# PaPILO: A parallel presolving library for MIP and LP with multi-precision support

Ambros Gleixner and Leona Gottwald

Zuse Institute Berlin

May 20, 2020

# Outline of the talk

## Introduction
- Mixed Integer Programming
- Role of presolving
- Motivations for a new framework

## Introduction to IP presolving techniques
- Classification of presolvers
- IP presolvers in PaPILO

## PaPILO
- Workflow of PaPILO
- Difficulties parallelising presolvers

## Computational experiments
- Floating-point presolving
- Rational presolving

# Mixed Integer Programming

Concerned with the class of optimization problems of the form

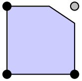
$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\quad & Ax \le b \\ & \ell \le x \le u \\ & x_i \in \mathbb{Z}\ \forall i \in I \end{aligned}$$

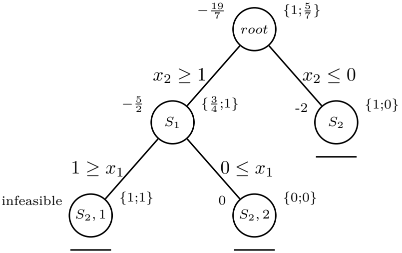
with $A \in \mathbb{R}^{m\times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m, u \in (\mathbb{R} \cup \{\infty\})^n, l \in (\mathbb{R} \cup \{-\infty\})^n$,
variables $x \in \mathbb{R}^n$ with $j \in I \subset N = \{1, \dots, n\}$
for every row $i \in M = \{1, \dots, m\}$.

Most successful general solving paradigm: LP-based branch-and-bound.

# A small example

$$\min -2x_1 - x_2$$
$$7x_1 + 8x_2 \leq 13$$
$$x \in \{0,1\}^2$$

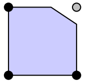




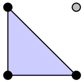
# A small example

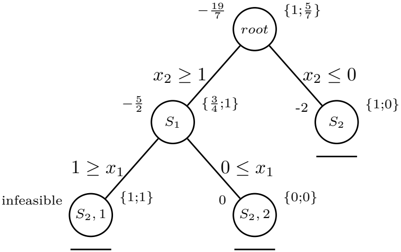
Before and after coefficient strengthening:

$$\min -2x_1 - x_2$$
$$7x_1 + 8x_2 \leq 13$$
$$x \in \{0,1\}^2$$



$$\min -2x_1 - x_2$$
$$x_1 + x_2 \leq 1$$
$$x \in \{0,1\}^2$$

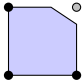




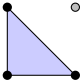
# A small example

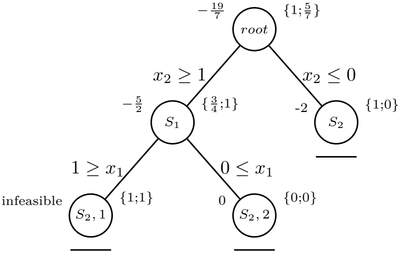
Before and after coefficient strengthening:

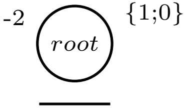
$$\min -2x_1 - x_2$$
$$7x_1 + 8x_2 \leq 13$$
$$x \in \{0,1\}^2$$



$$\min -2x_1 - x_2$$
$$x_1 + x_2 \leq 1$$
$$x \in \{0,1\}^2$$





# Goals of MIP presolving

MIP presolving has several goals:

- tighten the formulation, i.e., the LP relaxation
- reduce size: remove “redundant” variables, constraints, nonzeros
- collect global information: conflicts/cliques, implications
- identify structure, e.g., special constraint types or symmetry
- improve numerics

This is not only possible when the formulation contains trivial redundancies: Redundancies or special structures are sometimes created during presolving.

# Performance impact of presolving

Presolving is one of the components with largest impact on performance:

Table 1: Impact of disabling presolve

| | | default | disable presolve | | | | | affected | |
|---|---|---|---|---|---|---|---|---|---|
| bracket | models | tilim | tilim | faster | slower | time | nodes | models | time |
| all | 3047 | 547 | 1035 | 255 | 1755 | 3.36 | 1.27 | — | — |
| $\geq$ 0 sec | 2511 | 16 | 504 | 255 | 1755 | 4.52 | 1.91 | 2411 | 4.80 |
| $\geq$ 1 sec | 1944 | 16 | 504 | 210 | 1634 | 6.60 | 2.12 | 1929 | 6.73 |
| $\geq$ 10 sec | 1575 | 16 | 504 | 141 | 1380 | 9.05 | 2.29 | 1564 | 9.23 |
| $\geq$ 100 sec | 1099 | 16 | 504 | 86 | 983 | 12.36 | 2.43 | 1095 | 12.50 |
| $\geq$ 1000 sec | 692 | 16 | 504 | 34 | 643 | 19.48 | 2.17 | 691 | 19.57 |

