(1):53–64, September 2019.

References VII

[KM21] Sonja Kraiczy and Ciaran McCreesh.
Solving graph homomorphism and subgraph isomorphism problems faster through clique neighbourhood constraints.
In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI '21)*, pages 1396–1402, August 2021.

[LP10] Daniel Le Berre and Anne Parrain.
The Sat4j library, release 2.2.
Journal on Satisfiability, Boolean Modeling and Computation, 7:59–64, July 2010.

[MDM14] António Morgado, Carmine Dodaro, and João P. Marques-Silva.
Core-guided MaxSAT with soft cardinality constraints.
In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP '14)*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, September 2014.

References VIII

[MJML14] Ruben Martins, Saurabh Joshi, Vasco M. Manquinho, and Inês Lynce.
Incremental cardinality constraints for MaxSAT.
In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP '14)*, volume 8656 of *Lecture Notes in Computer Science*, pages 531–548. Springer, September 2014.

[NB14] Nina Narodytska and Fahiem Bacchus.
Maximum satisfiability using core-guided maxsat resolution.
In *AAAI*, pages 2717–2723. AAAI Press, 2014.

[PRB18] Tobias Paxian, Sven Reimer, and Bernd Becker.
Dynamic polynomial watchdog encoding for solving weighted MaxSAT.
In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT '18)*, volume 10929 of *Lecture Notes in Computer Science*, pages 37–53. Springer, July 2018.

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.

[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.