Certifying MIP-based Presolve Reductions for 0-1 Integer Linear Programs

Andy Oertel
Lund University & University of Copenhagen

WHOOPS 2024

May 24, 2024

Based on work together with Ambros Gleixner, Alexander Hoen, and Jakob Nordström
to appear at CPAIOR 2024

0-1 Integer Linear Programming (ILP)

$$\begin{aligned}
\text{min} \quad & 2x_2 + 3x_3 \\
\text{s.t.} \quad & x_1 + x_2 + x_3 \geq 2 \\
& x_2 + x_3 + x_4 \geq 2 \\
& x_1 - 2x_2 - 2x_3 + x_4 \geq 0
\end{aligned}$$

$\longrightarrow$ $\boxed{\text{0-1 ILP Solver}}$ $\longrightarrow$ Result: optimal value 2

- Input: 0-1 integer linear program (or pseudo-Boolean formula)
  - Integer linear objective function and collection of integer linear inequalities/constraints
  - Variables with domain $\{0, 1\}$
- Output:
  - Optimal value of objective subject to satisfying all inequalities

Grand Goal: Mixed-Integer (Linear) Programming (MIP)

$$\begin{aligned} \min \quad & 2.4x_2 + 1.2x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \geq 2.6 \\ & x_1 + 1.7x_2 \leq 1.5 \\ & x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}, x_3 \in \mathbb{R} \end{aligned}$$

$\longrightarrow$ MIP Solver $\longrightarrow$ Result: optimal value 1.92

- Coefficients are real-valued
- Some variables are integer and some are real-valued

Grand Goal: Mixed-Integer (Linear) Programming (MIP)

$$
\begin{aligned}
\text{min} \quad & 2.4x_2 + 1.2x_3 \\
\text{s.t.} \quad & x_1 + x_2 + x_3 \geq 2.6 \\
& x_1 + 1.7x_2 \leq 1.5 \\
& x_1 \geq 0,\, x_2 \geq 0,\, x_3 \geq 0 \\
& x_1, x_2 \in \mathbb{Z},\, x_3 \in \mathbb{R}
\end{aligned}
$$

$\longrightarrow$ $\boxed{\text{MIP Solver}}$ $\longrightarrow$ Result: optimal value 1.92

- Coefficients are real-valued
- Some variables are integer and some are real-valued

Is this relevant?

- Incredibly powerful paradigm
- Used daily to solve real-world problems in logistics, scheduling, ...

Grand Goal: Mixed-Integer (Linear) Programming (MIP)

$$
\begin{aligned}
\min \quad & 2.4x_2 + 1.2x_3 \\
\text{s.t.} \quad & x_1 + x_2 + x_3 \geq 2.6 \\
& x_1 + 1.7x_2 \leq 1.5 \\
& x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \\
& x_1, x_2 \in \mathbb{Z}, x_3 \in \mathbb{R}
\end{aligned}
$$

→ MIP Solver →

Result:
optimal value 1.92

- Coefficients are real-valued
- Some variables are integer and some are real-valued

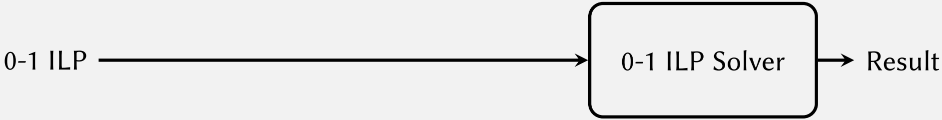
Is this relevant?

- Incredibly powerful paradigm
- Used daily to solve real-world problems in logistics, scheduling, ...

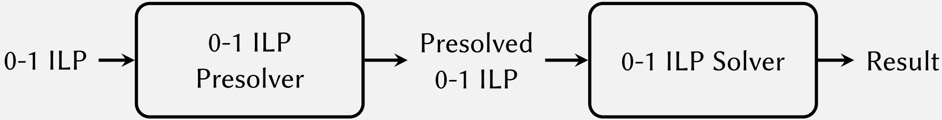
Why study 0-1 ILP?

- 0-1 ILP is very important special case

0-1 ILP (or MIP) Solving in Practice



0-1 ILP (or MIP) Solving in Practice



- Instances are presolved before given to solver
- Presolving is also known as preprocessing in other communities

Importance of Presolving?

