Tutorial on Mixed Integer Linear Programming (MIP) and Pseudo-Boolean Optimization

Jakob Nordström

University of Copenhagen and Lund University

1st International Workshop on Solving Linear Optimization Problems for Pseudo-Booleans and Yonder
Lund, Sweden
November 5, 2024

Outline of Tutorial on MIP Solving and PB Optimization

An Acknowledgement and an Apology

The MIP material relies heavily on the presentation *Computational Mixed-Integer Programming* by Ambros Gleixner at the Casa Matemática Oaxaca (CMO) workshop *Theory and Practice of Satisfiability Solving* in 2018 (`https://tinyurl.com/MIPtutorial`)

An Acknowledgement and an Apology

The MIP material relies heavily on the presentation *Computational Mixed-Integer Programming* by Ambros Gleixner at the Casa Matemática Oaxaca (CMO) workshop *Theory and Practice of Satisfiability Solving* in 2018 (`https://tinyurl.com/MIPtutorial`)

A bit too many references are still missing — see Gleixner's slides for full details

An Acknowledgement and an Apology

The MIP material relies heavily on the presentation *Computational Mixed-Integer Programming* by Ambros Gleixner at the Casa Matemática Oaxaca (CMO) workshop *Theory and Practice of Satisfiability Solving* in 2018 (`https://tinyurl.com/MIPtutorial`)

A bit too many references are still missing — see Gleixner's slides for full details

Another excellent source of resources are the webpages from the summer school *Computational Optimization at Work* in 2024 (`https://co-at-work.zib.de`)

Mixed Integer Linear Programming

Mixed integer linear program

- Minimize $\sum_j a_j x_j$
- Subject to $\sum_j a_{i,j} x_j \leq A_i$, $i = 1, \ldots, m$
- $x_j \in \mathbb{N}$ for $j = 1, \ldots, n$
- $x_j \in \mathbb{R}_{\geq 0}$ for $j = n+1, \ldots, N$

Mixed Integer Linear Programming

Mixed integer linear program

- Minimize $\sum_j a_j x_j$
- Subject to $\sum_j a_{i,j} x_j \leq A_i$, $i = 1, \ldots, m$
- $x_j \in \mathbb{N}$ for $j = 1, \ldots, n$
- $x_j \in \mathbb{R}_{\geq 0}$ for $j = n+1, \ldots, N$

- Linear constraints
- Integer-valued variables
- Real-valued variables
- Linear objective function

Mixed Integer Linear Programming

Mixed integer linear program

- Minimize $\sum_j a_j x_j$
- Subject to $\sum_j a_{i,j} x_j \leq A_i$, $i = 1, \ldots, m$
- $x_j \in \mathbb{N}$ for $j = 1, \ldots, n$
- $x_j \in \mathbb{R}_{\geq 0}$ for $j = n+1, \ldots, N$

- Linear constraints
- Integer-valued variables
- Real-valued variables
- Linear objective function

- No real-valued variables: integer linear program (ILP)
- $0 \leq x_j \leq 1$ for all j: 0-1 ILP
- Vacuous objective $\sum_j 0 \cdot x_j$: decision problem
- But MIP makes most sense for optimization

Two Differences Compared to SAT/PB

Academia vs. industry

- Best solvers are commercial and closed-source
- E.g., CPLEX [CPL] (historically), Gurobi [Gur], and FICO Xpress [FIC]
- Academic solvers like SCIP [SCI] and HiGHS [HiG] are excellent but not as good

Two Differences Compared to SAT/PB

Academia vs. industry

- Best solvers are commercial and closed-source
- E.g., CPLEX [CPL] (historically), GUROBI [Gur], and FICO XPRESS [FIC]
- Academic solvers like SCIP [SCI] and HIGHS [HiG] are excellent but not as good

Search vs. backtracking

- SAT/PB: Fast decisions; careful, slow(er) conflict analysis
- MIP: Lots of time & effort on decisions; backtracking not so advanced

MIP Solving at a High Level

1. Preprocessing (called presolving)
2. Linear programming + branch-and-bound
3. Add cutting planes ruling out infeasible LP-solutions (branch-and-cut method going back to [Gom58])
4. Heuristics for quickly finding good feasible solutions

MIP Solving at a High Level

1. Preprocessing (called presolving)
2. Linear programming + branch-and-bound
3. Add cutting planes ruling out infeasible LP-solutions (branch-and-cut method going back to [Gom58])
4. Heuristics for quickly finding good feasible solutions

Far from exhaustive list...

Linear Programming Relaxation

Linear Programming Relaxation (LPR)

- Minimize $\sum_j a_j x_j$
- Subject to $\sum_j a_{i,j} x_j \leq A_i$, $i = 1, \ldots, m$
- ~~$x_j \in \mathbb{N}$ for $j = 1, \ldots, n$~~ $x_j \in \mathbb{R}_{\geq 0}$ for $j = 1, \ldots, n$
- $x_j \in \mathbb{R}_{\geq 0}$ for $j = n + 1, \ldots, N$

Linear Programming Relaxation

Linear Programming Relaxation (LPR)

- Minimize $\sum_j a_j x_j$
- Subject to $\sum_j a_{i,j} x_j \leq A_i$, $i = 1, \ldots, m$
- $\cancel{x_j \in \mathbb{N} \text{ for } j = 1, \ldots, n}$ $x_j \in \mathbb{R}_{\geq 0}$ for $j = 1, \ldots, n$
- $x_j \in \mathbb{R}_{\geq 0}$ for $j = n+1, \ldots, N$

- Fast to solve (just linear programming)
- LP solution x^* yields lower bound
- Or, if x^* "accidentally" feasible, have optimal solution
- Use simplex algorithm
  - many LP calls for same problem with different variable bounds
  - need efficient hot restarts

LP-Based Branch-and-Bound

Branch-and-bound

Choose integer-valued x_j and $B \in \mathbb{N}$

- Solve MIP plus constraint $x_j \geq B$
- Solve MIP plus constraint $x_j \leq B - 1$

LP-Based Branch-and-Bound

Branch-and-bound

Choose integer-valued x_j and $B \in \mathbb{N}$

- Solve MIP plus constraint $x_j \geq B$
- Solve MIP plus constraint $x_j \leq B - 1$

Creates (growing) branch-and-bound tree of subproblems

Prune subproblem/node when

- LP is infeasible
- LP bound $>$ incumbent (current best solution)

