Pseudo-Boolean Solving: In Between SAT and ILP

Jakob Nordström

University of Copenhagen
and Lund University

Theoretical Foundations of SAT/SMT Solving
Simons Institute for the Theory of Computing
March 17, 2021

Pseudo-Boolean?

Pseudo-Boolean function: $f : \{0,1\}^n \to \mathbb{R}$

Studied since 1960s in operations research and 0-1 integer linear programming [BH02]

Restricted version: f represented as linear form [focus of this talk]

Many problems expressible as optimizing value of linear pseudo-Boolean function under linear pseudo-Boolean constraints

Pseudo-Boolean?

Pseudo-Boolean function: $f : \{0,1\}^n \rightarrow \mathbb{R}$

Studied since 1960s in operations research and 0-1 integer linear programming [BH02]

Restricted version: f represented as linear form [focus of this talk]

Many problems expressible as optimizing value of linear pseudo-Boolean function under linear pseudo-Boolean constraints

---

See Simons boot camp tutorial `https://tinyurl.com/pbsolving` for (much) longer version of this talk

Pseudo-Boolean vs. SAT

- Pseudo-Boolean format richer than conjunctive normal form (CNF)

Compare

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

and

$$\begin{aligned}
&(x_1 \lor x_2 \lor x_3 \lor x_4) \land (x_1 \lor x_2 \lor x_3 \lor x_5) \land (x_1 \lor x_2 \lor x_3 \lor x_6) \\
\land&(x_1 \lor x_2 \lor x_4 \lor x_5) \land (x_1 \lor x_2 \lor x_4 \lor x_6) \land (x_1 \lor x_2 \lor x_5 \lor x_6) \\
\land&(x_1 \lor x_3 \lor x_4 \lor x_5) \land (x_1 \lor x_3 \lor x_4 \lor x_6) \land (x_1 \lor x_3 \lor x_4 \lor x_6) \\
\land&(x_1 \lor x_4 \lor x_5 \lor x_6) \land (x_2 \lor x_3 \lor x_4 \lor x_5) \land (x_2 \lor x_3 \lor x_4 \lor x_6) \\
\land&(x_2 \lor x_3 \lor x_5 \lor x_6) \land (x_2 \lor x_4 \lor x_5 \lor x_6) \land (x_3 \lor x_4 \lor x_5 \lor x_6)
\end{aligned}$$

Pseudo-Boolean vs. SAT

- Pseudo-Boolean format richer than conjunctive normal form (CNF)

Compare

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

and

$$\begin{aligned}
&(x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3 \vee x_5) \wedge (x_1 \vee x_2 \vee x_3 \vee x_6) \\
\wedge&(x_1 \vee x_2 \vee x_4 \vee x_5) \wedge (x_1 \vee x_2 \vee x_4 \vee x_6) \wedge (x_1 \vee x_2 \vee x_5 \vee x_6) \\
\wedge&(x_1 \vee x_3 \vee x_4 \vee x_5) \wedge (x_1 \vee x_3 \vee x_4 \vee x_6) \wedge (x_1 \vee x_3 \vee x_4 \vee x_6) \\
\wedge&(x_1 \vee x_4 \vee x_5 \vee x_6) \wedge (x_2 \vee x_3 \vee x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4 \vee x_6) \\
\wedge&(x_2 \vee x_3 \vee x_5 \vee x_6) \wedge (x_2 \vee x_4 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_5 \vee x_6)
\end{aligned}$$

- And pseudo-Boolean reasoning exponentially stronger than conflict-driven clause learning (CDCL)

Pseudo-Boolean vs. SAT

- Pseudo-Boolean format richer than conjunctive normal form (CNF)