[Achterberg, et al. „Presolve Reductions in Mixed Integer Programming.“, INFORMS Journal on Computing, vol 32, pp 473-506, 2020]

# PaPILO: a new presolving framework

The main motivation for a new framework comes from current limitations of the existing implementations:

- solver-specific
- do not exploit parallelism
- bound to floating-point arithmetic

# PaPILO: a new presolving framework

The main motivation for a new framework comes from current limitations of the existing implementations:

- solver-specific
- do not exploit parallelism
- bound to floating-point arithmetic

$\rightsquigarrow$ PaPILO: **Pa**rallel **P**resolve for **I**nteger and **L**inear **O**ptimization provides

- solver-independent presolving
- a new parallelization scheme
- templatized arithmetic

# Outline of the talk

# Types of reductions

## Primal reductions

- based on feasibility reasoning
- no feasible solution is cut off

## Dual reductions

- based on optimality reasoning
- weak dual reduction: no optimal solution is cut off
- strong dual reduction: at least one optimal solution remains

# Types of reductions

## Primal reductions

- based on feasibility reasoning
- no feasible solution is cut off
  - e.g. $0 \leq x, y \leq 3;\ x + y \leq 2 \rightarrow x, y \leq 2$

## Dual reductions

- based on optimality reasoning
- weak dual reduction: no optimal solution is cut off

- strong dual reduction: at least one optimal solution remains

# Types of reductions

## Primal reductions

- based on feasibility reasoning
- no feasible solution is cut off
  - e.g. $0 \leq x, y \leq 3;\ x + y \leq 2 \rightarrow x, y \leq 2$

## Dual reductions

- based on optimality reasoning
- weak dual reduction: no optimal solution is cut off
  - e.g. $\min x \ s.t.\ x \geq 0 \ \rightarrow x = 0$
- strong dual reduction: at least one optimal solution remains

# Types of reductions

## Primal reductions

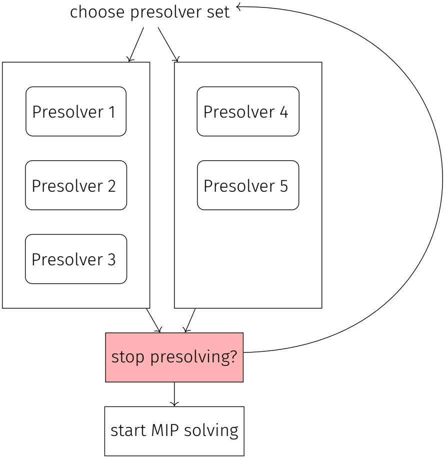
- based on feasibility reasoning
- no feasible solution is cut off
  - e.g. $0 \leq x, y \leq 3;\ x + y \leq 2 \to x, y \leq 2$

## Dual reductions

- based on optimality reasoning
- weak dual reduction: no optimal solution is cut off
  - e.g. $\min x$ $s.t.$ $x \geq 0 \ \to x = 0$
- strong dual reduction: at least one optimal solution remains
  - e.g. $\min y$ $s.t.$ $x, y \geq 0 \ \to x = 0$

# Organization in rounds

Reductions of one presolver can enable further reductions by other presolvers $\rightsquigarrow$ iterative procedure until reductions stall:



# Complexity of presolvers

In PaPILO we attempt a formal specification of computational complexity:

**Fast:** $\mathcal{O}(n \log n)$ where $n =$ changed number of nonzeros since last call

- e.g. Coefficient/bound Strengthening

**Medium:** $\mathcal{O}(N \log N)$ where $N =$ number of nonzeros

- e.g. Dual fix, Simple probing

**Exhaustive:** $\mathcal{O}(N^2)$ where $N =$ number of nonzeros

- e.g. Probing, Dominated columns

# Presolvers in PaPILO

- Singleton columns/row
- Coefficient Strengthening
- Bound Strengthening
- Simple Probing
- Dualfix
- Detection of parallel rows and columns
- Substitution of implied free variables with special treatment for singleton columns and doubleton equations
- Simplify Inequalities
- Primal and dual implied integer detection
- Exploitation of complementary slackness
- Probing
- Dominated Columns
- Removal of redundant penalty variables
- Removal of linear dependent equations