- Performance analysis of presolve reductions in MIP [ABG+20]

		default	disabled presolving			
bracket	models	timeout	timeout	faster	slower	times slower
all	3047	547	1035	255	1755	3.36
$\geq$ 0 sec	2511	16	504	255	1755	4.52
$\geq$ 1 sec	1944	16	504	210	1634	6.60
$\geq$ 10 sec	1575	16	504	141	1380	9.05
$\geq$ 100 sec	1099	16	504	86	983	12.36
$\geq$ 1000 sec	692	16	504	34	643	19.48

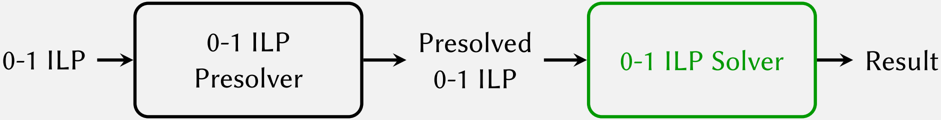
Importance of Presolving?

- Performance analysis of presolve reductions in MIP [ABG+20]

		default	disabled presolving			
bracket	models	timeout	timeout	faster	slower	times slower
all	3047	547	1035	255	1755	3.36
$\geq$ 0 sec	2511	16	504	255	1755	4.52
$\geq$ 1 sec	1944	16	504	210	1634	6.60
$\geq$ 10 sec	1575	16	504	141	1380	9.05
$\geq$ 100 sec	1099	16	504	86	983	12.36
$\geq$ 1000 sec	692	16	504	34	643	19.48

Presolving is one of the most important heuristic in mixed-integer programming!

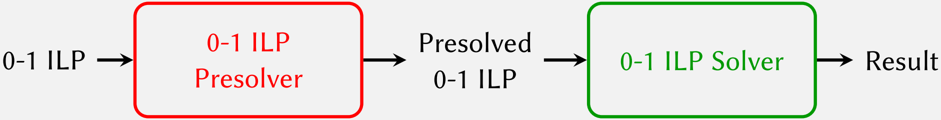
Our Result



Preliminary work:

- Proof logging for branch-and-cut MIP using VIPR [CGS17]

Our Result



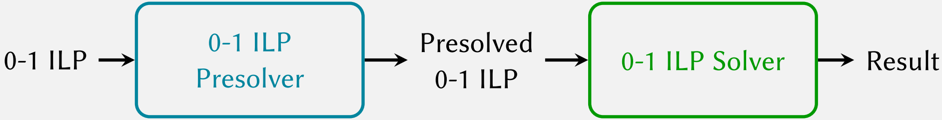
Preliminary work:

- Proof logging for branch-and-cut MIP using VIPR [CGS17]

However...

- VIPR does not extend to presolving

Our Result



Preliminary work:

- Proof logging for branch-and-cut MIP using VIPR [CGS17]

However...

- VIPR does not extend to presolving

Our contribution

- Proof logging for 0-1 ILP presolving
- Proofs verified using VeriPB
- End-to-end certification for state-of-the-art 0-1 ILP solving

Basic Notation

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

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

- Any 0-1 ILP constraint is PB constraint

Basic Notation

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

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

- Any 0-1 ILP constraint is PB constraint
- Equality constraint: syntactic sugar for 2 inequalities

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

Cutting Planes Proof System [CCT87]

- Literal axiom

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

Cutting Planes Proof System [CCT87]

- Literal axiom

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

- Addition

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

Cutting Planes Proof System [CCT87]

- Literal axiom

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

- Addition

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

- Multiplication

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

Cutting Planes Proof System [CCT87]

- 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]

- 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)

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

Strengthening Rules for Cutting Planes (1/2)

- Sometimes we want to add or remove solutions

Strengthening Rules for Cutting Planes (1/2)

- Sometimes we want to add or remove solutions

Redundance-based strengthening:

- Based on [BT19, GN21] and inspired by [JHB12]
- Requires substitution ω (mapping variables to truth values or literals)
- We can introduce C with respect to constraints F and objective f if

$$F \cup \{\neg C\} \vDash \{F \cup C\}\restriction_\omega \cup \{f \geq f\restriction_\omega\}$$