LP-Based Branch-and-Bound

Branch-and-bound

Choose integer-valued x_j and $B \in \mathbb{N}$

- Solve MIP plus constraint $x_j \geq B$
- Solve MIP plus constraint $x_j \leq B - 1$

Creates (growing) branch-and-bound tree of subproblems

Prune subproblem/node when

- LP is infeasible
- LP bound $>$ incumbent (current best solution)

Branch on

- Variables
- General linear constraints (powerful but difficult)
  Corresponds to stabbing planes proof system [BFI+18]

Branch-and-Cut

General cutting plane method

1. Solve LP relaxation
2. If solution x^* feasible for MIP $\Rightarrow$ found optimum
3. Otherwise generate and add constraint $\sum_j b_j x_j \leq B$ that is
    - valid for MIP
    - violated by LP solution x^*
4. Repeat from the top

Branch-and-Cut

General cutting plane method

1. Solve LP relaxation
2. If solution x^* feasible for MIP $\Rightarrow$ found optimum
3. Otherwise generate and add constraint $\sum_j b_j x_j \leq B$ that is
   - valid for MIP
   - violated by LP solution x^*
4. Repeat from the top

Pseudo-Boolean solving rules division and saturation are examples of cut rules

Branch-and-Cut

General cutting plane method

1. Solve LP relaxation
2. If solution x^* feasible for MIP $\Rightarrow$ found optimum
3. Otherwise generate and add constraint $\sum_j b_j x_j \leq B$ that is
   - valid for MIP
   - violated by LP solution x^*
4. Repeat from the top

Pseudo-Boolean solving rules division and saturation are examples of cut rules

Branch-and-cut

- Run branch-and-bound
- But in each subproblem, use cutting plane method to repeatedly solve LP relaxation and add cut

Example Cut 1: Knapsack Cover Cut

Given constraint

$$\sum_{j \in I} a_j x_j \leq A$$

for $x_j \in \{0,1\}$ and $a_j, A \in \mathbb{N}^+$

Example Cut 1: Knapsack Cover Cut

Given constraint

$$\sum_{j \in I} a_j x_j \leq A$$

for $x_j \in \{0,1\}$ and $a_j, A \in \mathbb{N}^+$

Find minimal cover $C \subseteq I$ such that

$$\sum_{j \in C} a_j > A$$

$$\sum_{j \in C \setminus \{i\}} a_j \leq A \qquad \text{for all } i \in C$$

Example Cut 1: Knapsack Cover Cut

Given constraint

$$\sum_{j \in I} a_j x_j \leq A$$

for $x_j \in \{0,1\}$ and $a_j, A \in \mathbb{N}^+$

Find minimal cover $C \subseteq I$ such that

$$\sum_{j \in C} a_j > A$$

$$\sum_{j \in C \setminus \{i\}} a_j \leq A \qquad \text{for all } i \in C$$

Then can derive

$$\sum_{j \in C} x_j \leq |C| - 1$$

Example Cut 1: Knapsack Cover Cut

Given constraint

$$\sum_{j \in I} a_j x_j \leq A$$

for $x_j \in \{0,1\}$ and $a_j, A \in \mathbb{N}^+$

Find minimal cover $C \subseteq I$ such that

$$\sum_{j \in C} a_j > A$$

$$\sum_{j \in C \setminus \{i\}} a_j \leq A \qquad \text{for all } i \in C$$

Then can derive

$$\sum_{j \in C} x_j \leq |C| - 1$$

(In cutting planes proof system, weaken & divide $\sum_{j \in I} a_j \overline{x}_j \geq -A + \sum_{j \in I} a_j$ to get disjunctive clause $\sum_{j \in C} \overline{x}_j \geq 1$)

Example Cut 2: Mixed Integer Rounding (MIR) Cut

Mixed integer rounding (MIR) cut [MW01] applied to (normalized) pseudo-Boolean constraint

$$\sum_i a_i \ell_i \geq A$$

with divisor $d \in \mathbb{N}^+$ produces constraint

$$\sum_i \Big(\min(a_i \bmod d, A \bmod d) + \left\lfloor \frac{a_i}{d} \right\rfloor (A \bmod d) \Big) \ell_i \geq \left\lceil \frac{A}{d} \right\rceil (A \bmod d)$$

Example Cut 2: Mixed Integer Rounding (MIR) Cut

Mixed integer rounding (MIR) cut [MW01] applied to (normalized) pseudo-Boolean constraint

$$\sum_i a_i \ell_i \geq A$$

with divisor $d \in \mathbb{N}^+$ produces constraint

$$\sum_i \left(\min(a_i \bmod d, A \bmod d) + \left\lfloor \frac{a_i}{d} \right\rfloor (A \bmod d) \right) \ell_i \geq \left\lceil \frac{A}{d} \right\rceil (A \bmod d)$$

Concretely, MIR cut with divisor 3 applied on

$$x + 2y + 3z + 4w + 5u \geq 5$$

(so $(A \bmod d) = (5 \bmod 3) = 2$)

Example Cut 2: Mixed Integer Rounding (MIR) Cut

Mixed integer rounding (MIR) cut [MW01] applied to (normalized) pseudo-Boolean constraint

$$\sum_i a_i \ell_i \geq A$$

with divisor $d \in \mathbb{N}^+$ produces constraint

$$\sum_i \left(\min(a_i \bmod d, A \bmod d) + \left\lfloor \frac{a_i}{d} \right\rfloor (A \bmod d) \right) \ell_i \geq \left\lceil \frac{A}{d} \right\rceil (A \bmod d)$$

Concretely, MIR cut with divisor 3 applied on

$$x + 2y + 3z + 4w + 5u \geq 5$$

(so $(A \bmod d) = (5 \bmod 3) = 2$) yields

$$x + 2y + 2z + 3w + 4u \geq 4$$

Example Cut 2: Mixed Integer Rounding (MIR) Cut

Mixed integer rounding (MIR) cut [MW01] applied to (normalized) pseudo-Boolean constraint

$$\sum_i a_i \ell_i \geq A$$

with divisor $d \in \mathbb{N}^+$ produces constraint

$$\sum_i \Big(\min(a_i \bmod d, A \bmod d) + \left\lfloor \frac{a_i}{d} \right\rfloor (A \bmod d) \Big) \ell_i \geq \left\lceil \frac{A}{d} \right\rceil (A \bmod d)$$

Concretely, MIR cut with divisor 3 applied on

$$x + 2y + 3z + 4w + 5u \geq 5$$

(so $(A \bmod d) = (5 \bmod 3) = 2$) yields

$$x + 2y + 2z + 3w + 4u \geq 4$$

For comparison, division by 3 and multiplication by 2 produces

$$2x + 2y + 2z + 4w + 4u \geq 4$$

Presolving

Presolving is a topic for a full separate lecture or two
(well, like most other aspects of MIP solving that we touch on. . .)

