Pseudo-Boolean Solving and Optimization

Jakob Nordström

University of Copenhagen
and Lund University

"Satisfiability: Theory, Practice, and Beyond" Boot Camp
Simons Institute for the Theory of Computing
February 4, 2021

Organization of This Tutorial

Part I: Pseudo-Boolean Preliminaries

Part II: Pseudo-Boolean Solving

Part III: Pseudo-Boolean Optimization

Part IV: Mixed Integer Linear Programming

Organization of This Tutorial

Part I: Pseudo-Boolean Preliminaries

Part II: Pseudo-Boolean Solving

Part III: Pseudo-Boolean Optimization

Part IV: Mixed Integer Linear Programming

Outline of Part I: Pseudo-Boolean Preliminaries

1. Pseudo-Boolean Functions and Constraints
2. Pseudo-Boolean Solving and Optimization
3. Some Further References

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 versions:

- f represented as polynomial
- f represented as linear form [focus of this tutorial]

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

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

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

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

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

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]

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

- 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

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

2. Cardinality constraints

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

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

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

Conversion to Normalized Form: Example

Normalized form used for convenience and without loss of generality

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 < 0$$

Conversion to Normalized Form: Example

Normalized form used for convenience and without loss of generality

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 < 0$$

1. Make inequality non-strict

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 \leq -1$$

Conversion to Normalized Form: Example

Normalized form used for convenience and without loss of generality

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 < 0$$

1. Make inequality non-strict

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 \leq -1$$

2. Multiply by -1 to get greater-than-or-equal

$$x_1 - 2x_2 + 3x_3 - 4x_4 + 5x_5 \geq 1$$

Conversion to Normalized Form: Example

Normalized form used for convenience and without loss of generality

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 < 0$$

1. Make inequality non-strict

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 \leq -1$$

2. Multiply by -1 to get greater-than-or-equal

$$x_1 - 2x_2 + 3x_3 - 4x_4 + 5x_5 \geq 1$$

3. Replace $-\ell$ by $-(1-\overline{\ell})$ [where we define $\overline{\overline{x}} \doteq x$]

$$\begin{aligned} x_1 - 2(1-\overline{x}_2) + 3x_3 - 4(1-\overline{x}_4) + 5x_5 &\geq 1 \\ x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 &\geq 7 \end{aligned}$$

Conversion to Normalized Form: Example

Normalized form used for convenience and without loss of generality

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 < 0$$

1. Make inequality non-strict

$$-x_1 + 2x_2 - 3x_3 + 4x_4 - 5x_5 \leq -1$$

2. Multiply by -1 to get greater-than-or-equal

$$x_1 - 2x_2 + 3x_3 - 4x_4 + 5x_5 \geq 1$$

3. Replace $-\ell$ by $-(1-\overline{\ell})$ [where we define $\overline{\overline{x}} \doteq x$]

$$x_1 - 2(1-\overline{x}_2) + 3x_3 - 4(1-\overline{x}_4) + 5x_5 \geq 1$$
$$x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$$

4. Replace "$=$" by two inequalities "$\geq$" and "$\leq$"

Conversion to Normalized Form: Formal Details

Given linear form $\sum_i a_i \ell_i$ with $\sum_i a_i = M$

Syntactic sugar	**Meaning**
$\sum_i a_i \ell_i > A$	$\sum_i a_i \ell_i \geq A + 1$
$\sum_i a_i \ell_i \leq A$	$\sum_i a_i \overline{\ell}_i \geq M - A$
$\sum_i a_i \ell_i < A$	$\sum_i a_i \overline{\ell}_i \geq M - A + 1$
$\sum_i a_i \ell_i = A$	$\sum_i a_i \ell_i \geq A$ and $\sum_i a_i \overline{\ell}_i \geq M - A$

In what follows:

- Use syntactic sugar when convenient
- Assume (implicit) normalization whenever it matters

Linearization

Possible to linearize nonlinear constraints

$$\sum_{i=1}^{k} a_i m_i \geq A$$

with

$$m_i \doteq \prod_{j=1}^{d_i} \ell_{i,j}$$

Linearization

Possible to linearize nonlinear constraints

$$\sum_{i=1}^{k} a_i m_i \geq A$$

with

$$m_i \doteq \prod_{j=1}^{d_i} \ell_{i,j}$$

For instance, using fresh variables y_i we can write:

$$\begin{aligned} \sum_{i=1}^{k} a_i y_i &\geq A & \\ d_i \cdot \overline{y}_i + \sum_{j=1}^{d_i} \ell_{i,j} &\geq d_i & i \in [k] \\ y_i + \sum_{j=1}^{d_i} \overline{\ell}_{i,j} &\geq 1 & i \in [k] \end{aligned}$$

Linearization

Possible to linearize nonlinear constraints

$$\sum_{i=1}^{k} a_i m_i \geq A$$

with

$$m_i \doteq \prod_{j=1}^{d_i} \ell_{i,j}$$

For instance, using fresh variables y_i we can write:

$$\begin{aligned} \sum_{i=1}^{k} a_i y_i &\geq A & \\ d_i \cdot \overline{y}_i + \sum_{j=1}^{d_i} \ell_{i,j} &\geq d_i & i \in [k] \\ y_i + \sum_{j=1}^{d_i} \overline{\ell}_{i,j} &\geq 1 & i \in [k] \end{aligned}$$

We won't go further into this during this talk, though...

Some Notation for Operations on Constraints (1/2)

Given

- constraints $C_1 \doteq \sum_i a_i \ell_i \geq A$ and $C_2 \doteq \sum_i b_i \ell_i \geq B$
- linear form $L \doteq \sum_i c \ell_i$
- positive integer $k \in \mathbb{N}^+$

we will use notation:

$$C_1 + C_2 \doteq \sum_i (a_i + b_i) \cdot \ell_i \geq A + B$$

$$C_1 + L \doteq \sum_i (a_i + c_i) \cdot \ell_i \geq A$$

$$k \cdot C_1 \doteq \sum_i k a_i \cdot \ell_i \geq kA$$

(assuming appropriate normalization whenever needed)

Some Notation for Operations on Constraints (2/2)

Given constraint $C \doteq \sum_i a_i \ell_i \geq A$ with $\sum_i a_i = M$

Negation

$$\neg C \doteq \sum_i a_i \overline{\ell}_i \geq M - A + 1$$

Reification

$$z \Rightarrow C \doteq A \cdot \overline{z} + \sum_i a_i \ell_i \geq A$$

$$z \Leftarrow C \doteq (M - A + 1) \cdot z + \sum_i a_i \overline{\ell}_i \geq M - A + 1$$

$$z \Leftrightarrow C \doteq z \Rightarrow C \text{ and } z \Leftarrow C$$

Some calculations

$$C + \neg C \doteq 0 \geq 1$$

$$z \Leftarrow C \doteq \overline{z} \Rightarrow \neg C$$

$$deg(C) \cdot (z \geq 1) + (z \Rightarrow C) \doteq C$$

$$C + (z \Leftarrow C) \doteq deg(\neg C) \cdot z \geq 1$$

Formulas, Decision Problems, and Optimization Problems

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

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 $-\sum_i w_i \ell_i$)

Some Problems Expressed as PBO (1/2)

Input:

- undirected graph $G = (V, E)$
- weight function $w : V \to \mathbb{N}^+$

Weighted minimum vertex cover

$$\min \sum_{v \in V} w(v) \cdot x_v$$

$$x_u + x_v \geq 1 \qquad (u, v) \in E$$

Weighted maximum clique

$$\min -\sum_{v \in V} w(v) \cdot x_v$$

$$\overline{x}_u + \overline{x}_v \geq 1 \qquad (u, v) \notin E$$

Some Problems Expressed as PBO (2/2)

Input:

- sets $S_1, \ldots, S_m \subseteq \mathcal{U}$
- weight function $w : \mathcal{U} \to \mathbb{N}^+$

Weighted minimum hitting set

Find $H \subseteq \mathcal{U}$ such that

- $H \cap S_i \neq \emptyset$ for all $i \in [m]$ (H is a hitting set)
- $\sum_{h \in H} w(h)$ is minimal

Some Problems Expressed as PBO (2/2)

Input:

- sets $S_1, \ldots, S_m \subseteq \mathcal{U}$
- weight function $w : \mathcal{U} \to \mathbb{N}^+$

Weighted minimum hitting set

Find $H \subseteq \mathcal{U}$ such that

- $H \cap S_i \neq \emptyset$ for all $i \in [m]$ (H is a hitting set)
- $\sum_{h \in H} w(h)$ is minimal

$$\begin{aligned} &\min \ \textstyle\sum_{e \in \mathcal{U}} w(e) \cdot x_e \\ &\textstyle\sum_{e \in S_i} x_e \geq 1 \qquad\qquad i \in [m] \end{aligned}$$

Some Problems Expressed as PBO (2/2)

Input:

- sets $S_1, \ldots, S_m \subseteq \mathcal{U}$
- weight function $w : \mathcal{U} \to \mathbb{N}^+$

Weighted minimum hitting set

Find $H \subseteq \mathcal{U}$ such that

- $H \cap S_i \neq \emptyset$ for all $i \in [m]$ (H is a hitting set)
- $\sum_{h \in H} w(h)$ is minimal

$$\begin{aligned} &\min \; \textstyle\sum_{e \in \mathcal{U}} w(e) \cdot x_e \\ &\textstyle\sum_{e \in S_i} x_e \geq 1 \qquad\qquad\qquad\qquad i \in [m] \end{aligned}$$

Note: In all of these examples, the problem is to

- optimize a linear function
- subject to a CNF formula (all constraints are clausal)

Already expressive framework!