# A fast presolver: bound strengthening

## Example

$$\begin{aligned} x_1, x_2, x_3 &\in \{0, 1, 2, 3, 4\} \\ 4x_1 - 3x_2 + 5x_3 &\leq 2 \end{aligned}$$

$$\Rightarrow x_3 \leq \frac{2 - (4x_1 - 3x_2)}{5}$$

# A fast presolver: bound strengthening

## Example

$$x_1, x_2, x_3 \in \{0, 1, 2, 3, 4\}$$
$$4x_1 - 3x_2 + 5x_3 \leq 2$$

$$\Rightarrow x_3 \leq \frac{2 - (4x_1 - 3x_2)}{5} \leq \frac{2 - \alpha_{min}^{1,2}}{5}$$

# A fast presolver: bound strengthening

## Example

$$x_1, x_2, x_3 \in \{0, 1, 2, 3, 4\}$$
$$4x_1 - 3x_2 + 5x_3 \leq 2$$

$$\Rightarrow x_3 \leq \frac{2 - (4x_1 - 3x_2)}{5} \leq \frac{2 - \alpha_{min}^{1,2}}{5} \leq \frac{2 + 12}{5} = 2.8$$
$$\rightarrow x_3 \leq 2$$

## General

- if $a_{ik} > 0$ then we derive a new upper bound $u_k = \min\{u_k, \lfloor \frac{b_i - \alpha_{min}^S}{a_{ik}} \rfloor\}$
- if $a_{ik} < 0$ then we derive a new lower bound $l_k = \max\{l_k, \lceil \frac{b_i - \alpha_{min}^S}{a_{ik}} \rceil\}$

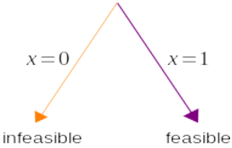
# An exhaustive presolver: probing

Probing is an important presolve step for MILP problems with binary variables

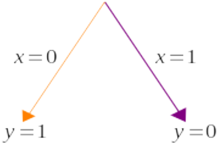
Consecutively fix a binary variable to 0 and 1 and inspect consequences from propagating those bound changes.

## Possible reductions

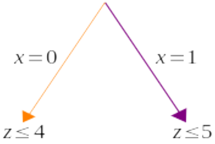
- If one branch is infeasible the variable can be fixed to the other branch
- If another variable is fixed to different values in both branches it can be substituted
- Bounds of variables can be tightened to the weakest of both branches




Fix $x=1$




Substitute $y=1-x$



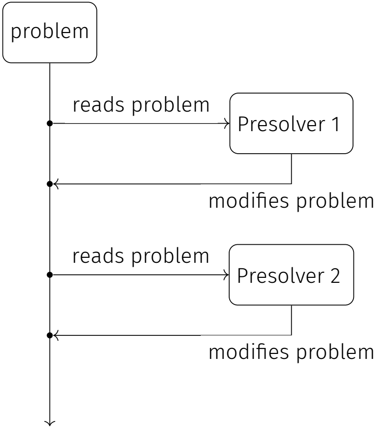

$z\leq 5$ valid globally

# Outline of the talk

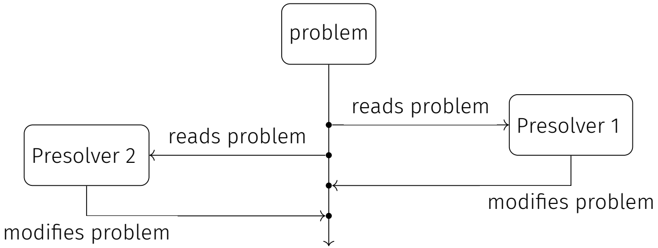
# What is PAPILO?