Compare

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

and

$$\begin{aligned} &(x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3 \vee x_5) \wedge (x_1 \vee x_2 \vee x_3 \vee x_6) \\ \wedge &(x_1 \vee x_2 \vee x_4 \vee x_5) \wedge (x_1 \vee x_2 \vee x_4 \vee x_6) \wedge (x_1 \vee x_2 \vee x_5 \vee x_6) \\ \wedge &(x_1 \vee x_3 \vee x_4 \vee x_5) \wedge (x_1 \vee x_3 \vee x_4 \vee x_6) \wedge (x_1 \vee x_3 \vee x_4 \vee x_6) \\ \wedge &(x_1 \vee x_4 \vee x_5 \vee x_6) \wedge (x_2 \vee x_3 \vee x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4 \vee x_6) \\ \wedge &(x_2 \vee x_3 \vee x_5 \vee x_6) \wedge (x_2 \vee x_4 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_5 \vee x_6) \end{aligned}$$

- And pseudo-Boolean reasoning exponentially stronger than conflict-driven clause learning (CDCL)
- Yet close enough to SAT to benefit from SAT solving advances

Pseudo-Boolean vs. SAT

- Pseudo-Boolean format richer than conjunctive normal form (CNF)

Compare

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

and

$$\begin{aligned}
&(x_1 \lor x_2 \lor x_3 \lor x_4) \land (x_1 \lor x_2 \lor x_3 \lor x_5) \land (x_1 \lor x_2 \lor x_3 \lor x_6) \\
\land &(x_1 \lor x_2 \lor x_4 \lor x_5) \land (x_1 \lor x_2 \lor x_4 \lor x_6) \land (x_1 \lor x_2 \lor x_5 \lor x_6) \\
\land &(x_1 \lor x_3 \lor x_4 \lor x_5) \land (x_1 \lor x_3 \lor x_4 \lor x_6) \land (x_1 \lor x_3 \lor x_4 \lor x_6) \\
\land &(x_1 \lor x_4 \lor x_5 \lor x_6) \land (x_2 \lor x_3 \lor x_4 \lor x_5) \land (x_2 \lor x_3 \lor x_4 \lor x_6) \\
\land &(x_2 \lor x_3 \lor x_5 \lor x_6) \land (x_2 \lor x_4 \lor x_5 \lor x_6) \land (x_3 \lor x_4 \lor x_5 \lor x_6)
\end{aligned}$$

- And pseudo-Boolean reasoning exponentially stronger than conflict-driven clause learning (CDCL)
- Yet close enough to SAT to benefit from SAT solving advances
- Also possible synergies with 0-1 integer linear programming (ILP)

Pseudo-Boolean Constraints and Normalized Form

In this talk, pseudo-Boolean constraints are 0-1 integer linear constraints

$$\sum_i a_i \ell_i \bowtie A$$

- $\bowtie \in \{\geq, \leq, =, >, <\}$
- $a_i, A \in \mathbb{Z}$
- literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)
- variables x_i take values $0 = \mathit{false}$ or $1 = \mathit{true}$

Pseudo-Boolean Constraints and Normalized Form

In this talk, pseudo-Boolean constraints are 0-1 integer linear constraints

$$\sum_i a_i \ell_i \bowtie A$$

- $\bowtie \in \{\geq, \leq, =, >, <\}$
- $a_i, A \in \mathbb{Z}$
- literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)
- variables x_i take values $0 = \mathit{false}$ or $1 = \mathit{true}$

Convenient to use normalized form [Bar95] (without loss of generality)

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

- constraint always greater-than-or-equal
- $a_i, A \in \mathbb{N}$
- $A = \mathit{deg}(\sum_i a_i \ell_i \geq A)$ referred to as degree (of falsity)

Some Types of Pseudo-Boolean Constraints

1. Clauses are pseudo-Boolean constraints

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

Some Types of Pseudo-Boolean Constraints

1. Clauses are pseudo-Boolean constraints

$$x \lor \overline{y} \lor z \quad \Leftrightarrow \quad x + \overline{y} + z \geq 1$$