Approaches for Pseudo-Boolean Problems

What we will discuss in this tutorial:

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)

Approaches for Pseudo-Boolean Problems

What we will discuss in this tutorial:

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)

Rough conceptual difference:

- **PB/SAT:** Focus on integral solutions, try to find optimal one
- **ILP/MIP:** Find optimal non-integer solution; search for integral solutions nearby

Basic trade-off: Inference power vs. inference speed

Some References for Further Reading (and Watching)

Handbook of Satisfiability (PB and MaxSAT)

- Chapter 7: Proof Complexity and SAT Solving
- Chapter 23: MaxSAT, Hard and Soft Constraints
- Chapter 24: Maximum Satisfiability
- Chapter 28: Pseudo-Boolean and Cardinality Constraints

Mixed integer linear programming

- `https://tinyurl.com/MIPsurveypaper` [Wol08]
- `https://tinyurl.com/MIPperformance` [KMP13]

Videos

- MaxSAT tutorial by Berg et al. `https://tinyurl.com/MaxSATtutorial`
- MIP tutorial by Gleixner `https://tinyurl.com/MIPtutorial`

Organization of This Tutorial

Part I: Pseudo-Boolean Preliminaries

Part II: Pseudo-Boolean Solving

Part III: Pseudo-Boolean Optimization

Part IV: Mixed Integer Linear Programming

Outline of Part II: Pseudo-Boolean Solving

A Quick Recap of Modern SAT Solving

DPLL method [DP60, DLL62]

- Assign values to variables (in some smart way)
- Backtrack when conflict with falsified clause

A Quick Recap of Modern SAT Solving

DPLL method [DP60, DLL62]

- Assign values to variables (in some smart way)
- Backtrack when conflict with falsified clause

Conflict-driven clause learning (CDCL) [MS96, BS97, MMZ+01]

- Analyse conflicts in more detail — add new clauses to formula
- More efficient backtracking
- Also let conflicts guide other heuristics

A Quick Recap of Modern SAT Solving

DPLL method [DP60, DLL62]

- Assign values to variables (in some smart way)
- Backtrack when conflict with falsified clause

Conflict-driven clause learning (CDCL) [MS96, BS97, MMZ$^+$01]

- Analyse conflicts in more detail — add new clauses to formula
- More efficient backtracking
- Also let conflicts guide other heuristics

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$$

$w \stackrel{\mathsf{d}}{=} 0$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \overset{\mathsf{d}}{=} 0}$

Decision

Free choice to assign value to variable

Notation $w \overset{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \overset{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$w \overset{\mathsf{d}}{=} 0$

$u \overset{\overline{u} \lor w}{=} 0$

Decision

Free choice to assign value to variable

Notation $w \overset{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \overset{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \vee w}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \vee w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \vee w}{=} 0$ ($\overline{u} \vee w$ is reason)

Always propagate if possible, otherwise decide

Until satisfying assignment or conflict clause

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Always propagate if possible, otherwise decide

Until satisfying assignment or conflict clause

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

$\boxed{z \stackrel{x \lor \overline{y} \lor z}{=} 1}$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Always propagate if possible, otherwise decide

Until satisfying assignment or conflict clause

Variable Assignments

Two kinds of assignments — illustrate on example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

$\boxed{z \stackrel{x \lor \overline{y} \lor z}{=} 1}$

$\boxed{\stackrel{\overline{y} \lor \overline{z}}{\bot}}$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Always propagate if possible, otherwise decide

Until satisfying assignment or conflict clause

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

$\boxed{z \stackrel{x \lor \overline{y} \lor z}{=} 1}$

$\boxed{\stackrel{\overline{y} \lor \overline{z}}{\bot}}$

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \lor w}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

$y \stackrel{u \lor x \lor y}{=} 1$

$z \stackrel{x \lor \overline{y} \lor z}{=} 1$

$\stackrel{\overline{y} \lor \overline{z}}{\bot}$

Could backtrack by flipping last decision

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \lor w}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

$y \stackrel{u \lor x \lor y}{=} 1$

$z \stackrel{x \lor \overline{y} \lor z}{=} 1$

$\stackrel{\overline{y} \lor \overline{z}}{\bot}$

Could backtrack by flipping last decision

But want to learn from conflict and cut away as much of search space as possible

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$w \overset{\mathsf{d}}{=} 0$

$u \overset{\overline{u} \lor w}{=} 0$

$x \overset{\mathsf{d}}{=} 0$

$y \overset{u \lor x \lor y}{=} 1$

$z \overset{x \lor \overline{y} \lor z}{=} 1$ $\rightarrow$ $x \lor \overline{y}$

$\overset{\overline{y} \lor \overline{z}}{\bot}$ $\rightarrow$ $x \lor \overline{y}$

Could backtrack by flipping last decision

But want to learn from conflict and cut away as much of search space as possible

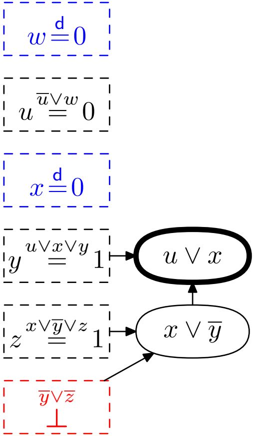
Case analysis over z for last two clauses:

- $x \lor \overline{y} \lor z$ wants $z = 1$
- $\overline{y} \lor \overline{z}$ wants $z = 0$
- Merge & remove z — must satisfy $x \lor \overline{y}$

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Could backtrack by flipping last decision

But want to learn from conflict and cut away as much of search space as possible

Case analysis over z for last two clauses:

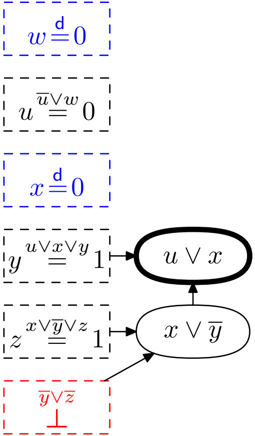
- $x \lor \overline{y} \lor z$ wants $z = 1$
- $\overline{y} \lor \overline{z}$ wants $z = 0$
- Merge & remove z — must satisfy $x \lor \overline{y}$

Repeat until only 1 variable after last decision — learn that clause (1UIP) and backjump

Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

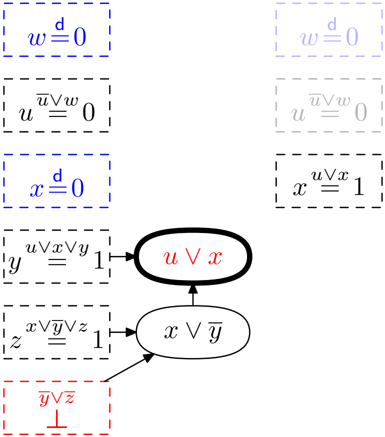
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

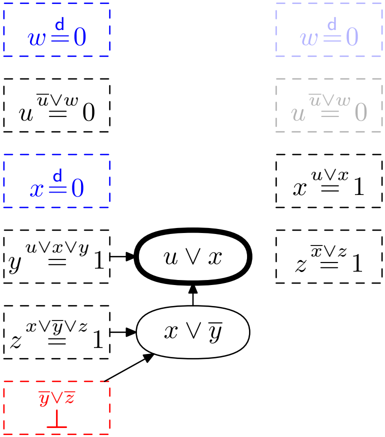
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

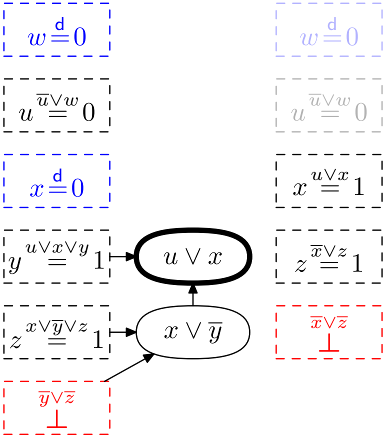
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

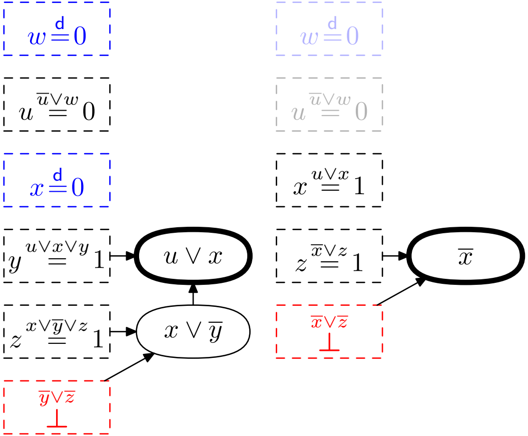
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

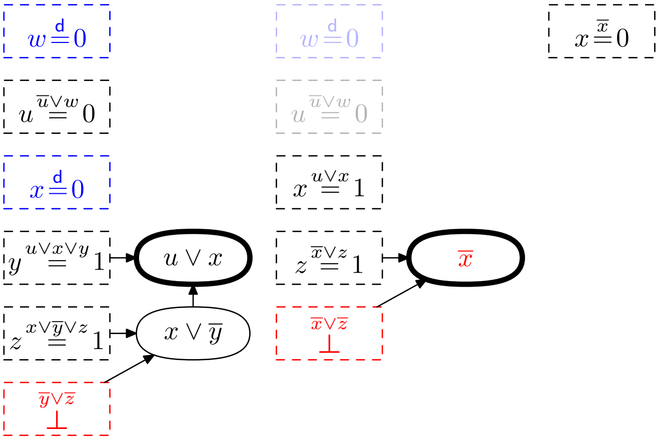
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

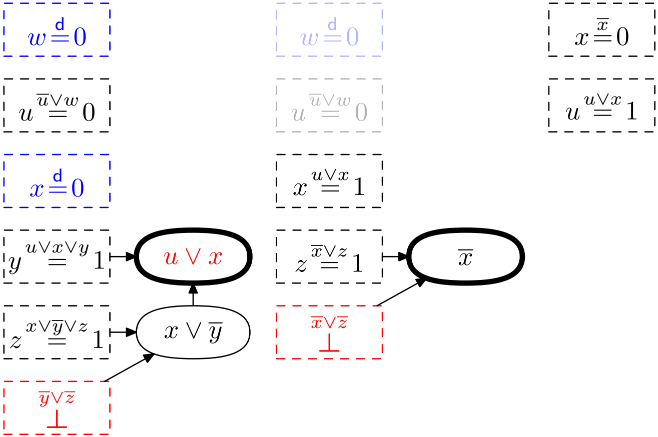
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