Important for performance (but not quite as important as in CDCL SAT solving?)

Presolving

Presolving is a topic for a full separate lecture or two
(well, like most other aspects of MIP solving that we touch on. . .)

Important for performance (but not quite as important as in CDCL SAT solving?)

Some simple (but efficient) techniques:

- Substitution of fixed variables
- Normalization of constraints: divide integer constraints by $\gcd$ on left-hand side and round on right-hand side
- Probing: tentatively assign binary variables and propagate
- Dominance test: remove constraints implied by other constraints

Presolving

Presolving is a topic for a full separate lecture or two
(well, like most other aspects of MIP solving that we touch on...)

Important for performance (but not quite as important as in CDCL SAT solving?)

Some simple (but efficient) techniques:

- Substitution of fixed variables
- Normalization of constraints: divide integer constraints by $\gcd$ on left-hand side and round on right-hand side
- Probing: tentatively assign binary variables and propagate
- Dominance test: remove constraints implied by other constraints

For more details, see talk by Gleixner `https://tinyurl.com/MIPtutorial`

MIP Conflict Analysis

MIP conflict analysis [Ach07] analogous to CDCL, but

- operate on clausal reasons extracted from constraints
- **not** on constraints themselves

Exponential loss in power!

MIP Conflict Analysis

MIP conflict analysis [Ach07] analogous to CDCL, but

- operate on clausal reasons extracted from constraints
- **not** on constraints themselves

Exponential loss in power!

Pigeonhole principle

$$\sum_{j=1}^{n} x_{i,j} \geq 1 \qquad i \in [n+1]$$
$$\sum_{i=1}^{n+1} x_{i,j} \leq 1 \qquad j \in [n]$$

Conflict analysis with clausal reasons $\Rightarrow$ same as resolution on CNF encoding $\Rightarrow$ exponential lower bound in [Hak85] applies

MIP Conflict Analysis

MIP conflict analysis [Ach07] analogous to CDCL, but

- operate on clausal reasons extracted from constraints
- **not** on constraints themselves

Exponential loss in power!

Pigeonhole principle

$$\sum_{j=1}^{n} x_{i,j} \geq 1 \qquad i \in [n+1]$$
$$\sum_{i=1}^{n+1} x_{i,j} \leq 1 \qquad j \in [n]$$

Conflict analysis with clausal reasons ⇒ same as resolution on CNF encoding ⇒ exponential lower bound in [Hak85] applies

Perhaps a bit stupid example—solved immediately, since LP relaxation is infeasible. . .

But can find other, more interesting benchmarks where MIP conflict analysis seems to really suffer from this problem [DGN21]

Branching Heuristics

Dual gain

Given LP solution x^*, branch on x_j such that $x_j \geq \lceil x_j^* \rceil$ and $x_j \leq \lfloor x_j^* \rfloor$ both provide good lower bound increase

Branching Heuristics

Dual gain

Given LP solution x^*, branch on x_j such that $x_j \geq \lceil x_j^* \rceil$ and $x_j \leq \lfloor x_j^* \rfloor$ both provide good lower bound increase

Look ahead (strong branching)

- Consider all free variables x_j
- Solve LP for all branching decisions $x_j \geq \lceil x_j^* \rceil$ and $x_j \leq \lfloor x_j^* \rfloor$
- Pick variable yielding strongest bound increases

Branching Heuristics

Dual gain

Given LP solution x^*, branch on x_j such that $x_j \geq \lceil x_j^* \rceil$ and $x_j \leq \lfloor x_j^* \rfloor$ both provide good lower bound increase

Look ahead (strong branching)

- Consider all free variables x_j
- Solve LP for all branching decisions $x_j \geq \lceil x_j^* \rceil$ and $x_j \leq \lfloor x_j^* \rfloor$
- Pick variable yielding strongest bound increases

Look back

Compute estimate on gains based on past branching history (pseudo-costs)

Branching Heuristics

Dual gain

Given LP solution x^*, branch on x_j such that $x_j \geq \lceil x_j^* \rceil$ and $x_j \leq \lfloor x_j^* \rfloor$ both provide good lower bound increase

Look ahead (strong branching)

- Consider all free variables x_j
- Solve LP for all branching decisions $x_j \geq \lceil x_j^* \rceil$ and $x_j \leq \lfloor x_j^* \rfloor$
- Pick variable yielding strongest bound increases

Look back

Compute estimate on gains based on past branching history (pseudo-costs)

Keep also other statistics about variables to guide search

Node Selection

How to grow search tree?

- Depth-first search (DFS): keeps cost for simplex calls small *[corresponds to what SAT and PB solvers* ***always*** *do]*
- Best bound search (BBS): Focus on improving lower bound (dual bound)
- Best estimate search (BES): Focus on improving solution (primal bound)

Node Selection

How to grow search tree?

- Depth-first search (DFS): keeps cost for simplex calls small *[corresponds to what SAT and PB solvers **always** do]*
- Best bound search (BBS): Focus on improving lower bound (dual bound)
- Best estimate search (BES): Focus on improving solution (primal bound)

Combine BBS and BES with DFS plunges to exploit simplex hot restarts

Primal Heuristics

- Improve solution (primal bound)
- Guide remaining search

Primal Heuristics

- Improve solution (primal bound)
- Guide remaining search

Example: Relaxation-enforced neighbourhood search

1. Solve LP relaxation to get x^*
2. Fix values of all x_j such that $x_j^* \in \mathbb{N}$
3. For x_j with fractional solution, reduce domain to $x_j \in \{\lfloor x_j^* \rfloor, \lceil x_j^* \rceil\}$
4. Solve new subproblem

Primal Heuristics

- Improve solution (primal bound)
- Guide remaining search

Example: Relaxation-enforced neighbourhood search

1. Solve LP relaxation to get x^*
2. Fix values of all x_j such that $x_j^* \in \mathbb{N}$
3. For x_j with fractional solution, reduce domain to $x_j \in \{\lfloor x_j^* \rfloor, \lceil x_j^* \rceil\}$
4. Solve new subproblem

Example of "fix-and-MIP" local neighbourhood search heuristic
(Note that, interestingly, this turns ILP into 0-1 ILP subproblem)

And More. . .