2. Cardinality constraints

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

Some Types of Pseudo-Boolean Constraints

1. Clauses are pseudo-Boolean constraints

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

2. Cardinality constraints

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

3. General constraints

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

Some Types of Pseudo-Boolean Constraints

1. Clauses are pseudo-Boolean constraints

$$x \lor \overline{y} \lor z \quad \Leftrightarrow \quad x + \overline{y} + z \geq 1$$

2. Cardinality constraints

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

3. General constraints

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

4. Reified constraints encoding $z \Leftrightarrow x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$

$$7\overline{z} + x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$$
$$9z + \overline{x}_1 + 2x_2 + 3\overline{x}_3 + 4x_4 + 5\overline{x}_5 \geq 9$$

Formulas, Decision Problems, and Optimization Problems

Pseudo-Boolean (PB) formula
Conjunction of pseudo-Boolean constraints
$F \doteq C_1 \wedge C_2 \wedge \cdots \wedge C_m$

Pseudo-Boolean Solving (PBS)
Decide whether F is satisfiable/feasible

Pseudo-Boolean Optimization (PBO)
Find satisfying assignment to F that minimizes objective function $\sum_i w_i \ell_i$
(Maximization: minimize $-\sum_i w_i \ell_i$)

Approaches for Pseudo-Boolean Problems

1. Pseudo-Boolean (PB) solving and optimization [main focus]
2. MaxSAT solving
3. Integer linear programming (ILP) — or, more generally, mixed integer linear programming (MIP)

Two Approaches to Pseudo-Boolean Solving

Re-encode to CNF and run conflict-driven clause learning (CDCL)

- MiniSat+ [ES06]
- Open-WBO [MML14]
- NaPS [SN15]

Two Approaches to Pseudo-Boolean Solving

Re-encode to CNF and run conflict-driven clause learning (CDCL)

- MiniSat+ [ES06]
- Open-WBO [MML14]
- NaPS [SN15]

Native reasoning with pseudo-Boolean constraints

- PRS [DG02]
- Galena [CK05]
- Pueblo [SS06]
- Sat4j [LP10]
- RoundingSat [EN18]

Performance of CDCL-Based Pseudo-Boolean Solving

- CDCL-based pseudo-Boolean can be very competitive (sometimes beating native pseudo-Boolean solvers hands down)
- Extension variables potentially gives solver lots of power
  - Allows branching over complex statements
  - Can learn clauses corresponding to polytopes in original problem
- But performance gain from extension variables seems quite sensitive to input order [EGNV18]
- And sometimes extension variables cannot make up for CDCL being exponentially weaker than pseudo-Boolean reasoning [EGNV18]

Some Research Questions

1. How to find best CNF encodings of PB constraints for given problem?
   - Trade-offs between propagation strength and encoding size?
   - Rigorous mathematical insights?

2. How do CDCL-based and "native" cutting-planes-based PB solving approaches compare?
   - Theoretical results on computational complexity?
   - Harness complementary strengths in applied solvers?

"Native" Pseudo-Boolean Conflict-Driven Search

Want to do "same thing" as in conflict-driven clause learning (CDCL) SAT solving [MS96, BS97, MMZ⁺01]

But with cutting planes reasoning on PB constraints without re-encoding

- Variable assignments
  1. Always propagate forced assignment if possible
  2. Otherwise make assignment using decision heuristic
- At conflict
  1. Do conflict analysis to derive new constraint
  2. Add new constraint to constraint database and backjump

The Cutting Planes Proof System [CCT87, CK05]

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i) \ell_i \geq c_A A + c_B B}$$

Division $$\frac{\sum_i a_i \ell_i \geq A}{\sum_i \lceil a_i / c \rceil \ell_i \geq \lceil A/c \rceil}$$

Saturation $$\frac{\sum_i a_i \ell_i \geq A}{\sum_i \min\{a_i, A\} \cdot \ell_i \geq A}$$

The Cutting Planes Proof System [CCT87, CK05]