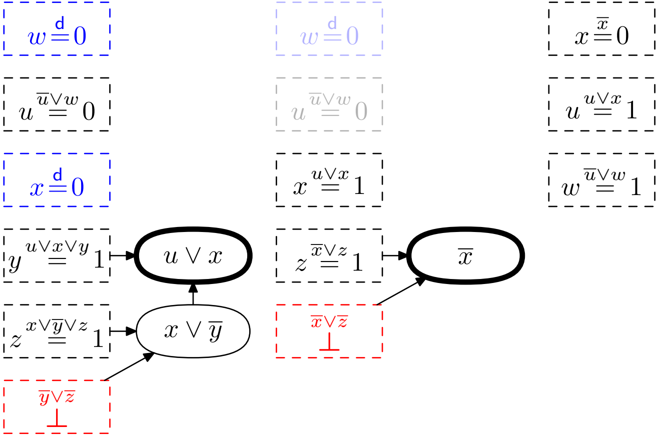
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

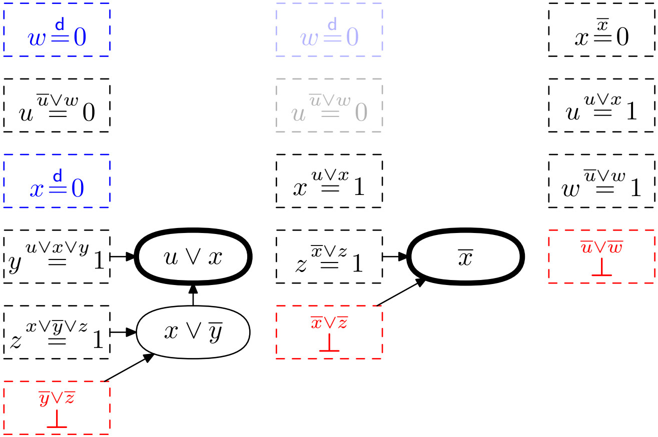
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

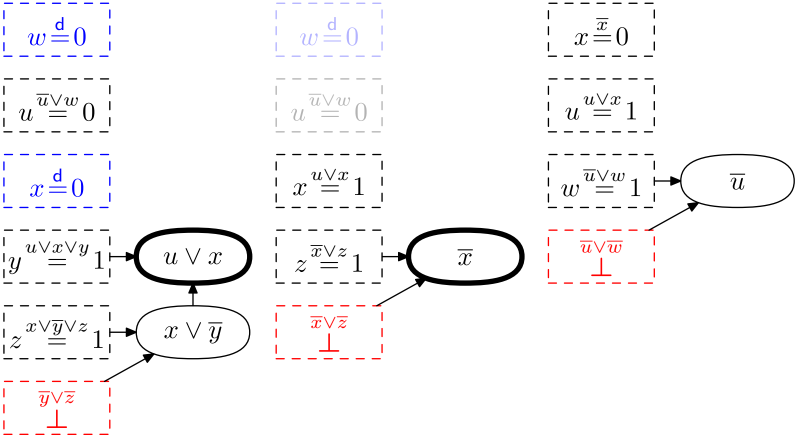
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

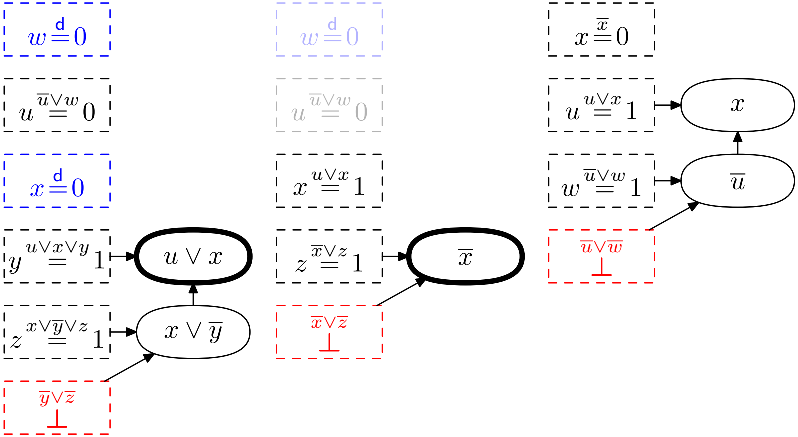
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

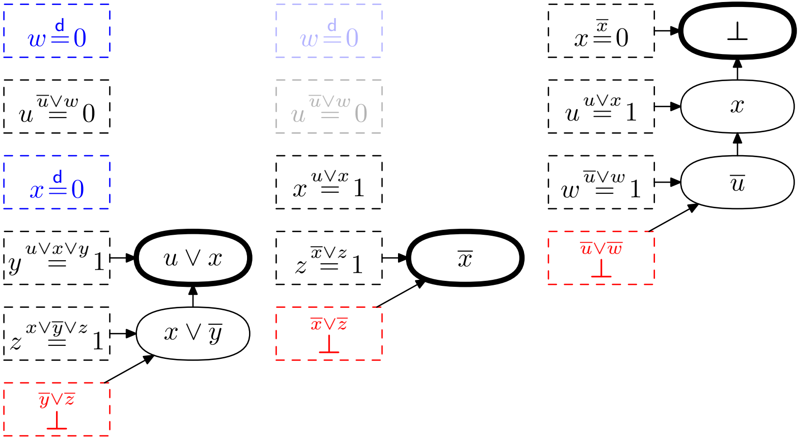
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #decisions so that last variable still flips

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Main Loop Pseudocode (High Level)

forever do
  if *current assignment falsifies clause* **then**
    apply learning scheme to derive new clause;
    if *learned clause empty* **then** output `UNSATISFIABLE` and exit;
    else
      add learned clause and backjump
    end
  else if *all variables assigned* **then** output `SATISFIABLE` and exit;
  else if *exists unit clause* C *propagating* x *to value* $b \in \{0, 1\}$ **then**
    add propagated assignment $x \stackrel{C}{=} b$
  else if *time to restart* **then**
    remove all variable assignments
  else
    if *time for clause database reduction* **then**
      erase (roughly) half of learned clauses in memory
    end
    use decision scheme to choose assignment $x \stackrel{\mathrm{d}}{=} b$;
  end
end

CDCL Main Loop Pseudocode (High Level)

forever do
  if *current assignment falsifies clause* **then**
    apply learning scheme to derive new clause;
    if *learned clause empty* **then** output `UNSATISFIABLE` and exit;
    else
      add learned clause and backjump
    end
  else if *all variables assigned* **then** output `SATISFIABLE` and exit;
  else if *exists unit clause C propagating x to value $b \in \{0, 1\}$* **then**
    add propagated assignment $x \stackrel{C}{=} b$
  else if *time to restart* **then**
    remove all variable assignments
  else
    if *time for clause database reduction* **then**
      erase (roughly) half of learned clauses in memory
    end
    use decision scheme to choose assignment $x \stackrel{\mathrm{d}}{=} b$;
  end
end

CDCL Analysis and the Resolution Proof System

How to analyse CDCL performance?

Many intricate, hard-to-understand heuristics

Best(?) rigorous method: Focus on underlying method of reasoning

CDCL Analysis and the Resolution Proof System

How to analyse CDCL performance?
Many intricate, hard-to-understand heuristics
Best(?) rigorous method: Focus on underlying method of reasoning

Resolution proof system

- Start with clauses of formula
- Derive new clauses by **resolution rule**

$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

- Done when contradiction $\perp$ in form of empty clause derived

CDCL Analysis and the Resolution Proof System

How to analyse CDCL performance?
Many intricate, hard-to-understand heuristics
Best(?) rigorous method: Focus on underlying method of reasoning

Resolution proof system

- Start with clauses of formula
- Derive new clauses by **resolution rule**

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

- Done when contradiction $\perp$ in form of empty clause derived

When run on unsatisfiable formula, CDCL generates resolution proof*
So lower bounds on proof size $\Rightarrow$ lower bounds on running time

CDCL Analysis and the Resolution Proof System

How to analyse CDCL performance?
Many intricate, hard-to-understand heuristics
Best(?) rigorous method: Focus on underlying method of reasoning

Resolution proof system

- Start with clauses of formula
- Derive new clauses by **resolution rule**

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

- Done when contradiction $\perp$ in form of empty clause derived

When run on unsatisfiable formula, CDCL generates resolution proof*
So lower bounds on proof size $\Rightarrow$ lower bounds on running time

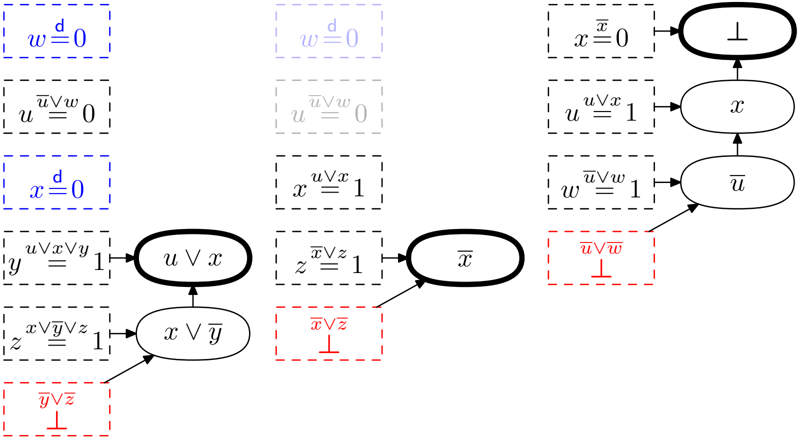
(*) Ignores preprocessing, but we don't have time to go into this

Resolution Proofs from CDCL Executions

Obtain resolution proof. . .