- ω has to be given explicitly
- Implication should be trivial to check

Strengthening Rules for Cutting Planes (2/2)

Strengthening useful for:

- Symmetry breaking
- Without loss of generality reasoning
- Introducing extension variables

Strengthening Rules for Cutting Planes (2/2)

Strengthening useful for:

- Symmetry breaking
- Without loss of generality reasoning
- Introducing extension variables

Additional strengthening rule:

- So-called dominance-based strengthening rule not needed for this talk
- See [BGMN23] for details

Deletion

Problem:

- Deleting constraints arbitrarily is unsound
- Can introduce better than optimal solution
- Deletion needs to be restricted

Deletion

Problem:

- Deleting constraints arbitrarily is unsound
- Can introduce better than optimal solution
- Deletion needs to be restricted

Solution:

- Constraint C can only be deleted if
  - C in derived set
  - C rederivable by redundance-based strengthening from core set without C

Objective Function Update

Effect:

- Changes objective function from f_{old} to f_{new}

Check:

- Equality $f_{old} = f_{new}$ trivially implied by constraints

Objective Function Update

Effect:

- Changes objective function from f_{old} to f_{new}

Check:

- Equality $f_{old} = f_{new}$ trivially implied by constraints

Update specification:

- Give new objective f_{new}
  - Bad for big objectives and small changes
- Give difference between new and old objective $f_{new} - f_{old}$

Why Add an Objective Function Update Rule?

- Naturally represents reasoning in solvers and presolvers
- Substitutions for redundance-based strengthening become complicated to impossible

Why Add an Objective Function Update Rule?

- Naturally represents reasoning in solvers and presolvers
- Substitutions for redundance-based strengthening become complicated to impossible

Example:

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 = 3 \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 = 3 \end{aligned}$$

Why Add an Objective Function Update Rule?

- Naturally represents reasoning in solvers and presolvers
- Substitutions for redundance-based strengthening become complicated to impossible

Example:

$$\begin{array}{ll} \min & x_1 + x_2 \\ \text{s.t.} & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 = 3 \end{array} \longrightarrow \begin{array}{ll} \min & x_3 + x_4 + 1 \\ \text{s.t.} & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 = 3 \end{array}$$

- $x_2 \geq 1$ by redundance-based strengthening with substitution $\{x_2 \mapsto 1\}$

Why Add an Objective Function Update Rule?

- Naturally represents reasoning in solvers and presolvers
- Substitutions for redundance-based strengthening become complicated to impossible

Example:

$$\begin{array}{ll} \min & x_1 + x_2 \\ \text{s.t.} & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 = 3 \end{array} \longrightarrow \begin{array}{ll} \min & x_3 + x_4 + 1 \\ \text{s.t.} & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 = 3 \end{array}$$

- $x_2 \geq 1$ by redundance-based strengthening with substitution $\{x_2 \mapsto 1\}$
- If objective unchanged, then $x_1 + x_2 \geq x_1 + 1$ has to be shown in subproof
- But this is not required if objective is updated

In general: Certifying Presolving

How to certify presolving?

- Presolving can and will change solution space
- Soundness of proof system guarantees that optimal value does not change
- Check that derived 0-1 ILP in proof is equivalent to presolved 0-1 ILP

In general: Certifying Presolving

How to certify presolving?

- Presolving can and will change solution space
- Soundness of proof system guarantees that optimal value does not change
- Check that derived 0-1 ILP in proof is equivalent to presolved 0-1 ILP

Guarantee:

- Original 0-1 ILP has same optimal value as presolved 0-1 ILP
- Except for logged solutions (especially optimal solutions)

Example: Probing

$$
\begin{aligned}
\min \quad & 1x_1 + 2x_2 + 3x_3 \\
\text{s.t.} \quad & x_1 + x_2 \geq 1 \\
& \bar{x}_1 + \bar{x}_2 + x_3 \geq 2 \\
& x_1 + x_2 + \bar{x}_3 + \bar{x}_4 + \bar{x}_5 \geq 4 \\
& \bar{x}_1 + \bar{x}_2 + x_3 + x_4 + x_5 \geq 1
\end{aligned}
$$