Literal axioms $\dfrac{}{\ell_i \geq 0}$

Linear combination $\dfrac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B}$

Division $\dfrac{\sum_i a_i \ell_i \geq A}{\sum_i \lceil a_i/c \rceil \ell_i \geq \lceil A/c \rceil}$

Saturation $\dfrac{\sum_i a_i \ell_i \geq A}{\sum_i \min\{a_i, A\} \cdot \ell_i \geq A}$

Division by 2 $\dfrac{x + 2y + 4z \geq 3}{x + y + 2z \geq 2}$ Saturation $\dfrac{x + 2y + 4z \geq 3}{x + 2y + 3z \geq 3}$

Some PB Solving Challenges I: Input Format

1. CNF: PB solvers degenerate to CDCL for CNF inputs — how to harness power of cutting planes in this setting?
   - Cardinality constraint detection proposed as preprocessing [BLLM14] or inprocessing [EN20]
   - Not yet competitive in practice
2. Linear programming: Sometimes very poor performance even on infeasible 0-1 LPs!
   - Unclear why
   - Very easy for cutting planes in theory
3. Preprocessing/presolving: Important in SAT solving and integer linear programming, but not done in PB solvers — why?
   - Follow up on preliminary work on PB preprocessing in [MLM09]?
   - Use presolver PaPILO [PaP] from MIP solver SCIP [SCI]?

Some PB Solving Challenges II: Conflict Analysis

1. Many more degrees of freedom than in CDCL, e.g.:
   - Choice of Boolean rule (division, saturation, or combination?)
   - Learn general PB constraints or more limited form?
   - How far to backjump when learned constraint asserting at many levels?
   - How large precision to use in integer arithmetic?
2. How to assess quality of learned constraints?
3. Theoretical potential and limitations poorly understood [VEG$^+$18]
   - Separations of subsystems of cutting planes?
   - In particular, is division reasoning stronger than saturation? [GNY19]

Linear Search SAT-UNSAT (LSU) Algorithm

- Minimize $\sum_{i=1}^{n} w_i \ell_i$
- Subject to collection of PB constraints $F = C_1 \wedge \cdots \wedge C_m$

Set $\rho_{\text{best}} = \emptyset$ and repeat the following:

1. Run SAT/PB solver
2. If solver returns `UNSATISFIABLE`, output ρ_{best} and terminate
3. Otherwise, let $\rho_{\text{best}} :=$ returned solution ρ
4. Add constraint $\sum_{i=1}^{n} w_i \ell_i \leq -1 + \sum_{i=1}^{n} w_i \cdot \rho(\ell_i)$
5. Start over from the top

More on Linear Search

Properties of linear search SAT-UNSAT:

- Can get **some decent** solution quickly, even if not optimal one
- Important for anytime solving (when time is limited and something is better than nothing)
- But get no estimate of how good the solution is

Core-Guided Pseudo-Boolean Search

- Minimize $\sum_{i=1}^{n} w_i \ell_i$
- Subject to collection of PB constraints $F = C_1 \wedge \cdots \wedge C_m$

Core-guided PB search: assume optimistically that objective can reach best imaginable value; derive contradiction if not possible

Core-Guided Pseudo-Boolean Search

- Minimize $\sum_{i=1}^{n} w_i \ell_i$
- Subject to collection of PB constraints $F = C_1 \land \cdots \land C_m$

Core-guided PB search: assume optimistically that objective can reach best imaginable value; derive contradiction if not possible

Set $val_{\text{best}} = 0$ and repeat the following:

1. Run pseudo-Boolean solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function

Core-Guided Pseudo-Boolean Search

- Minimize $\sum_{i=1}^{n} w_i \ell_i$
- Subject to collection of PB constraints $F = C_1 \land \cdots \land C_m$

Core-guided PB search: assume optimistically that objective can reach best imaginable value; derive contradiction if not possible

Set $val_{\mathrm{best}} = 0$ and repeat the following:

1. Run pseudo-Boolean solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns SATISFIABLE, output val_{best} and terminate

Core-Guided Pseudo-Boolean Search

- Minimize $\sum_{i=1}^{n} w_i \ell_i$
- Subject to collection of PB constraints $F = C_1 \wedge \cdots \wedge C_m$

Core-guided PB search: assume optimistically that objective can reach best imaginable value; derive contradiction if not possible

Set $val_{\mathrm{best}} = 0$ and repeat the following:

1. Run pseudo-Boolean solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn constraint $\sum_{i=1}^{k} \ell_i \geq A$ over assumption variables

Core-Guided Pseudo-Boolean Search

- Minimize $\sum_{i=1}^{n} w_i \ell_i$
- Subject to collection of PB constraints $F = C_1 \wedge \cdots \wedge C_m$

Core-guided PB search: assume optimistically that objective can reach best imaginable value; derive contradiction if not possible

Set $val_{\text{best}} = 0$ and repeat the following:

1. Run pseudo-Boolean solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn constraint $\sum_{i=1}^{k} \ell_i \geq A$ over assumption variables
4. Update val_{best} and rewrite objective function using new variables $z_j \Leftrightarrow \sum_{i=1}^{k} \ell_i \geq j$

Core-Guided Pseudo-Boolean Search

- Minimize $\sum_{i=1}^{n} w_i \ell_i$
- Subject to collection of PB constraints $F = C_1 \wedge \cdots \wedge C_m$

Core-guided PB search: assume optimistically that objective can reach best imaginable value; derive contradiction if not possible

Set $val_{\mathrm{best}} = 0$ and repeat the following:

1. Run pseudo-Boolean solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn constraint $\sum_{i=1}^{k} \ell_i \geq A$ over assumption variables
4. Update val_{best} and rewrite objective function using new variables $z_j \Leftrightarrow \sum_{i=1}^{k} \ell_i \geq j$
5. Start over from top (with modified objective function)

Properties of (Pure) Core-Guided Search

- Can get decent lower bounds on solution quickly
- Helps to cut off parts of search space that are "too good to be true"
- But find no actual solution until the final, optimal one
- Also, no estimate of how good the lower bound is
- Linear search much better at finding solutions — so try to get the best of both worlds by combining the two!

Evaluation of Core-Guided PB Solver in [DGD$^+$21]

RoundingSat variants with core-guided (CG) and linear search (LSU)

#instances solved to optimality; highlighting **1st**, **2nd**, and **3rd** best

Evaluation of Core-Guided PB Solver in [DGD+21]

RoundingSat variants with core-guided (CG) and linear search (LSU)

#instances solved to optimality; highlighting **1st**, **2nd**, and **3rd** best

	PB16opt (1600)	MIPopt (291)	KNAP (783)	CRAFT (985)
Hybrid (interleave CG & LSU)	**968**	**78**	306	**639**
HybridCl (w/ clausal cores)	937	75	298	**618**
HybridNL (w/ non-lazy variables)	936	70	186	607
HybridClNL (w/ both)	917	67	203	612
RoundingSat (only LSU)	853	75	341	309
CoreGuided (only CG)	911	61	43	595
CoreBoosted (10% CG, then LSU)	**959**	**80**	**344**	580
Sat4j	773	61	**373**	105
NaPS	896	65	111	345
SCIP	**1057**	**125**	**765**	**642**

Evaluation of Core-Guided PB Solver in [DGD+21]

RoundingSat variants with core-guided (CG) and linear search (LSU)

#instances solved to optimality; highlighting **1st**, **2nd**, and **3rd** best

	PB16opt (1600)	MIPopt (291)	KNAP (783)	CRAFT (985)
Hybrid (interleave CG & LSU)	**968**	**78**	306	**639**
HybridCl (w/ clausal cores)	937	75	298	**618**
HybridNL (w/ non-lazy variables)	936	70	186	607
HybridClNL (w/ both)	917	67	203	612
RoundingSat (only LSU)	853	75	341	309
CoreGuided (only CG)	911	61	43	595
CoreBoosted (10% CG, then LSU)	**959**	**80**	**344**	580
Sat4j	773	61	**373**	105
NaPS	896	65	111	345
SCIP	**1057**	**125**	**765**	**642**