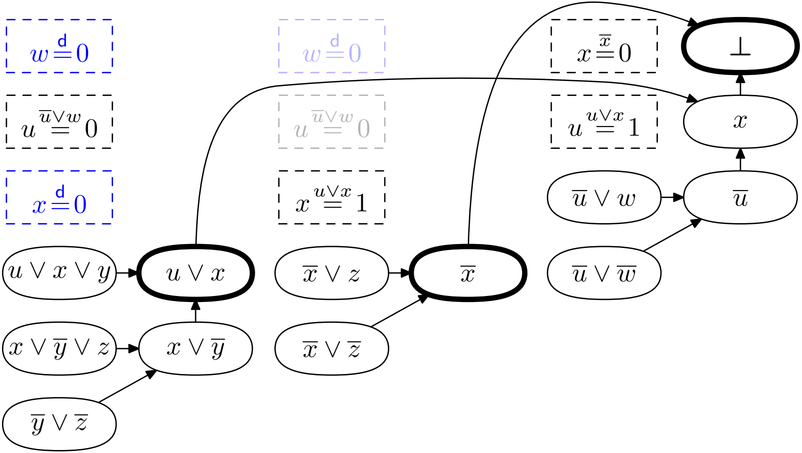
Resolution Proofs from CDCL Executions

Obtain resolution proof from our example CDCL execution. . .



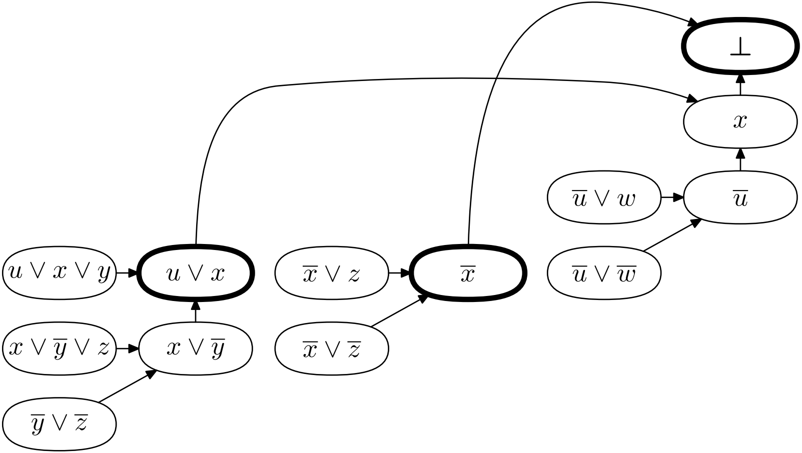
Resolution Proofs from CDCL Executions

Obtain resolution proof from our example CDCL execution by stringing together conflict analyses:



Resolution Proofs from CDCL Executions

Obtain resolution proof from our example CDCL execution by stringing together conflict analyses:



Current State of Affairs

- State-of-the-art CDCL solvers often perform amazingly well ("SAT is easy in practice")
- Very poor theoretical understanding:
  - Why do heuristics work?
  - Why are applied instances easy?
- Paradox: resolution quite weak proof system; many strong lower bounds for "obvious" formulas, e.g., [Hak85, Urq87, BW01, MN14]
- Explore stronger reasoning methods (potential exponential speed-up)
- In particular, pseudo-Boolean solving (a.k.a. 0-1 integer programming) corresponding to cutting planes proof system
- Importantly, extends to pseudo-Boolean optimization [we will return to this topic in Part III]

Approaches to Pseudo-Boolean Solving

Conversion to disjunctive clauses

- Lazy approach: learn clauses from PB constraints
  - SAT4J [LP10] (one of versions in library)

Approaches to Pseudo-Boolean Solving

Conversion to disjunctive clauses

- Lazy approach: learn clauses from PB constraints
  - Sat4j [LP10] (one of versions in library)
- Eager approach: re-encode to clauses and run CDCL
  - MiniSat+ [ES06]
  - Open-WBO [MML14]
  - NaPS [SN15]

Approaches to Pseudo-Boolean Solving

Conversion to disjunctive clauses

- Lazy approach: learn clauses from PB constraints
  - SAT4J [LP10] (one of versions in library)
- Eager approach: re-encode to clauses and run CDCL
  - MINISAT+ [ES06]
  - OPEN-WBO [MML14]
  - NAPS [SN15]

Native reasoning with pseudo-Boolean constraints

- PRS [DG02]
- GALENA [CK05]
- PUEBLO [SS06]
- SAT4J [LP10]
- ROUNDINGSAT [EN18]

Re-encoding to CNF

- CNF encoding can be exponentially larger than PB encoding
- Use extension variables for more compact encoding
- High-level idea: new variables = gates in circuit evaluating PB constraint
- Consider first two concrete examples for cardinality constraints

$$\sum_{i=1}^{n} x_i \bowtie k$$

(where $\bowtie \in \{\geq, \leq, =\}$)

Sequential Counter Encoding

$\sum_{i=1}^{n} x_i \bowtie k$ for $\bowtie \in \{\geq, \leq, =\}$

$s_{i,j} =$ "sum of i first variables $\geq j$" (from [Sin05] with slight twists)

Sequential Counter Encoding

$\sum_{i=1}^{n} x_i \bowtie k$ for $\bowtie \in \{\geq, \leq, =\}$

$s_{i,j}$ = "sum of i first variables $\geq j$" (from [Sin05] with slight twists)

Base case ($j > 1$):

$$\overline{x}_1 \vee s_{1,1}$$

$$\overline{s}_{1,j}$$

$$x_1 \vee \overline{s}_{1,1}$$

Inductive step ($i \geq 2$, $j \geq 1$):

$$\overline{x}_i \vee s_{i,1}$$

$$\overline{s}_{i-1,j} \vee s_{i,j}$$

$$\overline{x}_i \vee \overline{s}_{i-1,j} \vee s_{i,j+1}$$

$$x_i \vee s_{i-1,j+1} \vee \overline{s}_{i,j+1}$$

$$s_{i-1,j} \vee s_{i-1,j+1} \vee \overline{s}_{i,j+1}$$

Sequential Counter Encoding

$\sum_{i=1}^{n} x_i \bowtie k$ for $\bowtie \in \{\geq, \leq, =\}$

$s_{i,j}$ = "sum of i first variables $\geq j$" (from [Sin05] with slight twists)

Base case ($j > 1$):

$$\overline{x}_1 \lor s_{1,1}$$
$$\overline{s}_{1,j}$$
$$x_1 \lor \overline{s}_{1,1}$$

Inductive step ($i \geq 2$, $j \geq 1$):

$$\overline{x}_i \lor s_{i,1}$$
$$\overline{s}_{i-1,j} \lor s_{i,j}$$
$$\overline{x}_i \lor \overline{s}_{i-1,j} \lor s_{i,j+1}$$
$$x_i \lor s_{i-1,j+1} \lor \overline{s}_{i,j+1}$$
$$s_{i-1,j} \lor s_{i-1,j+1} \lor \overline{s}_{i,j+1}$$

To enforce cardinality constraint

- $\bowtie \doteq \geq$: Add unit clause $s_{n,k}$
- $\bowtie \doteq \leq$: Add unit clause $\overline{s}_{n,k+1}$
- $\bowtie \doteq =$: Add both unit clauses above

Totalizer Encoding

$\sum_{i=1}^{n} x_i \bowtie k$ for $\bowtie \in \{\geq, \leq, =\}$

Build binary tree: children have t bits a_i, b_i each; parent outputs $2t$ bits c_j
$c_j =$ "sum of input variables $\geq j$" [BB03]

Totalizer Encoding

$\sum_{i=1}^{n} x_i \bowtie k$ for $\bowtie \in \{\geq, \leq, =\}$

Build binary tree: children have t bits a_i, b_i each; parent outputs $2t$ bits c_j
$c_j =$ "sum of input variables $\geq j$" [BB03]

Base case (two bits x_1, x_2):

$$\overline{x}_i \vee c_1$$
$$\overline{x}_1 \vee \overline{x}_2 \vee c_2$$
$$x_1 \vee x_2 \vee \overline{c}_1$$
$$x_i \vee \overline{c}_2$$

Inductive step ($i + j \geq 1$):

$$\overline{a}_i \vee \overline{b}_j \vee c_{i+j}$$
$$a_{i+1} \vee b_{j+1} \vee \overline{c}_{i+j+1}$$
$$(a_0 = b_0 = 1)$$

Totalizer Encoding

$\sum_{i=1}^{n} x_i \bowtie k$ for $\bowtie \in \{\geq, \leq, =\}$

Build binary tree: children have t bits a_i, b_i each; parent outputs $2t$ bits c_j
$c_j =$ "sum of input variables $\geq j$" [BB03]

Base case (two bits x_1, x_2):

$$\overline{x}_i \vee c_1$$
$$\overline{x}_1 \vee \overline{x}_2 \vee c_2$$
$$x_1 \vee x_2 \vee \overline{c}_1$$
$$x_i \vee \overline{c}_2$$

Inductive step ($i + j \geq 1$):

$$\overline{a}_i \vee \overline{b}_j \vee c_{i+j}$$
$$a_{i+1} \vee b_{j+1} \vee \overline{c}_{i+j+1}$$
$$(a_0 = b_0 = 1)$$

To enforce cardinality constraint, add for root node

- $\bowtie \doteq \geq$: unit clause c_k
- $\bowtie \doteq \leq$: unit clause $\overline{c}_{k+1}$
- $\bowtie \doteq =$: both unit clauses above

Can be extended to arbitrary PB constraints [JMM15]; blow-up can be bad

Adder Network Encoding (Sketch)

- For general pseudo-Boolean constraints $\sum_{i=1}^{n} a_i \ell_i \geq A$, write coefficients a_i in binary $\langle a_{i,B}\, a_{i,B-1} \cdots a_{i,1}\, a_{i,0} \rangle$

Adder Network Encoding (Sketch)

- For general pseudo-Boolean constraints $\sum_{i=1}^{n} a_i\ell_i \geq A$, write coefficients a_i in binary $\langle a_{i,B}\, a_{i,B-1} \cdots a_{i,1}\, a_{i,0}\rangle$
- Assuming B large enough for rest of this slide, it clearly holds that

$$\sum_{i=1}^{n} a_i\ell_i = \sum_{i=1}^{n}\sum_{j=0}^{B} 2^j \cdot a_{i,j} \cdot \ell_i$$