- **Pa**rallel **P**resolve for **I**nteger and **L**inear **O**ptimization
- C++14 based software package
- Provides presolve and postsolve routines for MILP problems
- New addition to the SCIP Optimization Suite 7 [Gamrath et al. 2020]
- Additionally available: `https://github.com/scipopt` (*coming soon*)
- Supports versatile use-cases
  - Frontend for solvers like SCIP, SoPlex, HiGHS
  - File-based presolve and postsolve for MPS files
  - Header-only library
  - As SCIP presolver plugin

# Sequential presolving



# Difficulties for parallel presolving

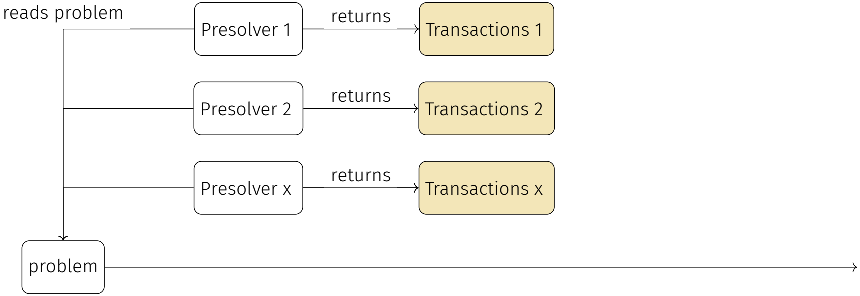



Problem might have changed!
Applying reductions might cause illegal states!

## General difficulties

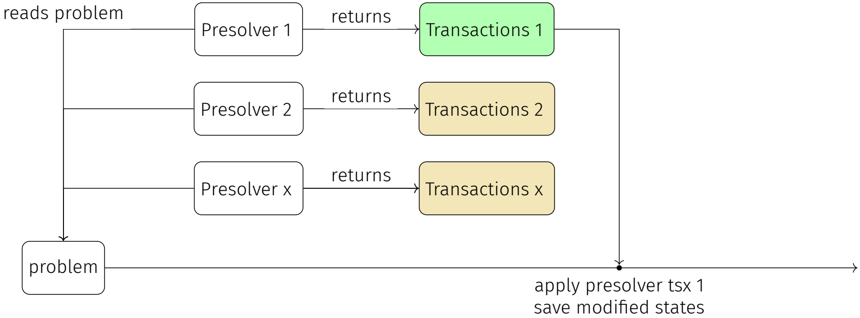
- individual presolving steps are usually fast
- regular synchronization may limit scalability
- ensuring deterministic behavior may limit scalability

# Transaction-based design



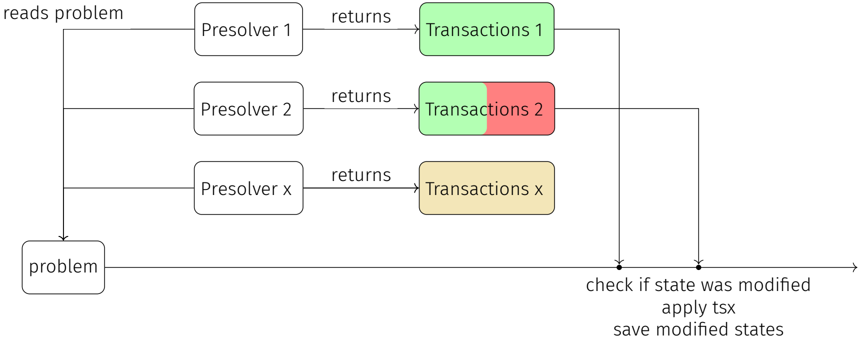

Transaction = reduction + data dependencies

# Transaction-based design



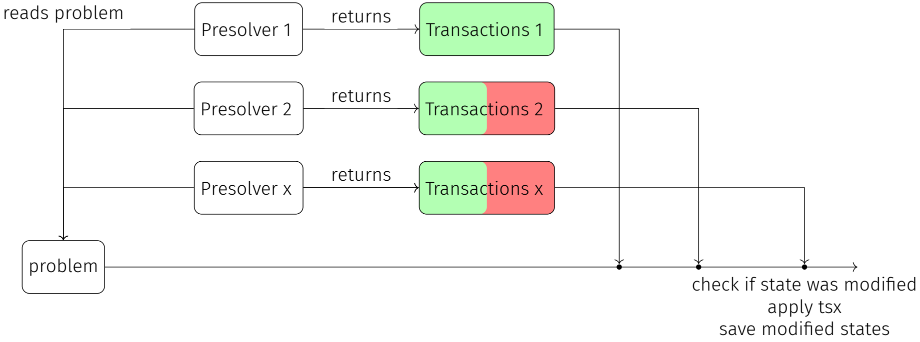

Transaction = reduction + data dependencies

# Transaction-based design




Transaction = reduction + data dependencies

# Transaction-based design




Transaction $=$ reduction $+$ data dependencies

# Conflict detection and resolution

How does PaPILO detect and resolve conflicts?