1. Decomposition
   - Branch-and-price / column generation
   - Bender's decomposition
     [Core-guided and IHS search similar in spirit to logic-based Benders decomposition [HO03]]
2. Symmetry handling
   - Via graph automorphism
   - Or dedicated symmetry detection (commercial solvers)
3. Extended formulations (with new variables and constraints)
4. Parallelization
5. Restarts

Numerics and Correctness

Numerics

- Use floating point for efficiency reasons
- Can lead to rounding errors
- Exact MIP solvers like [CKSW13, EG23]
  - are significantly slower
  - don't support the full range of state-of-the-art techniques

Numerics and Correctness

Numerics

- Use floating point for efficiency reasons
- Can lead to rounding errors
- Exact MIP solvers like [CKSW13, EG23]
  - are significantly slower
  - don't support the full range of state-of-the-art techniques

Proof logging / certification

- Currently not available for state-of-the-art MIP solvers
- Though known that even best commercial solvers sometimes give wrong results
- Some work on proof logging in [CGS17, EG23] — challenges:
  - How to capture wide diversity of techniques?
  - What is a convenient format?
  - How to generate proofs efficiently on-the-fly?

Some Interesting MIP Questions

1. Develop better heuristics to branch on general linear constraints (cf. stabbing planes [BFI+18])
2. Design stronger conflict analysis operating directly on linear constraints (borrowing ideas from native pseudo-Boolean solvers) *[Recent work [MBGN23, MSB+24]]*
3. Provide rigorous understanding of MIP solver performance
4. Develop families of theory benchmarks and computational complexity results for them (cf. interaction between SAT solving and proof complexity [BN21])
5. Steal pseudo-Boolean proof logging ideas and techniques and use for MIP solving *[Recent work [DEGH23, HOGN24]]*
6. Steal best MIP ideas and techniques and use for pseudo-Boolean solving!?

Some Interesting MIP Questions

1. Develop better heuristics to branch on general linear constraints (cf. stabbing planes [BFI⁺18])
2. Design stronger conflict analysis operating directly on linear constraints (borrowing ideas from native pseudo-Boolean solvers) *[Recent work [MBGN23, MSB⁺24]]*
3. Provide rigorous understanding of MIP solver performance
4. Develop families of theory benchmarks and computational complexity results for them (cf. interaction between SAT solving and proof complexity [BN21])
5. Steal pseudo-Boolean proof logging ideas and techniques and use for MIP solving *[Recent work [DEGH23, HOGN24]]*
6. Steal best MIP ideas and techniques and use for pseudo-Boolean solving!? *[Next and final topic]*

Combining Pseudo-Boolean Solving and Mixed Integer Programming

Pseudo-Boolean solvers

- Sophisticated conflict analysis using cutting planes method
- Sometimes terrible performance even when LP relaxation infeasible [EGNV18]

Combining Pseudo-Boolean Solving and Mixed Integer Programming

Pseudo-Boolean solvers

- Sophisticated conflict analysis using cutting planes method
- Sometimes terrible performance even when LP relaxation infeasible [EGNV18]

Mixed integer linear programming solvers

- Powerful search
- Exploits information from LP relaxations
- Rich variety of cut generation routines
- But conflict analysis not so great. . .

Combining Pseudo-Boolean Solving and Mixed Integer Programming

Pseudo-Boolean solvers

- Sophisticated conflict analysis using cutting planes method
- Sometimes terrible performance even when LP relaxation infeasible [EGNV18]

Mixed integer linear programming solvers

- Powerful search
- Exploits information from LP relaxations
- Rich variety of cut generation routines
- But conflict analysis not so great...

Why not merge the two to get the best of both worlds of pseudo-Boolean conflict-driven search and MIP-style branch-and-cut?

Balance Time Allocation for PB and LP Solving?

High-level idea: Give pseudo-Boolean solver access to LP solver

Balance Time Allocation for PB and LP Solving?

High-level idea: Give pseudo-Boolean solver access to LP solver

First challenge:

1. Using LP solver as preprocessor not sufficient
   - Pseudo-Boolean formulas can have feasible LP relaxations
   - but quickly turn infeasible after just a couple of decisions
   - Some such benchmarks very hard for PB solvers [EGNV18]

Balance Time Allocation for PB and LP Solving?

High-level idea: Give pseudo-Boolean solver access to LP solver

First challenge:

1. Using LP solver as preprocessor not sufficient
   - Pseudo-Boolean formulas can have feasible LP relaxations
   - but quickly turn infeasible after just a couple of decisions
   - Some such benchmarks very hard for PB solvers [EGNV18]
2. Consulting LP solver before each variable decision impractical
   - Pseudo-Boolean solving based on rapid alternation of decisions and propagations
   - Solving an LP relaxation is orders of magnitude slower

Balance Time Allocation for PB and LP Solving?

High-level idea: Give pseudo-Boolean solver access to LP solver

First challenge:

1. Using LP solver as preprocessor not sufficient
   - Pseudo-Boolean formulas can have feasible LP relaxations
   - but quickly turn infeasible after just a couple of decisions
   - Some such benchmarks very hard for PB solvers [EGNV18]
2. Consulting LP solver before each variable decision impractical
   - Pseudo-Boolean solving based on rapid alternation of decisions and propagations
   - Solving an LP relaxation is orders of magnitude slower

Need to carefully balance time allocation for PB solver and LP solver

Backtracking from LP Infeasibility?

What to do if LP solver call shows LP relaxation infeasible under current trail?

- Obviously, PB solver should backtrack
- But can only do conflict analysis on violated pseudo-Boolean constraint
- And PB solver blissfully unaware of any conflict. . .

Backtracking from LP Infeasibility?

What to do if LP solver call shows LP relaxation infeasible under current trail?

- Obviously, PB solver should backtrack
- But can only do conflict analysis on violated pseudo-Boolean constraint
- And PB solver blissfully unaware of any conflict...

More subtle issue:

- Efficient LP solvers use inexact floating-point arithmetic
- How to incorporate LP solver inferences into Boolean solver that must maintain perfectly sound reasoning?