Adder Network Encoding (Sketch)

- For general pseudo-Boolean constraints $\sum_{i=1}^{n} a_i \ell_i \geq A$, write coefficients a_i in binary $\langle a_{i,B}\, a_{i,B-1} \cdots a_{i,1}\, a_{i,0} \rangle$
- Assuming B large enough for rest of this slide, it clearly holds that

$$\sum_{i=1}^{n} a_i \ell_i = \sum_{i=1}^{n} \sum_{j=0}^{B} 2^j \cdot a_{i,j} \cdot \ell_i$$

- Introduce new variables c_{out}, s_{out} and use encodings of full adders

$$2 \cdot c_{\mathrm{out}} + s_{\mathrm{out}} = x + y + z$$

in CNF to enforce

$$\sum_{i=1}^{n} \sum_{j=0}^{B} 2^j \cdot a_{i,j} \cdot \ell_i = \sum_{j=0}^{B} 2^j \cdot s_j \cdot \ell_i \qquad \text{and} \qquad \sum_{j=0}^{B} 2^j \cdot s_j \cdot \ell_i \geq A$$

Adder Network Encoding (Sketch)

- For general pseudo-Boolean constraints $\sum_{i=1}^{n} a_i \ell_i \geq A$, write coefficients a_i in binary $\langle a_{i,B}\, a_{i,B-1} \cdots a_{i,1}\, a_{i,0} \rangle$
- Assuming B large enough for rest of this slide, it clearly holds that

$$\sum_{i=1}^{n} a_i \ell_i = \sum_{i=1}^{n} \sum_{j=0}^{B} 2^j \cdot a_{i,j} \cdot \ell_i$$

- Introduce new variables c_{out}, s_{out} and use encodings of full adders

$$2 \cdot c_{\mathrm{out}} + s_{\mathrm{out}} = x + y + z$$

  in CNF to enforce

$$\sum_{i=1}^{n} \sum_{j=0}^{B} 2^j \cdot a_{i,j} \cdot \ell_i = \sum_{j=0}^{B} 2^j \cdot s_j \cdot \ell_i \qquad \text{and} \qquad \sum_{j=0}^{B} 2^j \cdot s_j \cdot \ell_i \geq A$$

- See [ES06] for all the missing details...

CNF Encoding Desiderata

Generalized arc consistency (GAC)

For F_C encoding PB constraint C and ρ partial assignment, want:

- If C propagates under ρ, then F_C should yield same propagations
- If ρ falsifies C, then F_C should unit propagate to contradiction

True for sequential counter and totalizer; false for adder network

CNF Encoding Desiderata

Generalized arc consistency (GAC)

For F_C encoding PB constraint C and ρ partial assignment, want:

- If C propagates under ρ, then F_C should yield same propagations
- If ρ falsifies C, then F_C should unit propagate to contradiction

True for sequential counter and totalizer; false for adder network

Encoding size

Want as few variables and clauses as possible

Adder network very compact
Totalizer has fewer variables than sequential counter
But generalized totalizer encoding can get exponentially large

CNF Encoding Desiderata

Generalized arc consistency (GAC)

For F_C encoding PB constraint C and ρ partial assignment, want:

- If C propagates under ρ, then F_C should yield same propagations
- If ρ falsifies C, then F_C should unit propagate to contradiction

True for sequential counter and totalizer; false for adder network

Encoding size

Want as few variables and clauses as possible

Adder network very compact
Totalizer has fewer variables than sequential counter
But generalized totalizer encoding can get exponentially large

Possible to achieve both GAC and polynomial-size encoding [BBR09]
But complicated; and in practice not better than totalizer [JMM15]?
Rich literature on encodings — see SAT handbook for more references

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]

Question About Forward vs. Backward Propagation

- Forward propagation: If $\sum_{i=1}^{n} x_i \geq k$ true, then $s_{n,k}$ / c_k propagates to true
- Backward propagation: If $\sum_{i=1}^{n} x_i \geq k$ false, then $s_{n,k}$ / c_k propagates to false

Sequential counter

$$\overline{x}_i \vee s_{i,1}$$
$$\overline{s}_{i-1,j} \vee s_{i,j}$$
$$\overline{x}_i \vee \overline{s}_{i-1,j} \vee s_{i,j+1}$$
$$x_i \vee s_{i-1,j+1} \vee \overline{s}_{i,j+1}$$
$$s_{i-1,j} \vee s_{i-1,j+1} \vee \overline{s}_{i,j+1}$$

Totalizer

$$\overline{a}_i \vee \overline{b}_j \vee c_{i+j}$$
$$a_{i+1} \vee b_{j+1} \vee \overline{c}_{i+j+1}$$

Question About Forward vs. Backward Propagation

- Forward propagation: If $\sum_{i=1}^{n} x_i \geq k$ true, then $s_{n,k}$ / c_k propagates to true
- Backward propagation: If $\sum_{i=1}^{n} x_i \geq k$ false, then $s_{n,k}$ / c_k propagates to false

Sequential counter

$$\overline{x}_i \lor s_{i,1}$$

$$\overline{s}_{i-1,j} \lor s_{i,j}$$

$$\overline{x}_i \lor \overline{s}_{i-1,j} \lor s_{i,j+1}$$

$$x_i \lor s_{i-1,j+1} \lor \overline{s}_{i,j+1}$$

$$s_{i-1,j} \lor s_{i-1,j+1} \lor \overline{s}_{i,j+1}$$

Totalizer

$$\overline{a}_i \lor \overline{b}_j \lor c_{i+j}$$

$$a_{i+1} \lor b_{j+1} \lor \overline{c}_{i+j+1}$$

Question About Forward vs. Backward Propagation

- Forward propagation: If $\sum_{i=1}^{n} x_i \geq k$ true, then $s_{n,k}$ / c_k propagates to true
- Backward propagation: If $\sum_{i=1}^{n} x_i \geq k$ false, then $s_{n,k}$ / c_k propagates to false

Sequential counter

$$\overline{x}_i \lor s_{i,1}$$
$$\overline{s}_{i-1,j} \lor s_{i,j}$$
$$\overline{x}_i \lor \overline{s}_{i-1,j} \lor s_{i,j+1}$$
$$x_i \lor s_{i-1,j+1} \lor \overline{s}_{i,j+1}$$
$$s_{i-1,j} \lor s_{i-1,j+1} \lor \overline{s}_{i,j+1}$$

Totalizer

$$\overline{a}_i \lor \overline{b}_j \lor c_{i+j}$$
$$a_{i+1} \lor b_{j+1} \lor \overline{c}_{i+j+1}$$

Question About Forward vs. Backward Propagation

- Forward propagation: If $\sum_{i=1}^{n} x_i \geq k$ true, then $s_{n,k}$ / c_k propagates to true
- Backward propagation: If $\sum_{i=1}^{n} x_i \geq k$ false, then $s_{n,k}$ / c_k propagates to false

Sequential counter

$$\overline{x}_i \lor s_{i,1}$$
$$\overline{s}_{i-1,j} \lor s_{i,j}$$
$$\overline{x}_i \lor \overline{s}_{i-1,j} \lor s_{i,j+1}$$
$$x_i \lor s_{i-1,j+1} \lor \overline{s}_{i,j+1}$$
$$s_{i-1,j} \lor s_{i-1,j+1} \lor \overline{s}_{i,j+1}$$

Totalizer

$$\overline{a}_i \lor \overline{b}_j \lor c_{i+j}$$
$$a_{i+1} \lor b_{j+1} \lor \overline{c}_{i+j+1}$$

Solvers like Open-WBO [MML14] only encode forward propagation

- Can having propagation in both directions help?
- Or does it on the contrary hurt? Why?

More Questions

1. How to find best possible CNF encodings of PB constraints for given problem?
   - Trade-offs between propagation strength and encoding size?
   - Rigorous mathematical insights?
2. Understand complementary strengths of CDCL-based and "native" cutting-planes-based PB solving?
   - Theoretical results on computational complexity?
   - Harness complementary strengths in applied solvers?
3. How to make sure re-encoding into CNF is guaranteed to be correct?

"Native" Pseudo-Boolean Conflict-Driven Search

Want to do "same thing" as CDCL but with pseudo-Boolean 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 instance
  3. Backjump by rolling back decisions so that asserting literal flips

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{\text{(ordered) set of literals assigned true}\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

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

ρ	$slack(C;\rho)$	comment

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

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

ρ	$slack(C;\rho)$	comment
$\{\}$	8	

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

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

ρ	$slack(C;\rho)$	comment
$\{\}$	8	
$\{\overline{x}_5\}$	3	propagates $\overline{x}_4$ (coefficient $>$ slack)

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i\ell_i \geq A$

$$slack(\textstyle\sum_i a_i\ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

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

ρ	$slack(C; \rho)$	comment
$\{\}$	8	
$\{\overline{x}_5\}$	3	propagates $\overline{x}_4$ (coefficient $>$ slack)
$\{\overline{x}_5, \overline{x}_4\}$	3	propagation doesn't change slack

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

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

ρ	$slack(C;\rho)$	comment
$\{\}$	8	
$\{\overline{x}_5\}$	3	propagates $\overline{x}_4$ (coefficient $>$ slack)
$\{\overline{x}_5, \overline{x}_4\}$	3	propagation doesn't change slack
$\{\overline{x}_5, \overline{x}_4, \overline{x}_3, x_2\}$	-2	conflict (slack < 0)

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

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

ρ	$slack(C;\rho)$	comment
$\{\}$	8	
$\{\overline{x}_5\}$	3	propagates $\overline{x}_4$ (coefficient $>$ slack)
$\{\overline{x}_5, \overline{x}_4\}$	3	propagation doesn't change slack
$\{\overline{x}_5, \overline{x}_4, \overline{x}_3, x_2\}$	-2	conflict (slack < 0)

Note that constraint can be conflicting though not all variables assigned

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

$\boxed{z \stackrel{x \lor \overline{y} \lor z}{=} 1}$

$\boxed{\stackrel{\overline{y} \lor \overline{z}}{\bot}}$

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$w \overset{\mathsf{d}}{=} 0$