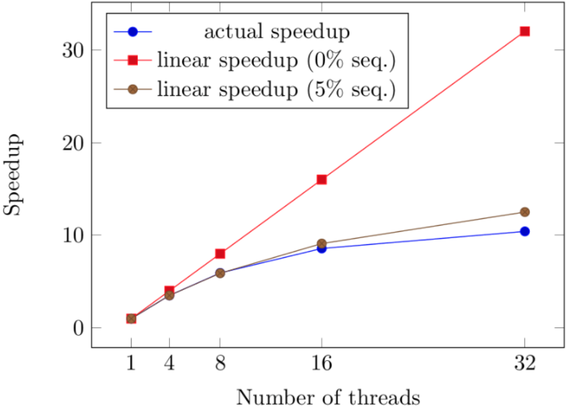
- Obtains a set of transactions by calling presolvers
- Transactions are applied in some sequential order
- When rows or columns are modified their state is recorded as such
- If a lock of a transaction conflicts with the state of rows/columns discard it

Bound changes on the same column can be resolved by keeping the tightest bounds.

# Trade-off: External vs. internal parallelism

## Internal parallelization of probing

Speedup of ex10 using SCIP 7.0 with PaPILO for different numbers of threads. Additionally show ideal linear speedup curves when 0% and 5% of the time are spent inside sequential code.



# Outline of the talk

# Impact on MIPLIB 2017

| | | PaPILO 1.2.0+SCIP 7.0.2.4 | | | SCIP 7.0.2.4 wo presolving | | | relative | |
|---|---|---|---|---|---|---|---|---|---|
| Subset | instances | solved | time | nodes | solved | time | nodes | time | nodes |
| all | 233 | 112 | 753.4 | 3942 | 92 | 1299.3 | 4044 | 1.72 | 1.03 |
| [0,tilim] | 117 | 112 | 159.2 | 2129 | 92 | 473.5 | 3580 | 2.97 | 1.68 |
| [1,tilim] | 116 | 111 | 165.8 | 2189 | 91 | 497.7 | 3695 | 3.00 | 1.69 |
| [10,tilim] | 110 | 105 | 202.3 | 2517 | 85 | 665.3 | 4456 | 3.29 | 1.77 |
| [100,tilim] | 90 | 85 | 338.7 | 4020 | 65 | 1220.9 | 7320 | 3.61 | 1.82 |
| [1000,tilim] | 60 | 55 | 510.5 | 4843 | 35 | 2388.8 | 10260 | 4.68 | 2.12 |
| affected | 117 | 112 | 159.2 | 2129 | 92 | 473.5 | 3580 | 2.97 | 1.68 |

# Exact SCIP with and without rational presolving

| | | presolving disabled | | | presolving enabled | | | |
|---|---|---|---|---|---|---|---|---|
| Test set | size | solved | time | nodes | solved | time | (presolving) | nodes |
| FPEASY | 168 | 165 | 42.1 | 6145.3 | 168 | 25.5 | (0.22) | 4724.1 |
| NUMDIFF | 91 | 66 | 216.6 | 7237.2 | 86 | 58.7 | (1.23) | 2867.2 |

# Comparison of exact and floating-point presolving

| | | floating-point presolving | | | | | exact presolving | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Test set | thrds | time | rnds | fixed | agg | bdchg | time | rnds | fixed | agg | bdchg |
| FPEASY | 1 | 0.01 | 3.2 | 8.5 | 3.5 | 10.4 | 0.25 | 3.2 | 8.5 | 3.5 | 10.4 |
| | 20 | 0.01 | 3.2 | 8.5 | 3.5 | 10.4 | 0.14 | 3.2 | 8.5 | 3.5 | 10.4 |
| NUMDIFF | 1 | 0.04 | 8.3 | 53.8 | 55.7 | 51.4 | 0.89 | 7.2 | 41.4 | 42.9 | 55.8 |
| | 20 | 0.04 | 8.3 | 53.8 | 55.7 | 51.4 | 0.50 | 7.2 | 41.4 | 42.9 | 55.8 |

# Take-away message

PaPILO is the first

- parallel
- multi-precision
- solver-independent

library for presolving MIPs and LPs.

Not discussed

- presolving techniques
- runtime scheduling of parallel tasks (exploit TBB)
- detailed computational analysis (ongoing)
- ...