Sharing of Cut Constraints?

Cut constraints from LP solver

- When LP relaxation feasible, MIP solver generates cut constraint to remove the found LP solution
- Should such constraints be shared with the PB solver?

Sharing of Cut Constraints?

Cut constraints from LP solver

- When LP relaxation feasible, MIP solver generates cut constraint to remove the found LP solution
- Should such constraints be shared with the PB solver?

Cut constraints from PB solver

- PB solvers learns new constraints at high rate from conflict analysis
- These learned constraints can also be viewed as cuts
- Should such constraints be passed from PB solver to LP solver?

Report on Proof-of-Concept PB-LP Integration [DGN21]

1. Interleave LP solving within conflict-driven PB search
   - Limit LP time by enforcing total #LP pivots $\leq$ #PB conflicts
   - Only run LP solver when this condition holds
   - Abort if $> P$ pivots in single LP call; but if so also double limit P to avoid wasted LP calls in future

Report on Proof-of-Concept PB-LP Integration [DGN21]

1. Interleave LP solving within conflict-driven PB search
   - Limit LP time by enforcing total $\#$LP pivots $\leq$ $\#$PB conflicts
   - Only run LP solver when this condition holds
   - Abort if $> P$ pivots in single LP call; but if so also double limit P to avoid wasted LP calls in future
2. When LP solver detects that LP relaxation infeasible
   - Farkas' lemma $\Rightarrow$ violated linear combination of constraints
   - Use this Farkas constraint as starting point for conflict analysis
   - Computed using exact arithmetic, so no rounding errors
   - But might not be violated — if so, ignore and continue PB search

Report on Proof-of-Concept PB-LP Integration [DGN21]

1. Interleave LP solving within conflict-driven PB search
   - Limit LP time by enforcing total #LP pivots $\leq$ #PB conflicts
   - Only run LP solver when this condition holds
   - Abort if $> P$ pivots in single LP call; but if so also double limit P to avoid wasted LP calls in future
2. When LP solver detects that LP relaxation infeasible
   - Farkas' lemma $\Rightarrow$ violated linear combination of constraints
   - Use this Farkas constraint as starting point for conflict analysis
   - Computed using exact arithmetic, so no rounding errors
   - But might not be violated — if so, ignore and continue PB search
3. When LP solver finds solution to LP relaxation
   - Generate MIP-style Gomory cut
   - Share constraint to tighten search space on both PB and LP side
   - Try to use LP solution to guide PB search (e.g., variable decisions)

Report on Proof-of-Concept PB-LP Integration [DGN21]

1. Interleave LP solving within conflict-driven PB search
   - Limit LP time by enforcing total #LP pivots $\leq$ #PB conflicts
   - Only run LP solver when this condition holds
   - Abort if $> P$ pivots in single LP call; but if so also double limit P to avoid wasted LP calls in future
2. When LP solver detects that LP relaxation infeasible
   - Farkas' lemma $\Rightarrow$ violated linear combination of constraints
   - Use this Farkas constraint as starting point for conflict analysis
   - Computed using exact arithmetic, so no rounding errors
   - But might not be violated — if so, ignore and continue PB search
3. When LP solver finds solution to LP relaxation
   - Generate MIP-style Gomory cut
   - Share constraint to tighten search space on both PB and LP side
   - Try to use LP solution to guide PB search (e.g., variable decisions)
4. Also explore letting PB solver pass learned constraints to LP solver

(What We Need from) Farkas Lemma [Far02]

Pseudo-Boolean Farkas Lemma

Given

- Pseudo-Boolean formula $F = \{C_1, \ldots, C_m\}$,
- partial assignment ρ,

such that LP relaxation of residual formula $F{\upharpoonright_\rho}$ infeasible
Then $\exists$ coefficients $k_i \in \mathbb{N}$ such that linear combination

$$\sum_{i=1}^{m} k_i \cdot C_i$$

is violated by ρ, i.e.,

$$\mathit{slack}\left(\sum_{i=1}^{m} k_i \cdot C_i; \rho\right) < 0$$

Observed in [MM04] that $\sum_{i=1}^{m} k_i \cdot C_i$ is valid starting point for PB conflict analysis

Relation to MIP Solvers with Conflict Analysis?

MIP solvers also combine constraint propagation and SAT-style clause learning with LP solving

- Implemented in SCIP [ABKW08]
- And also in closed-source solvers (see [AW13])

Important to understand similarities and differences — let's give high-level description of PB search and conflict analysis phrased in MIP language

Relation to MIP Solvers with Conflict Analysis?

MIP solvers also combine constraint propagation and SAT-style clause learning with LP solving

- Implemented in SCIP [ABKW08]
- And also in closed-source solvers (see [AW13])

Important to understand similarities and differences — let's give high-level description of PB search and conflict analysis phrased in MIP language

Pseudo-Boolean search

1. Make decision to assign free variable to 0 or 1
2. Propagate all assignments implied by some linear constraint until saturation
3. If no contradiction, go to step 1
4. Otherwise some constraint C violated $\Rightarrow$ trigger conflict analysis

PB Conflict Analysis "in MIP Language"

Pseudo-Boolean conflict analysis (simplified description)

1. Find reason constraint R responsible for propagating last variable x in violated constraint C to "wrong value"

PB Conflict Analysis "in MIP Language"

Pseudo-Boolean conflict analysis (simplified description)

1. Find reason constraint R responsible for propagating last variable x in violated constraint C to "wrong value"
2. Apply division/saturation to generate (globally valid) cut R_{cut} propagating x to $\{0,1\}$-value (over the reals)

PB Conflict Analysis "in MIP Language"

Pseudo-Boolean conflict analysis (simplified description)

1. Find reason constraint R responsible for propagating last variable x in violated constraint C to "wrong value"
2. Apply division/saturation to generate (globally valid) cut R_{cut} propagating x to $\{0,1\}$-value (over the reals)
3. Set $D :=$ smallest integer linear combination of R_{cut} and C for which x cancels — D violated by current solvers assignment with x removed

PB Conflict Analysis "in MIP Language"

Pseudo-Boolean conflict analysis (simplified description)

1. Find reason constraint R responsible for propagating last variable x in violated constraint C to "wrong value"
2. Apply division/saturation to generate (globally valid) cut R_{cut} propagating x to $\{0, 1\}$-value (over the reals)
3. Set $D :=$ smallest integer linear combination of R_{cut} and C for which x cancels — D violated by current solvers assignment with x removed
4. Unless D satisfies termination criterion (assertiveness), set $C := D$ and go to step 1