$u \overset{\overline{u} \lor w}{=} 0$

$x \overset{\mathsf{d}}{=} 0$

$y \overset{u \lor x \lor y}{=} 1$

$z \overset{x \lor \overline{y} \lor z}{=} 1$

$\overset{\overline{y} \lor \overline{z}}{\perp}$

Assignment "left on trail" always falsifies derived clause

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \lor \overline{w}}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

$y \stackrel{u \lor x \lor y}{=} 1$

$z \stackrel{x \lor \overline{y} \lor z}{=} 1$

$\stackrel{\overline{y} \lor \overline{z}}{\bot}$

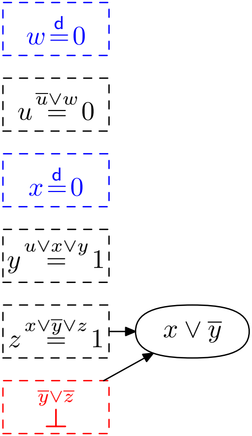
Assignment "left on trail" always falsifies derived clause

$\overline{y} \lor \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Assignment "left on trail" always falsifies derived clause

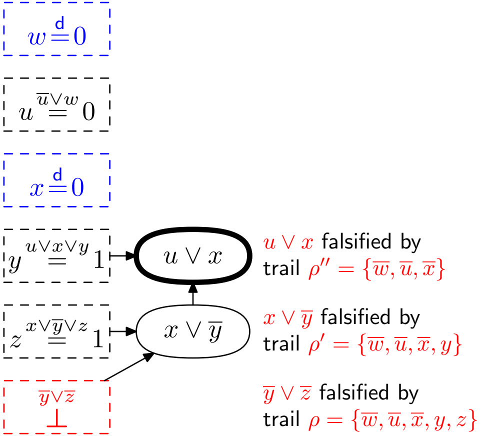
$x \lor \overline{y}$ falsified by trail $\rho' = \{\overline{w}, \overline{u}, \overline{x}, y\}$

$\overline{y} \lor \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

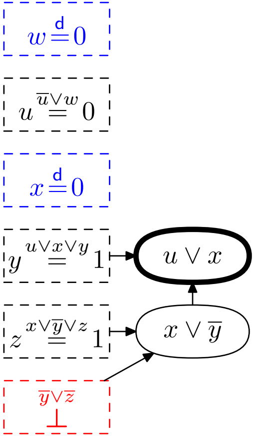


Assignment "left on trail" always falsifies derived clause

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



$u \lor x$ falsified by trail $\rho'' = \{\overline{w}, \overline{u}, \overline{x}\}$

$x \lor \overline{y}$ falsified by trail $\rho' = \{\overline{w}, \overline{u}, \overline{x}, y\}$

$\overline{y} \lor \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

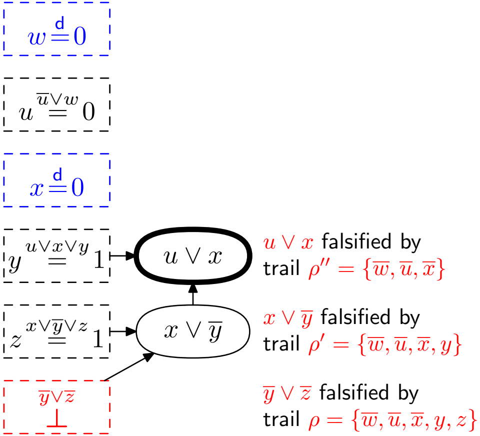
Assignment "left on trail" always falsifies derived clause

$\Rightarrow$ every derived constraint "explains" conflict

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Assignment "left on trail" always falsifies derived clause

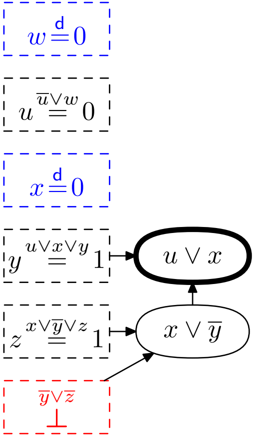
$\Rightarrow$ every derived constraint "explains" conflict

Terminate conflict analysis when explanation looks nice

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



$u \lor x$ falsified by trail $\rho'' = \{\overline{w}, \overline{u}, \overline{x}\}$

$x \lor \overline{y}$ falsified by trail $\rho' = \{\overline{w}, \overline{u}, \overline{x}, y\}$

$\overline{y} \lor \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

Assignment "left on trail" always falsifies derived clause

$\Rightarrow$ every derived constraint "explains" conflict

Terminate conflict analysis when explanation looks nice

Learn asserting constraint: after backjump, some variable guaranteed to flip

Generalized Resolution

Can mimic resolution step

$$\frac{x \lor \overline{y} \lor z \qquad \overline{y} \lor \overline{z}}{x \lor \overline{y}}$$

Generalized Resolution

Can mimic resolution step

$$\frac{x \lor \overline{y} \lor z \qquad \overline{y} \lor \overline{z}}{x \lor \overline{y}}$$

by adding clauses as pseudo-Boolean constraints

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

(Recall $z + \overline{z} = 1$)

Generalized Resolution

Can mimic resolution step

$$\frac{x \vee \overline{y} \vee z \qquad \overline{y} \vee \overline{z}}{x \vee \overline{y}}$$

by adding clauses as pseudo-Boolean constraints

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

(Recall $z + \overline{z} = 1$)

Generalized resolution rule (from [Hoo88, Hoo92])
Positive linear combination so that some variable cancels

$$\frac{a_1 x_1 + \sum_{i \geq 2} a_i \ell_i \geq A \qquad b_1 \overline{x}_1 + \sum_{i \geq 2} b_i \ell_i \geq B}{\sum_{i \geq 2} \left(\frac{c}{a_1} a_i + \frac{c}{b_1} b_i\right) \ell_i \geq \frac{c}{a_1} A + \frac{c}{b_1} B - c} \; [c = \mathrm{lcm}(a_1, b_1)]$$

Saturation

Actually, don't get quite the right constraint in mimicking of resolution

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

Saturation

Actually, don't get quite the right constraint in mimicking of resolution

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

But clearly valid to conclude

$$\frac{x + 2\overline{y} \geq 1}{x + \overline{y} \geq 1}$$

Saturation

Actually, don't get quite the right constraint in mimicking of resolution

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

But clearly valid to conclude

$$\frac{x + 2\overline{y} \geq 1}{x + \overline{y} \geq 1}$$

Saturation rule

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

Sound over integers, not over rationals (need such rules for SAT solving)

Saturation

Actually, don't get quite the right constraint in mimicking of resolution

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

But clearly valid to conclude

$$\frac{x + 2\overline{y} \geq 1}{x + \overline{y} \geq 1}$$

Saturation rule

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

Sound over integers, not over rationals (need such rules for SAT solving)

[Generalized resolution as defined in [Hoo88, Hoo92] includes fix above, but convenient here to make the two separate steps explicit]

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2
(Note: same constraint can propagate several times!)

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2
(Note: same constraint can propagate several times!)

- Resolve reason$(x_3, \rho) \doteq C_1$ with C_2 over x_3 to get resolve(C_1, C_2, x_3)

$$\frac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{x_4 \geq 1}$$

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2
(Note: same constraint can propagate several times!)

- Resolve reason$(x_3, \rho) \doteq C_1$ with C_2 over x_3 to get resolve(C_1, C_2, x_3)

$$\frac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{x_4 \geq 1}$$

- Applying saturate$(x_4 \geq 1)$ does nothing

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2
(Note: same constraint can propagate several times!)

- Resolve reason$(x_3, \rho) \doteq C_1$ with C_2 over x_3 to get resolve(C_1, C_2, x_3)

$$\frac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{x_4 \geq 1}$$

- Applying saturate$(x_4 \geq 1)$ does nothing
- Non-negative slack w.r.t. $\rho' = \{x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1\}$ — not conflicting!

What Went Wrong? And What to Do About It?

Accident report

- Generalized resolution sound over the reals
- Given $\rho' = \{x_1 = 0, x_2 = 1\}$, over the reals have
  - $C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$ propagates $x_3 \geq \frac{1}{2}$
  - $C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$ satisfied by $x_3 \leq \frac{1}{2}$
- So after resolving away x_3, "can't see any conflict"

What Went Wrong? And What to Do About It?

Accident report

- Generalized resolution sound over the reals
- Given $\rho' = \{x_1 = 0, x_2 = 1\}$, over the reals have
  - $C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$ propagates $x_3 \geq \frac{1}{2}$
  - $C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$ satisfied by $x_3 \leq \frac{1}{2}$
- So after resolving away x_3, "can't see any conflict"

Remedial action

- Strengthen propagation to $x_3 \geq 1$ also over the reals
- I.e., want reason C with $slack(C; \rho') = 0$
- **Fix** (non-obvious): Apply weakening

$$\text{weaken}(\textstyle\sum_i a_i\ell_i \geq A, \ell_j) = \textstyle\sum_{i \neq j} a_i\ell_i \geq A - a_j$$

  to reason constraint and then saturate
- Approach in [CK05] (seems to go back to observations in [Wil76])

Try to Reduce the Reason Constraint

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

Let's try to

1. Weaken reason on non-falsified literal (but not last propagated)
2. Saturate weakened constraint
3. Resolve with conflicting constraint over propagated literal

Try to Reduce the Reason Constraint

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

Let's try to

1. Weaken reason on non-falsified literal (but not last propagated)
2. Saturate weakened constraint
3. Resolve with conflicting constraint over propagated literal

$$\text{resolve } x_3 \dfrac{\text{saturate} \dfrac{\text{weaken } x_2 \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 + x_4 \geq 2}}{2x_1 + 2x_3 + x_4 \geq 2} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 + x_4 \geq 1}$$

Try to Reduce the Reason Constraint

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

Let's try to

1. Weaken reason on non-falsified literal (but not last propagated)
2. Saturate weakened constraint
3. Resolve with conflicting constraint over propagated literal

$$\dfrac{\dfrac{\dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 + x_4 \geq 2}\,\text{weaken } x_2}{2x_1 + 2x_3 + x_4 \geq 2}\,\text{saturate} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 + x_4 \geq 1}\,\text{resolve } x_3$$

Bummer! Still non-negative slack — not conflicting

Try Again to Reduce the Reason Constraint. . .

$$
\begin{aligned}
C_1 &\doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \\
C_2 &\doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3
\end{aligned}
$$

Trail $\rho = \{x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

Try Again to Reduce the Reason Constraint...