- Detect that $x_3 = 1$ by unit propagation

Example: Probing

$$\begin{array}{ll} \min & 1x_1 + 2x_2 + 3x_3 \\ \text{s.t.} & x_1 + x_2 \geq 1 \\ & \bar{x}_1 + \bar{x}_2 + x_3 \geq 2 \\ & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 + \bar{x}_5 \geq 4 \\ & \bar{x}_1 + \bar{x}_2 + x_3 + x_4 + x_5 \geq 1 \end{array} \longrightarrow \begin{array}{ll} \min & 1x_1 + 2x_2 + 3x_3 \\ \text{s.t.} & x_1 + x_2 \geq 1 \\ & \bar{x}_1 + \bar{x}_2 \geq 1 \\ & x_1 + x_2 + \bar{x}_4 + \bar{x}_5 \geq 4 \\ & \bar{x}_1 + \bar{x}_2 + x_4 + x_5 \geq 0 \\ & x_3 \geq 1 \end{array}$$

- Detect that $x_3 = 1$ by unit propagation

Certification:

- Add $x_3 \geq 1$ by reverse unit propagation
- Use addition (and literal axiom) to eliminate x_3 in all constraints

Example: Objective Function Update

$$
\begin{aligned}
\min \quad & 1x_1 + 2x_2 + 3x_3 \\
\text{s.t.} \quad & x_1 + x_2 \geq 1 \\
& \bar{x}_1 + \bar{x}_2 \geq 1 \\
& x_1 + x_2 + \bar{x}_4 + \bar{x}_5 \geq 4 \\
& \bar{x}_1 + \bar{x}_2 + x_4 + x_5 \geq 0 \\
& x_3 \geq 1
\end{aligned}
$$

- As $x_3 = 1$, we can set x_3 to 1 in the objective
- $x_3 \geq 1$ can be removed from the constraints

Example: Objective Function Update

$$
\begin{array}{ll}
\min & 1x_1 + 2x_2 + 3x_3 \\
\text{s.t.} & x_1 + x_2 \geq 1 \\
& \bar{x}_1 + \bar{x}_2 \geq 1 \\
& x_1 + x_2 + \bar{x}_4 + \bar{x}_5 \geq 4 \\
& \bar{x}_1 + \bar{x}_2 + x_4 + x_5 \geq 0 \\
& x_3 \geq 1
\end{array}
\longrightarrow
\begin{array}{ll}
\min & 1x_1 + 2x_2 + 3 \\
\text{s.t.} & x_1 + x_2 \geq 1 \\
& \bar{x}_1 + \bar{x}_2 \geq 1 \\
& x_1 + x_2 + \bar{x}_4 + \bar{x}_5 \geq 4 \\
& \bar{x}_1 + \bar{x}_2 + x_4 + x_5 \geq 0
\end{array}
$$

- As $x_3 = 1$, we can set x_3 to 1 in the objective
- $x_3 \geq 1$ can be removed from the constraints

Certification:

- Objective update rule checking if $1x_1 + 2x_2 + 3x_3 = 1x_1 + 2x_2 + 3$ implied
- Deletion of $x_3 \geq 1$ justified by substitution $\{x_3 \mapsto 1\}$

Example: Dominated Variable

$$
\begin{aligned}
\min \quad & 1x_1 + 2x_2 + 3 \\
\text{s.t.} \quad & x_1 + x_2 \geq 1 \\
& \bar{x}_1 + \bar{x}_2 \geq 1 \\
& x_1 + x_2 + \bar{x}_4 + \bar{x}_5 \geq 4 \\
& \bar{x}_1 + \bar{x}_2 + x_4 + x_5 \geq 0
\end{aligned}
$$

- W.l.o.g. $x_1 \geq x_2$, as
  - Coefficient of x_1 is at least coefficient of x_2 in all constraints
  - Coefficient of x_1 is at most coefficient of x_2 in the objective