Significant improvement over PB state of the art, but MIP still better

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 best for optimization

MIP Solving at a High Level

1. Preprocessing (called presolving)
2. Linear programming relaxations + 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

Combining PB 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 PB 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 PB 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 SAT-style conflict-driven search and MIP-style branch-and-cut?

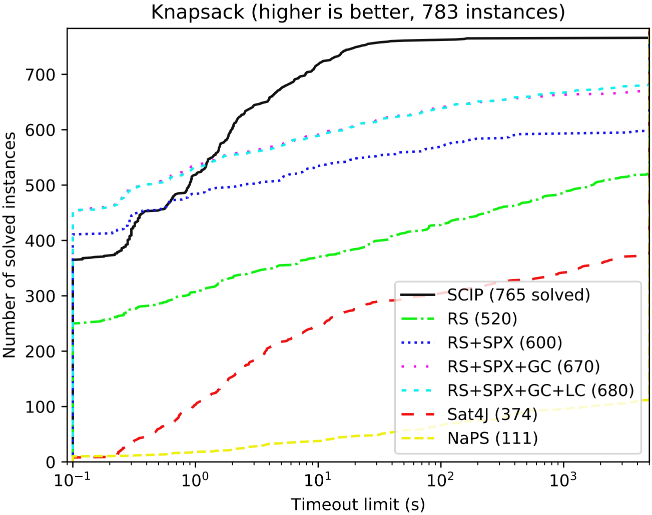
Experimental Results for Knapsack Benchmarks [Pis05]

ROUNDINGSAT (RS) enhanced with

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

as in [DGN21]

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)

as in [DGN21] 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)

as in [DGN21] 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

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)

as in [DGN21] 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

Hybrid PB-LP solver well-rounded — always competitive with best solver

Some Future Research Directions for PB-LP Integration

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?

Some Future Research Directions for PB-LP Integration

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. Make more intelligent use in PB solver of information from solutions to LP relaxations

Some Future Research Directions for PB-LP Integration

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. Make more intelligent use in PB solver of information from solutions to LP relaxations
3. Use MIP presolving in pseudo-Boolean solvers

Some Future Research Directions for PB-LP Integration

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. Make more intelligent use in PB solver of information from solutions to LP relaxations
3. Use MIP presolving in pseudo-Boolean solvers
4. Use MIP cut rules to improve pseudo-Boolean conflict analysis

Balancing the Picture

Cutting-planes-based pseudo-Boolean solvers sometimes outperform even commercial MIP solvers by orders of magnitude:

- Arithmetic circuit verification [LBD^{+}20]
- Matching of children with adoptive families (compared to [DGG^{+}19])
- Automated planning using neural networks (compared to [SS18], see also [SDNS20] — reified constraints hard for MIP)

Summing up

- Pseudo-Boolean optimization powerful and expressive framework
- Can be attacked with methods from
  - SAT solving and MaxSAT solving
  - "Native" cutting-planes-based pseudo-Boolean reasoning
  - Mixed integer linear programming
- Approaches with complementary strengths — room for synergies?
- For cutting-planes-based reasoning, challenges regarding
  - Algorithm design
  - Efficient implementation
  - Theoretical understanding
- But cutting-planes-based solvers sometimes very powerful — worth trying out if you have a MaxSAT/PB optimization problem!

Summing up

- Pseudo-Boolean optimization powerful and expressive framework
- Can be attacked with methods from
  - SAT solving and MaxSAT solving
  - "Native" cutting-planes-based pseudo-Boolean reasoning
  - Mixed integer linear programming
- Approaches with complementary strengths — room for synergies?
- For cutting-planes-based reasoning, challenges regarding
  - Algorithm design
  - Efficient implementation
  - Theoretical understanding
- But cutting-planes-based solvers sometimes very powerful — worth trying out if you have a MaxSAT/PB optimization problem!