PB Conflict Analysis "in MIP Language"

Pseudo-Boolean conflict analysis (simplified description)

1. Find reason constraint R responsible for propagating last variable x in violated constraint C to "wrong value"
2. Apply division/saturation to generate (globally valid) cut R_{cut} propagating x to $\{0,1\}$-value (over the reals)
3. Set $D :=$ smallest integer linear combination of R_{cut} and C for which x cancels — D violated by current solvers assignment with x removed
4. Unless D satisfies termination criterion (assertiveness), set $C := D$ and go to step 1
5. Learn assertive D, i.e., add to solver database of constraints

PB Conflict Analysis "in MIP Language"

Pseudo-Boolean conflict analysis (simplified description)

1. Find reason constraint R responsible for propagating last variable x in violated constraint C to "wrong value"
2. Apply division/saturation to generate (globally valid) cut R_{cut} propagating x to $\{0, 1\}$-value (over the reals)
3. Set $D :=$ smallest integer linear combination of R_{cut} and C for which x cancels — D violated by current solvers assignment with x removed
4. Unless D satisfies termination criterion (assertiveness), set $C := D$ and go to step 1
5. Learn assertive D, i.e., add to solver database of constraints
6. Backjump by undoing further assignments in reverse chronological order until D is no longer violated

PB Conflict Analysis “in MIP Language”

Pseudo-Boolean conflict analysis (simplified description)

1. Find reason constraint R responsible for propagating last variable x in violated constraint C to “wrong value”
2. Apply division/saturation to generate (globally valid) cut R_{cut} propagating x to $\{0,1\}$-value (over the reals)
3. Set $D :=$ smallest integer linear combination of R_{cut} and C for which x cancels — D violated by current solvers assignment with x removed
4. Unless D satisfies termination criterion (assertiveness), set $C := D$ and go to step 1
5. Learn assertive D, i.e., add to solver database of constraints
6. Backjump by undoing further assignments in reverse chronological order until D is no longer violated
7. Switch back to search phase

Comparison to MIP Propagation and Conflict Analysis

Propagation in SCIP

- Fast, simple propagation in PB solvers
- Plus powerful, but slower, method of solving LP relaxations

Comparison to MIP Propagation and Conflict Analysis

Propagation in SCIP

- Fast, simple propagation in PB solvers
- Plus powerful, but slower, method of solving LP relaxations

Conflict analysis in SCIP [Ach07]

- Perform derivation not on reason constraints R as described above
- Instead use disjunctive clauses extracted from reason constraints via conflict graph
- Incurs exponential loss in power compared to operating on actual linear constraints (follows from [BKS04, CCT87, Hak85])

Comparison to MIP Propagation and Conflict Analysis

Propagation in SCIP

- Fast, simple propagation in PB solvers
- Plus powerful, but slower, method of solving LP relaxations

Conflict analysis in SCIP [Ach07]

- Perform derivation not on reason constraints R as described above
- Instead use disjunctive clauses extracted from reason constraints via conflict graph
- Incurs exponential loss in power compared to operating on actual linear constraints (follows from [BKS04, CCT87, Hak85])

Arithmetic

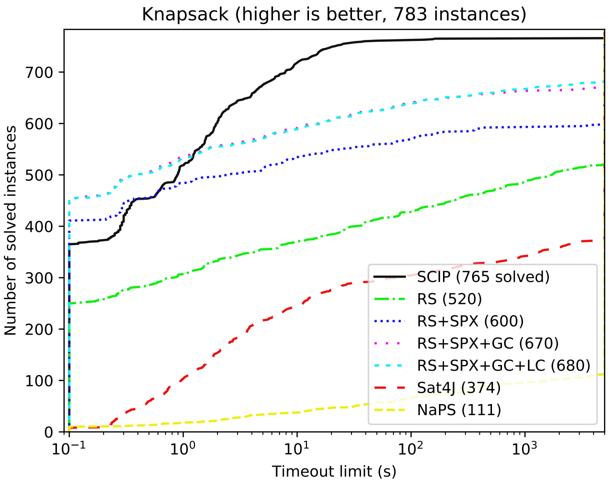
- SCIP uses floating point
- Reasoning steps in pseudo-Boolean solver computed with exact integer arithmetic
- No issues with possible rounding errors

Experimental Results for Knapsack Benchmarks [Pis05]

ROUNDINGSAT (RS) enhanced with

- LP solver SOPLEX (SPX) (from SCIP)
- Gomory cuts (GC)
- shared learned PB cuts (LC)

compared to other solvers



Experimental Results for PB and MIPLIB Benchmarks

RoundingSat (RS) run on PB and 0-1 ILP instances with

- LP solver (+SPX)
- plus Gomory cuts (+GC)
- plus sharing cuts learned by PB solver (+LC)

compared to other solvers

instances solved (to optimality for optimization problems)
Highlighting **1st**, **2nd**, and **3rd** best

Experimental Results for PB and MIPLIB Benchmarks

ROUNDINGSAT (RS) run on PB and 0-1 ILP instances with

- LP solver (+SPX)
- plus Gomory cuts (+GC)
- plus sharing cuts learned by PB solver (+LC)

compared to other solvers

instances solved (to optimality for optimization problems)
Highlighting **1st**, **2nd**, and **3rd** best

	SCIP	RS	+SPX	+GC	+LC	SAT4J	NAPS
PB16dec (1783)	1123	**1472**	**1453**	**1452**	1451	1432	1400
PB16opt (1600)	**1057**	862	**988**	986	**993**	776	896
MIPdec (556)	**264**	203	**263**	**261**	259	169	170
MIPopt (291)	**125**	78	101	**102**	**102**	62	65

Performance of Integrated PB-LP Solver

1. Best of both worlds?
   - At least well-rounded performance
   - Hybrid PB-LP solver always competitive with best solver
   - Pretty dramatic improvements for optimization problems compared to pseudo-Boolean state of the art
   - SCIP is hard to beat, but also pulls quite a few extra tricks that we haven't implemented (like problem-type-specific approaches)