Example: Dominated Variable

$$
\begin{aligned}
\min \quad & 1x_1 + 2x_2 + 3 \\
\text{s.t.} \quad & x_1 + x_2 \geq 1 \\
& \bar{x}_1 + \bar{x}_2 \geq 1 \\
& x_1 + x_2 + \bar{x}_4 + \bar{x}_5 \geq 4 \\
& \bar{x}_1 + \bar{x}_2 + x_4 + x_5 \geq 0
\end{aligned}
\longrightarrow
\begin{aligned}
\min \quad & 1x_1 + 2x_2 + 3 \\
\text{s.t.} \quad & x_1 + x_2 \geq 1 \\
& \bar{x}_1 + \bar{x}_2 \geq 1 \\
& x_1 + x_2 + \bar{x}_4 + \bar{x}_5 \geq 4 \\
& \bar{x}_1 + \bar{x}_2 + x_4 + x_5 \geq 0 \\
& x_1 + \bar{x}_2 \geq 1
\end{aligned}
$$

- W.l.o.g. $x_1 \geq x_2$, as
  - Coefficient of x_1 is at least coefficient of x_2 in all constraints
  - Coefficient of x_1 is at most coefficient of x_2 in the objective

Certification:

- Add $x_1 + \bar{x}_2 \geq 1$ by redundance-based strengthening using $\{x_1 \mapsto x_2, x_2 \mapsto x_1\}$

Experimental Setup

Tools:

- Added pseudo-Boolean proof logging to ILP presolver PaPILO[1]
- Proof checked using proof checker VeriPB[2]

[1] https://github.com/scipopt/papilo

[2] https://gitlab.com/MIAOresearch/software/VeriPB

Experimental Setup

Tools:

- Added pseudo-Boolean proof logging to ILP presolver PaPILO[1]
- Proof checked using proof checker VeriPB[2]

Benchmarks:

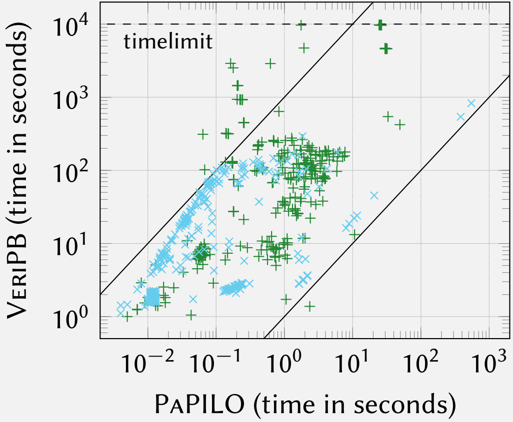
- PB competition 2016 instances [Pse16]
- MIPLIB17 instances translated to OPB format [Dev20]

[1] https://github.com/scipopt/papilo
[2] https://gitlab.com/MIAOresearch/software/VeriPB

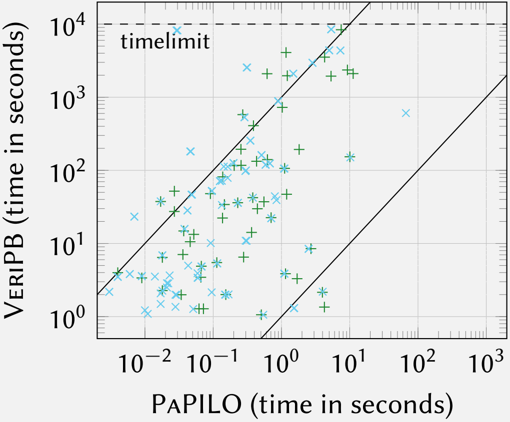
Proof Logging Overhead in PaPILO

Test set	size	default	w/proof log	relative
PB16-dec	1398	0.050	0.077	1.54
MIPLIB01-dec	295	0.498	0.631	1.27
PB16-opt	532	0.439	0.565	1.29
MIPLIB01-opt	144	0.337	0.473	1.40

Certificate Checking Performance (1/2)



(a) PaPILO vs. VeriPB on PB16 instances.



(b) PaPILO vs. VeriPB on MIPLIB01 instances.