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

$$\dfrac{\text{resolve } x_3\;\dfrac{\text{saturate}\;\dfrac{\text{weaken } \{x_2, x_4\}\;\dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 \geq 1}}{x_1 + x_3 \geq 1} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 \geq 1}$$

Try Again to Reduce the Reason Constraint...

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

$$\dfrac{\text{resolve } x_3\ \dfrac{\text{saturate}\ \dfrac{\text{weaken } \{x_2, x_4\}\ \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 \geq 1}}{x_1 + x_3 \geq 1} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 \geq 1}$$

Negative slack — conflicting!

Try Again to Reduce the Reason Constraint...

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

$$\dfrac{\text{resolve } x_3\ \dfrac{\text{saturate}\ \dfrac{\text{weaken } \{x_2, x_4\}\ \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 \geq 1}}{x_1 + x_3 \geq 1} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 \geq 1}$$

Negative slack — conflicting!

Backjump propagates to conflict without decisions

Done! Next conflict analysis will derive contradiction

(Or, in practice, terminate immediately when conflict without decisions)

Reason Reduction Using Saturation [CK05]

$\text{reduceSat}(C_{\text{confl}}, C_{\text{reason}}, \ell, \rho)$

while $\mathit{slack}(\mathsf{resolve}(C_{\text{confl}}, C_{\text{reason}}, \ell); \rho) \geq 0$ **do**
  $\ell' \leftarrow$ literal in $C_{\text{reason}} \setminus \{\ell\}$ not falsified by ρ;
  $C_{\text{reason}} \leftarrow \mathsf{saturate}(\mathsf{weaken}(C_{\text{reason}}, \ell'))$;
end
return C_{reason};

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while $slack(\mathsf{resolve}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
  $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
  $C_{\mathrm{reason}} \leftarrow \mathsf{saturate}(\mathsf{weaken}(C_{\mathrm{reason}}, \ell'))$;
end
return C_{reason};

Why does this work?

- Slack is subadditive

$$slack(c \cdot C + d \cdot D; \rho) \leq c \cdot slack(C; \rho) + d \cdot slack(D; \rho)$$

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
    $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
    $C_{\mathrm{reason}} \leftarrow \mathsf{saturate}(\mathsf{weaken}(C_{\mathrm{reason}}, \ell'))$;
end
return C_{reason};

Why does this work?

- Slack is subadditive

$$\mathit{slack}(c \cdot C + d \cdot D; \rho) \leq c \cdot \mathit{slack}(C; \rho) + d \cdot \mathit{slack}(D; \rho)$$

- By invariant have $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
  $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
  $C_{\mathrm{reason}} \leftarrow \mathsf{saturate}(\mathsf{weaken}(C_{\mathrm{reason}}, \ell'))$;
end
return C_{reason};

Why does this work?

- Slack is subadditive

$$\mathit{slack}(c \cdot C + d \cdot D; \rho) \leq c \cdot \mathit{slack}(C; \rho) + d \cdot \mathit{slack}(D; \rho)$$

- By invariant have $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
- Weakening leaves $\mathit{slack}(C_{\mathrm{reason}}; \rho)$ unchanged

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while $\mathit{slack}(\text{resolve}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
    $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
    $C_{\mathrm{reason}} \leftarrow$ saturate(weaken($C_{\mathrm{reason}}, \ell'$));
end
return C_{reason};

Why does this work?

- Slack is subadditive

$$\mathit{slack}(c \cdot C + d \cdot D; \rho) \leq c \cdot \mathit{slack}(C; \rho) + d \cdot \mathit{slack}(D; \rho)$$

- By invariant have $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
- Weakening leaves $\mathit{slack}(C_{\mathrm{reason}}; \rho)$ unchanged
- Saturation decreases slack — reach 0 when max #literals weakened

Pseudo-Boolean Conflict Analys

analyzePBconflict(C_{confl}, ρ)

while C_{confl} *not asserting* **do**
  $\ell \leftarrow$ literal assigned last on trail ρ;
  if $\bar{\ell}$ *occurs in* C_{confl} **then**
    $C_{\mathrm{reason}} \leftarrow$ reason(ℓ, ρ);
    $\boldsymbol{C_{\mathrm{reason}} \leftarrow \mathbf{reduceSat}(C_{\mathrm{reason}}, C_{\mathrm{confl}}, \ell, \rho)}$;
    $C_{\mathrm{confl}} \leftarrow$ resolve$(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell)$;
    $C_{\mathrm{confl}} \leftarrow$ saturate(C_{confl});
  end
  $\rho \leftarrow \mathit{removeLast}(\rho)$;
end
return C_{confl};

Reduction of reason new compared to CDCL — everything else the same
Essentially conflict analysis used in SAT4J [LP10]

Some Problems Compared to CDCL

- Compared to clauses harder to detect propagation for constraints like

$$\sum_{i=1}^{n} x_i \geq n - 1$$

Some Problems Compared to CDCL

- Compared to clauses harder to detect propagation for constraints like

$$\sum_{i=1}^{n} x_i \geq n-1$$

- Generalized resolution for general pseudo-Boolean constraints
  $\Rightarrow$ lots of lcm computations
  $\Rightarrow$ coefficient sizes can explode (expensive arithmetic)

Some Problems Compared to CDCL

- Compared to clauses harder to detect propagation for constraints like

$$\sum_{i=1}^{n} x_i \geq n - 1$$

- Generalized resolution for general pseudo-Boolean constraints
  $\Rightarrow$ lots of lcm computations
  $\Rightarrow$ coefficient sizes can explode (expensive arithmetic)

- For CNF inputs, degenerates to resolution!
  $\Rightarrow$ CDCL but with super-expensive data structures

The Cutting Planes Proof System

Cutting planes as defined in theory literature [CCT87] doesn't use saturation but instead division (a.k.a. Chvátal-Gomory cut)

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

The Cutting Planes Proof System

Cutting planes as defined in theory literature [CCT87] doesn't use saturation but instead division (a.k.a. Chvátal-Gomory cut)

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

- Cutting planes with division implicationally complete
- Cutting planes with saturation is **not** [VEG+18]
- Can division yield stronger conflict analysis?

The Cutting Planes Proof System

Cutting planes as defined in theory literature [CCT87] doesn't use saturation but instead division (a.k.a. Chvátal-Gomory cut)

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

- Cutting planes with division implicationally complete
- Cutting planes with saturation is **not** [VEG+18]
- Can division yield stronger conflict analysis?
  (Used for general integer linear programming in CutSat [JdM13])

Using Division to Reduce the Reason

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

Using Division to Reduce the Reason

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

1. Weaken reason on non-falsified literal(s) with coefficient not divisible by propagating literal coefficient
2. Divide weakened constraint by propagating literal coefficient
3. Resolve with conflicting constraint over propagated literal

Using Division to Reduce the Reason

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

1. Weaken reason on non-falsified literal(s) with coefficient not divisible by propagating literal coefficient
2. Divide weakened constraint by propagating literal coefficient
3. Resolve with conflicting constraint over propagated literal

$$\dfrac{\text{divide by } 2\ \dfrac{\text{weaken } x_4\ \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_2 + 2x_3 \geq 3}}{x_1 + x_2 + x_3 \geq 2} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{0 \geq 1}\ \text{resolve } x_3$$

Using Division to Reduce the Reason

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \{x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\}$ $\Rightarrow$ Conflict with C_2

1. Weaken reason on non-falsified literal(s) with coefficient not divisible by propagating literal coefficient
2. Divide weakened constraint by propagating literal coefficient
3. Resolve with conflicting constraint over propagated literal

$$\dfrac{\text{resolve } x_3 \quad \dfrac{\text{divide by } 2 \quad \dfrac{\text{weaken } x_4 \quad 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_2 + 2x_3 \geq 3}}{x_1 + x_2 + x_3 \geq 2} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{0 \geq 1}$$

Terminate immediately!

Reason Reduction Using Division [EN18]

$\mathrm{reduceDiv}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

$c \leftarrow \mathit{coeff}(C_{\mathrm{reason}}, \ell)$;
while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, \mathsf{divide}(C_{\mathrm{reason}}, c), \ell); \rho) \geq 0$ **do**
  $\ell_j \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ such that $\overline{\ell}_j \notin \rho$ and $c \nmid \mathit{coeff}(C, \ell_j)$;
  $C_{\mathrm{reason}} \leftarrow \mathsf{weaken}(C_{\mathrm{reason}}, \ell_j)$;
end
return $\mathsf{divide}(C_{\mathrm{reason}}, c)$;

Reason Reduction Using Division [EN18]

$\mathrm{reduceDiv}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

$c \leftarrow \mathit{coeff}(C_{\mathrm{reason}}, \ell)$;
while slack(resolve(C_{confl}, divide(C_{reason}, c), ℓ); ρ) ≥ 0 **do**
  $\ell_j \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ such that $\overline{\ell}_j \notin \rho$ and $c \nmid \mathit{coeff}(C, \ell_j)$;
  $C_{\mathrm{reason}} \leftarrow$ weaken($C_{\mathrm{reason}}, \ell_j$);
end
return divide(C_{reason}, c);

So now why does **this** work?

- Sufficient to get reason with slack 0 since
  1. $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
  2. slack is subadditive

Reason Reduction Using Division [EN18]

$\mathrm{reduceDiv}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

$c \leftarrow \mathit{coeff}(C_{\mathrm{reason}}, \ell)$;
while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, \mathsf{divide}(C_{\mathrm{reason}}, c), \ell); \rho) \geq 0$ **do**
  $\ell_j \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ such that $\overline{\ell}_j \notin \rho$ and $c \nmid \mathit{coeff}(C, \ell_j)$;
  $C_{\mathrm{reason}} \leftarrow \mathsf{weaken}(C_{\mathrm{reason}}, \ell_j)$;
end
return $\mathsf{divide}(C_{\mathrm{reason}}, c)$;

So now why does **this** work?