Performance of Integrated PB-LP Solver

1. Best of both worlds?
    - At least well-rounded performance
    - Hybrid PB-LP solver always competitive with best solver
    - Pretty dramatic improvements for optimization problems compared to pseudo-Boolean state of the art
    - SCIP is hard to beat, but also pulls quite a few extra tricks that we haven't implemented (like problem-type-specific approaches)
2. Adding LP solving causes performance loss on PB decision instances
    - Worse results on satisfiable instances
    - Better search (lower conflict count) but slower — doesn't pay off in terms of running time

Performance of Integrated PB-LP Solver

1. Best of both worlds?
   - At least well-rounded performance
   - Hybrid PB-LP solver always competitive with best solver
   - Pretty dramatic improvements for optimization problems compared to pseudo-Boolean state of the art
   - SCIP is hard to beat, but also pulls quite a few extra tricks that we haven't implemented (like problem-type-specific approaches)
2. Adding LP solving causes performance loss on PB decision instances
   - Worse results on satisfiable instances
   - Better search (lower conflict count) but slower — doesn't pay off in terms of running time
3. Sharing Gomory cuts and learned cuts not so helpful
   - Except for knapsack benchmarks, where they help a lot
   - And maybe we could/should fine-tune how sharing is done?

Usefulness/Usage of Constraints

Estimate usefulness of different types of constraints

- Proxy: how often used in conflict analysis?
- Certainly not perfect measure
- But hopefully tells us something interesting

Usefulness/Usage of Constraints

Estimate usefulness of different types of constraints

- Proxy: how often used in conflict analysis?
- Certainly not perfect measure
- But hopefully tells us something interesting

Farkas constraints

- More useful than regular learned constraints for optimization problems
- Not so for decision problems

Usefulness/Usage of Constraints

Estimate usefulness of different types of constraints

- Proxy: how often used in conflict analysis?
- Certainly not perfect measure
- But hopefully tells us something interesting

Farkas constraints

- More useful than regular learned constraints for optimization problems
- Not so for decision problems

Constraints learned after Farkas-based conflicts

- Less useful than regular learned constraints
- But big spread in usage measurements

Pseudo-Boolean Solver Performance: Balancing the Picture

To provide fuller view, should also be mentioned that RoundingSat can outperform commercial MIP solvers by 1-2 orders of magnitude for problems such as, e.g.,

- matching of children with adoptive families [DGG+19]
- automated planning using binarized neural networks (BNNs) [SS18]

as reported by authors of these papers

Pseudo-Boolean Solver Performance: Balancing the Picture

To provide fuller view, should also be mentioned that RoundingSat can outperform commercial MIP solvers by 1-2 orders of magnitude for problems such as, e.g.,

- matching of children with adoptive families [DGG+19]
- automated planning using binarized neural networks (BNNs) [SS18]

as reported by authors of these papers

See also our paper [SDNS20] on problems involving BNNs

Pseudo-Boolean Solver Performance: Balancing the Picture

To provide fuller view, should also be mentioned that RoundingSat can outperform commercial MIP solvers by 1-2 orders of magnitude for problems such as, e.g.,

- matching of children with adoptive families [DGG+19]
- automated planning using binarized neural networks (BNNs) [SS18]

as reported by authors of these papers

See also our paper [SDNS20] on problems involving BNNs

RoundingSat seems particularly good for "big-M constraints" like

$$A\overline{z} + \sum_i a_i \ell_i \geq A$$

encoding $z \Rightarrow \sum_i a_i \ell_i \geq A$

Coefficient A of $\overline{z}$ can be very large compared to a_i's $\Rightarrow$ LP relaxation quite uninformative

Future Research Directions for PB-LP Integration (1/2)

1. Fine-tune heuristics
   - Improved LP-based cut generation?
   - Smarter sharing of PB constraints with LP solver?
   - Dynamic allocation of PB and LP solving time based on contributions?

Future Research Directions for PB-LP Integration (1/2)

1. Fine-tune heuristics
   - Improved LP-based cut generation?
   - Smarter sharing of PB constraints with LP solver?
   - Dynamic allocation of PB and LP solving time based on contributions?
2. Understand better how constraints from LP solver contribute
   - Why are Farkas constraints so useful?
   - But constraints learned from Farkas conflicts **not** useful?

Future Research Directions for PB-LP Integration (1/2)

1. Fine-tune heuristics
   - Improved LP-based cut generation?
   - Smarter sharing of PB constraints with LP solver?
   - Dynamic allocation of PB and LP solving time based on contributions?
2. Understand better how constraints from LP solver contribute
   - Why are Farkas constraints so useful?
   - But constraints learned from Farkas conflicts **not** useful?
3. Make more intelligent use in pseudo-Boolean solver of information from solutions to LP relaxations

Future Research Directions for PB-LP Integration (1/2)

1. Fine-tune heuristics
   - Improved LP-based cut generation?
   - Smarter sharing of PB constraints with LP solver?
   - Dynamic allocation of PB and LP solving time based on contributions?
2. Understand better how constraints from LP solver contribute
   - Why are Farkas constraints so useful?
   - But constraints learned from Farkas conflicts **not** useful?
3. Make more intelligent use in pseudo-Boolean solver of information from solutions to LP relaxations
4. Use MIP presolving in pseudo-Boolean solvers

Future Research Directions for PB-LP Integration (1/2)

1. Fine-tune heuristics
   - Improved LP-based cut generation?
   - Smarter sharing of PB constraints with LP solver?
   - Dynamic allocation of PB and LP solving time based on contributions?
2. Understand better how constraints from LP solver contribute
   - Why are Farkas constraints so useful?
   - But constraints learned from Farkas conflicts **not** useful?
3. Make more intelligent use in pseudo-Boolean solver of information from solutions to LP relaxations
4. Use MIP presolving in pseudo-Boolean solvers
5. Use MIR cuts and/or other MIP cut rules to improve conflict analysis [MBGN23]

Future Research Directions for PB-LP Integration (2/2)

6. Combine LP solver with core-guided search or IHS approach

Future Research Directions for PB-LP Integration (2/2)

6. Combine LP solver with core-guided search or IHS approach
7. Improve pseudo-Boolean search
   - RoundingSat-like solving with LP integration and/or core-guided search or IHS seems to be state of the art for pseudo-Boolean optimization
   - But solver much better on unsatisfiable instances (proving optimality) than on satisfiable ones (finding solutions)

Future Research Directions for PB-LP Integration (2/2)