Certificate Checking Performance (2/2)

			PaPILO time (in s)		VeriPB	relative time w.r.t.	
Test set	size	verified	w/proof log	default	time (in s)	w/proof log	default
PB16-dec	1398	1398	0.076	0.050	1.28	16.81	25.54
MIPLIB01-dec	293	261	0.55	0.42	17.36	31.78	41.37
PB16-opt	531	520	0.78	0.44	16.17	20.74	36.75
MIPLIB01-opt	140	133	1.38	0.27	10.40	7.53	38.32

Certificate Checking Performance (2/2)

			PaPILO time (in s)		VeriPB	relative time w.r.t.	
Test set	size	verified	w/proof log	default	time (in s)	w/proof log	default
PB16-dec	1398	1398	0.076	0.050	1.28	16.81	25.54
MIPLIB01-dec	293	261	0.55	0.42	17.36	31.78	41.37
PB16-opt	531	520	0.78	0.44	16.17	20.74	36.75
MIPLIB01-opt	140	133	1.38	0.27	10.40	7.53	38.32

- Most instances verified within 10 000s timeout
- Overhead can be explained by PaPILO having more context than VeriPB
- PaPILO parallelizes some tasks, VeriPB works only sequentially

RUP vs. Cutting Planes

RUP:

- Just claim that constraint is implied, which is checked by unit propagation
- Shorthand for “simple” cutting planes derivation

RUP vs. Cutting Planes

RUP:

- Just claim that constraint is implied, which is checked by unit propagation
- Shorthand for “simple” cutting planes derivation

For instances with at least 10 propagation reductions:

		RUP		cutting planes		
test set	size	verified	time [s]	verified	time [s]	relative
PB-dec	284	284	2.21	284	2.14	0.968
MIPLIB-dec	35	31	153.23	31	148.88	0.972
PB-opt	153	142	28.43	142	28.22	0.993
MIPLIB-opt	16	14	147.11	14	127.83	0.869

Conclusion & Future Work

- 0-1 ILP seems like a good first step towards proof logging for MIP
- Presolving is an integral part to MIP solving
- Our approach provides proof logging for
  - 0-1 ILP presolving

Conclusion & Future Work

- 0-1 ILP seems like a good first step towards proof logging for MIP
- Presolving is an integral part to MIP solving
- Our approach provides proof logging for
  - 0-1 ILP presolving
  - SAT solving (including advanced techniques) [GN21, BGMN23]
  - MaxSAT solving [VDB22, BBN+23]
  - Constraint Programming [EGMN20, GMN22, MM23]
  - Subgraph problems [GMN20, GMM+20, GMM+24]

Conclusion & Future Work

- 0-1 ILP seems like a good first step towards proof logging for MIP
- Presolving is an integral part to MIP solving
- Our approach provides proof logging for
  - 0-1 ILP presolving
  - SAT solving (including advanced techniques) [GN21, BGMN23]
  - MaxSAT solving [VDB22, BBN+23]
  - Constraint Programming [EGMN20, GMN22, MM23]
  - Subgraph problems [GMN20, GMM+20, GMM+24]

Future research directions:

- Compare RUP/cutting planes approach with new annotated RUP
- Planning, MIP [DEGH23], dynamic programming, and other combinatorial problems
- Generalize our approach to enumeration and counting problems

Conclusion & Future Work

- 0-1 ILP seems like a good first step towards proof logging for MIP
- Presolving is an integral part to MIP solving
- Our approach provides proof logging for
  - 0-1 ILP presolving
  - SAT solving (including advanced techniques) [GN21, BGMN23]
  - MaxSAT solving [VDB22, BBN+23]
  - Constraint Programming [EGMN20, GMN22, MM23]
  - Subgraph problems [GMN20, GMM+20, GMM+24]

Future research directions:

- Compare RUP/cutting planes approach with new annotated RUP
- Planning, MIP [DEGH23], dynamic programming, and other combinatorial problems
- Generalize our approach to enumeration and counting problems

Thank you for your attention!

References I

[ABG+20] Tobias Achterberg, Robert E. Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger.
Presolve reductions in mixed integer programming.
INFORMS Journal on Computing, 32(2):473–506, 2020.

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

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

[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.

References II

[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.

[DEGH23] Jasper van Doornmalen, Leon Eifler, Ambros Gleixner, and Christopher Hojny.
A proof system for certifying symmetry and optimality reasoning in integer programming.
Technical Report 2311.03877, arXiv.org, November 2023.

[Dev20] Jo Devriendt.
Miplib 0-1 instances in opb format, May 2020.

[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.

References III

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

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

[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 IV

[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.

[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.

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

[Pse16] Pseudo-Boolean competition 2016.
`https://www.cril.univ-artois.fr/PB16/`, July 2016.

[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.

References V