Thank you for your attention!

References I

[Bar95] Peter Barth. A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization. Technical Report MPI-I-95-2-003, Max-Planck-Institut für Informatik, January 1995.

[BH02] Endre Boros and Peter L. Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1–3):155–225, November 2002.

[BLLM14] Armin Biere, Daniel Le Berre, Emmanuel Lonca, and Norbert Manthey. Detecting cardinality constraints in CNF. 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 285–301. Springer, July 2014.

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[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.

[CK05] Donald Chai and Andreas Kuehlmann. A fast pseudo-Boolean constraint solver. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(3):305–317, March 2005. Preliminary version in *DAC '03*.

References II

[DG02] Heidi E. Dixon and Matthew L. Ginsberg. Inference methods for a pseudo-Boolean satisfiability solver. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*, pages 635–640, July 2002.

[DGD+21] Jo Devriendt, Stephan Gocht, Emir Demirović, Jakob Nordström, and Peter Stuckey. Cutting to the core of pseudo-Boolean optimization: Combining core-guided search with cutting planes reasoning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*, February 2021. To appear.

[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*, January 2021. Preliminary version in *CPAIOR '20*.

References III

[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.

[EN18] Jan Elffers and Jakob Nordström. Divide and conquer: Towards faster pseudo-Boolean solving. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*, pages 1291–1299, July 2018.

[EN20] Jan Elffers and Jakob Nordström. A cardinal improvement to pseudo-Boolean solving. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*, pages 1495–1503, 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.

[GNY19] Stephan Gocht, Jakob Nordström, and Amir Yehudayoff. On division versus saturation in pseudo-Boolean solving. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI '19)*, pages 1711–1718, August 2019.

References IV

[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.

[LBD+20] Vincent Liew, Paul Beame, Jo Devriendt, Jan Elffers, and Jakob Nordström. Verifying properties of bit-vector multiplication using cutting planes reasoning. In *Proceedings of the 20th Conference on Formal Methods in Computer-Aided Design (FMCAD '20)*, pages 194–204, September 2020.

[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.

[MLM09] Ruben Martins, Inês Lynce, and Vasco M. Manquinho. Preprocessing in pseudo-Boolean optimization: An experimental evaluation. In *Proceedings of the 8th International Workshop on Constraint Modelling and Reformulation (ModRef '09)*, pages 87–101, September 2009. Available at `https://www-users.cs.york.ac.uk/~frisch/ModRef/09/proceedings.pdf`.

[MML14] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A modular MaxSAT solver. 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 438–445. Springer, July 2014.

References V

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[MS96] João P. Marques-Silva and Karem A. Sakallah. GRASP—a new search algorithm for satisfiability. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '96)*, pages 220–227, November 1996.

[PaP] PaPILO — parallel presolve for integer and linear optimization. `https://github.com/lgottwald/PaPILO`.

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

[SCI] SCIP: Solving constraint integer programs. `http://scip.zib.de/`.

[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.

References VI

[SN15] Masahiko Sakai and Hidetomo Nabeshima. Construction of an ROBDD for a PB-constraint in band form and related techniques for PB-solvers. *IEICE Transactions on Information and Systems*, 98-D(6):1121–1127, June 2015.

[SS06] Hossein M. Sheini and Karem A. Sakallah. Pueblo: A hybrid pseudo-Boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):165–189, March 2006. Preliminary version in *DATE '05*.

[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.

[VEG+18] Marc Vinyals, Jan Elffers, Jesús Giráldez-Cru, Stephan Gocht, and Jakob Nordström. In between resolution and cutting planes: A study of proof systems for pseudo-Boolean SAT solving. 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 292–310. Springer, July 2018.