6. Combine LP solver with core-guided search or IHS approach
7. Improve pseudo-Boolean search
   - ROUNDINGSAT-like solving with LP integration and/or core-guided search or IHS seems to be state of the art for pseudo-Boolean optimization
   - But solver much better on unsatisfiable instances (proving optimality) than on satisfiable ones (finding solutions)
8. Export pseudo-Boolean conflict analysis to MIP
   [Ongoing work in [MBGN23, MSB^{+}24]]

Future Research Directions for PB-LP Integration (2/2)

6. Combine LP solver with core-guided search or IHS approach
7. Improve pseudo-Boolean search
   - RoundingSat-like solving with LP integration and/or core-guided search or IHS seems to be state of the art for pseudo-Boolean optimization
   - But solver much better on unsatisfiable instances (proving optimality) than on satisfiable ones (finding solutions)
8. Export pseudo-Boolean conflict analysis to MIP
   [Ongoing work in [MBGN23, MSB$^+$24]]
9. Develop hybrid PB-LP solver to solve 0-1 MIP problems à la Bender
   - PB solver decides on Boolean variables and propagates
   - LP solver takes care of real-valued variables

Take-Away Message (for This and the Other Tutorials)

- Revolution in performance last two decades in
  - Boolean satisfiability (SAT) solving
  - Mixed integer linear programming (MIP)
- More recent addition: Cutting-planes-based pseudo-Boolean conflict-driven search
- Quite different approaches
  - Complementary strengths
  - Clear potential for synergies
- Lots of exciting research waiting to be done! ☺

Take-Away Message (for This and the Other Tutorials)

- Revolution in performance last two decades in
  - Boolean satisfiability (SAT) solving
  - Mixed integer linear programming (MIP)
- More recent addition: Cutting-planes-based pseudo-Boolean conflict-driven search
- Quite different approaches
  - Complementary strengths
  - Clear potential for synergies
- Lots of exciting research waiting to be done! ☺
- We're hiring! See `www.jakobnordstrom.se/openings`

Take-Away Message (for This and the Other Tutorials)

- Revolution in performance last two decades in
  - Boolean satisfiability (SAT) solving
  - Mixed integer linear programming (MIP)
- More recent addition: Cutting-planes-based pseudo-Boolean conflict-driven search
- Quite different approaches
  - Complementary strengths
  - Clear potential for synergies
- Lots of exciting research waiting to be done! ☺
- We're hiring! See `www.jakobnordstrom.se/openings`

Thanks for sticking till the end!

References I

[ABKW08] Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In *Proceedings of the 5th International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR '08)*, volume 5015 of *Lecture Notes in Computer Science*, pages 6–20. Springer, May 2008.

[Ach07] Tobias Achterberg. Conflict analysis in mixed integer programming. *Discrete Optimization*, 4(1):4–20, March 2007.

[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.

[BFI+18] Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere. Stabbing planes. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS '18)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:20, January 2018.

References II

[BKS04] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, December 2004. Preliminary version in *IJCAI '03*.

[BN21] Samuel R. Buss and Jakob Nordström. Proof complexity and SAT solving. In 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*, chapter 7, pages 233–350. IOS Press, 2nd edition, February 2021.

[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.

[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 III

[CPL] IBM ILOG CPLEX optimization studio. https://www.ibm.com/products/ilog-cplex-optimization-studio.

[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.

[DGG+19] Maxence Delorme, Sergio García, Jacek Gondzioa, Jörg Kalcsics, David Manlove, and William Pettersson. Mathematical models for stable matching problems with ties and incomplete lists. *European Journal of Operational Research*, 277(2):426–441, September 2019.

[DGN21] Jo Devriendt, Ambros Gleixner, and Jakob Nordström. Learn to relax: Integrating 0-1 integer linear programming with pseudo-Boolean conflict-driven search. *Constraints*, 26(1–4):26–55, October 2021. Preliminary version in *CPAIOR '20*.

[EG23] Leon Eifler and Ambros Gleixner. A computational status update for exact rational mixed integer programming. *Mathematical Programming*, 197(2):793–812, February 2023.

References IV

[EGNV18] Jan Elffers, Jesús Giráldez-Cru, Jakob Nordström, and Marc Vinyals. Using combinatorial benchmarks to probe the reasoning power of pseudo-Boolean solvers. 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 75–93. Springer, July 2018.

[Far02] Julius Farkas. Theorie der einfachen Ungleichungen. *Journal für die Reine und Angewandte Mathematik*, 1902(124):1–27, 1902.

[FIC] FICO Xpress optimization. `https://www.fico.com/en/products/fico-xpress-optimization`.

[Gom58] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[Gur] Gurobi optimizer. `https://www.gurobi.com/`.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[HiG] HiGHS – high performance software for linear optimization. `https://highs.dev/`.

References V

[HO03] John N. Hooker and Greger Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96(1):33–60, April 2003.

[HOGN24] Alexander Hoen, Andy Oertel, Ambros Gleixner, and Jakob Nordström. Certifying MIP-based presolve reductions for 0–1 integer linear programs. In *Proceedings of the 21st International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR '24)*, volume 14742 of *Lecture Notes in Computer Science*, pages 310–328. Springer, May 2024.

[MBGN23] Gioni Mexi, Timo Berthold, Ambros Gleixner, and Jakob Nordström. Improving conflict analysis in MIP solvers by pseudo-Boolean reasoning. 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 27:1–26:19, August 2023.

[MM04] Vasco M. Manquinho and João P. Marques-Silva. Satisfiability-based algorithms for Boolean optimization. *Annals of Mathematics and Artificial Intelligence*, 40(1):353–372, March 2004.

[MSB+24] Gioni Mexi, Felipe Serrano, Timo Berthold, Ambros Gleixner, and Jakob Nordström. Cut-based conflict analysis in mixed integer programming. Technical Report 2410.15110, arXiv.org, October 2024.

References VI

[MW01] Hugues Marchand and Laurence A. Wolsey. Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, 49(3):325–468, June 2001.

[Pis05] David Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, September 2005.

[SCI] SCIP: Solving constraint integer programs. `https://scipopt.org`.

[SDNS20] Buser Say, Jo Devriendt, Jakob Nordström, and Peter Stuckey. Theoretical and experimental results for planning with learned binarized neural network transition models. 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 917–934. Springer, September 2020.

[SS18] Buser Say and Scott Sanner. Planning in factored state and action spaces with learned binarized neural network transition models. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*, pages 4815–4821, July 2018.