- Sufficient to get reason with slack 0 since
  1. $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
  2. slack is subadditive
- Weakening doesn't change slack $\Rightarrow$ always $0 \leq \mathit{slack}(C_{\mathrm{reason}}; \rho) < c$

Reason Reduction Using Division [EN18]

$\mathrm{reduceDiv}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

$c \leftarrow \mathit{coeff}(C_{\mathrm{reason}}, \ell)$;
while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, \mathsf{divide}(C_{\mathrm{reason}}, c), \ell); \rho) \geq 0$ **do**
  $\ell_j \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ such that $\overline{\ell}_j \notin \rho$ and $c \nmid \mathit{coeff}(C, \ell_j)$;
  $C_{\mathrm{reason}} \leftarrow \mathsf{weaken}(C_{\mathrm{reason}}, \ell_j)$;
end
return $\mathsf{divide}(C_{\mathrm{reason}}, c)$;

So now why does **this** work?

- Sufficient to get reason with slack 0 since
  1. $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
  2. slack is subadditive
- Weakening doesn't change slack $\Rightarrow$ always $0 \leq \mathit{slack}(C_{\mathrm{reason}}; \rho) < c$
- After max #weakenings have $0 \leq \mathit{slack}(\mathsf{divide}(C_{\mathrm{reason}}, c); \rho) < 1$

Round-to-1 Reduction used in RoundingSat

Reduction method used in RoundingSat does max weakening right away

roundToOne(C, ℓ, ρ)

$c \leftarrow \mathit{coeff}(C, \ell)$;
foreach *literal* ℓ_j *in* C **do**
    if $\overline{\ell}_j \notin \rho$ *and* $c \nmid \mathit{coeff}(C, \ell_j)$ **then**
        $C \leftarrow$ weaken(C, ℓ_j);
    end
end
return divide(C, c);

And roundToOne used more aggressively in conflict analysis in [EN18] (though now we are dialling back on this. . .)

RoundingSat Conflict Analysis

$\text{analyzePBconflict}(C_{\text{confl}}, \rho)$

while C_{confl} *contains no or multiple falsified literals on last level* **do**
  if *no current solver decisions* **then**
    output `UNSATISFIABLE` and terminate
  end
  $\ell \leftarrow$ literal assigned last on trail ρ;
  if $\overline{\ell}$ *occurs in* C_{confl} **then**
    $C_{\text{confl}} \leftarrow \text{roundToOne}(C_{\text{confl}}, \overline{\ell}, \rho)$;
    $C_{\text{reason}} \leftarrow \text{roundToOne}(\text{reason}(\ell, \rho), \ell, \rho)$;
    $C_{\text{confl}} \leftarrow \text{resolve}(C_{\text{confl}}, C_{\text{reason}}, \ell)$;
  end
  $\rho \leftarrow \mathit{removeLast}(\rho)$;
end
$\ell \leftarrow$ literal in C_{confl} last falsified by ρ;
return $\text{roundToOne}(C_{\text{confl}}, \ell, \rho)$;

Division vs. Saturation

- Higher conflict speed when PB reasoning doesn't help [EN18]
- Seems to perform better when PB reasoning crucial [EGNV18]
- Keeps coefficients small — can (often) do fixed-precision arithmetic
- But SAT4J still better for some circuit verification problems [LBD+20]
- And still equally hard to detect propagation
- Also, still degenerates to resolution for CNF inputs
- Sometimes very poor performance even on infeasible 0-1 LPs!

Other PB Rules I: Cardinality Constraint Reduction

Given PB constraint

$$3x_1 + 2x_2 + x_3 + x_4 \geq 4$$

can compute least #literals that have to be true

Other PB Rules I: Cardinality Constraint Reduction

Given PB constraint

$$3x_1 + 2x_2 + x_3 + x_4 \geq 4$$

can compute least #literals that have to be true

$$x_1 + x_2 + x_3 + x_4 \geq 2$$

Other PB Rules I: Cardinality Constraint Reduction

Given PB constraint

$$3x_1 + 2x_2 + x_3 + x_4 \geq 4$$

can compute least #literals that have to be true

$$x_1 + x_2 + x_3 + x_4 \geq 2$$

GALENA [CK05] only learns cardinality constraints — easier to deal with

Other PB Rules I: Cardinality Constraint Reduction

Given PB constraint

$$3x_1 + 2x_2 + x_3 + x_4 \geq 4$$

can compute least #literals that have to be true

$$x_1 + x_2 + x_3 + x_4 \geq 2$$

GALENA [CK05] only learns cardinality constraints — easier to deal with

---

Cardinality constraint reduction rule

$$\frac{\sum_i a_i \ell_i \geq A}{\sum_{i\,:\,a_i>0} \ell_i \geq T} \quad T = \min\{|I| \,:\, I \subseteq [n], \sum_{i \in I} a_i \geq A\}$$

Can be simulated with weakening + division

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

- $y \overset{x+y\geq 1}{=} 1$ and $z \overset{x+z\geq 1}{=} 1 \Rightarrow y + z \geq 1$ oversatisfied by margin 1

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

- $y \overset{x+y\geq 1}{=} 1$ and $z \overset{x+z\geq 1}{=} 1 \Rightarrow y + z \geq 1$ oversatisfied by margin 1
- Hence, can deduce constraint $x + y + z \geq 2$

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

- $y \overset{x+y\geq 1}{=} 1$ and $z \overset{x+z\geq 1}{=} 1 \Rightarrow y + z \geq 1$ oversatisfied by margin 1
- Hence, can deduce constraint $x + y + z \geq 2$

Strengthening rule (imported by [DG02] from operations research)

- Suppose $\ell = 0 \Rightarrow \sum_i a_i \ell_i \geq A$ oversatisfied by amount K
- Then can deduce $K\ell + \sum_i a_i \ell_i \geq A + K$

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

- $y \stackrel{x+y\geq 1}{=} 1$ and $z \stackrel{x+z\geq 1}{=} 1 \Rightarrow y + z \geq 1$ oversatisfied by margin 1
- Hence, can deduce constraint $x + y + z \geq 2$

Strengthening rule (imported by [DG02] from operations research)

- Suppose $\ell = 0 \Rightarrow \sum_i a_i \ell_i \geq A$ oversatisfied by amount K
- Then can deduce $K\ell + \sum_i a_i \ell_i \geq A + K$

In theory, can recover from bad encodings (e.g., CNF)
In practice, seems inefficient and hard to get to work...

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

But only get from resolution

$$6y + 4z + 2w \geq 4$$

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

But only get from resolution + saturation

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

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

But only get from resolution + saturation + division

$$2y + 2z + \ w \geq 2$$

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

But only get from resolution + saturation + division

$$2y + 2z + \ w \geq 2$$

"Fusion resolution" [Goc17]

$$\frac{a\ell + \sum_i b_i\ell_i \geq B \qquad a\overline{\ell} + \sum_i b_i\ell_i \geq B'}{\sum_i b_i\ell_i \geq \min\{B, B'\}}$$

No obvious way for cutting planes to immediately derive this
Shows up in some tricky benchmarks in [EGNV18]

Some PB Solving Challenges I: Input Format

1. 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]?
2. CNF: How to go beyond conflict-driven clause learning CDCL for decision problems encoded in CNF?
3. Cardinality constraint detection: Proposed as preprocessing [BLLM14] or inprocessing [EN20] — not yet competitive in practice
4. Robustness: Make PB solvers less sensitive to presence of extra constraints (anecdotally, CDCL solvers seem more stable)

Some PB Solving Challenges II: Conflict Analysis

1. Choice of Boolean rule:
   - Division, saturation, or select adaptively?
   - Or some other cut rule from ILP?
   - Try to avoid irrelevant literals? [LMMW20]
2. Many more degrees of freedom than in CDCL:
   - Skip resolution steps when slack very negative?
   - How aggressively to weaken reason in reduction step? [LMW20]
   - 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?
3. Do constraint minimization à la [SB09, HS09]?
4. How to assess quality of learned constraints?
5. Theoretical potential and limitations poorly understood [VEG$^+$18]
   - Separations of subsystems of cutting planes?
   - In particular, is division reasoning stronger than saturation? [GNY19]

Some PB Solving Challenges III: Solver Heuristics

Many heuristics more or less copied from CDCL — maybe tailor more carefully to PB setting?

1. Variable selection: VSIDS [MMZ+01] or VMTF [Rya04] or something else?
2. Variable bumping: Consider different bumping score depending on
   - whether literal falsified,
   - whether literal cancels,
   - coefficient of literal and/or degree of constraint?
3. Phase saving: Standard as in [PD07], multiple phases [BF20], or something else?
4. Different "modes" for SAT-focused and UNSAT-focused search?

See [Wal20] for a first in-depth investigation of some of these questions

Some PB Solving Challenges IV: Efficiency and Correctness

1. Efficient unit propagation for PB constraints is a major challenge — latest news in [Dev20], but still much left to do
2. Efficient detection of assertiveness during conflict analysis
3. Efficient and concise proof logging for pseudo-Boolean solving (shameless self-plug: ongoing work on PB proof checker VeriPB [Ver19, GMN20b] in [EGMN20, GMN20a, GMM^{+}20, GN21])

Organization of This Tutorial

Part I: Pseudo-Boolean Preliminaries

Part II: Pseudo-Boolean Solving

Part III: Pseudo-Boolean Optimization

Part IV: Mixed Integer Linear Programming

Outline of Part III: Pseudo-Boolean Optimization

MaxSAT Problem

Pseudo-Boolean optimization and MaxSAT solving intimately connected, so let's do a detour and define MaxSAT

Weighted partial MaxSAT problem

Input: Soft clauses $C_1, \ldots, C_m$ with weights $w_i \in \mathbb{R}^+$, $i \in [m]$
Hard clauses $C_{m+1}, \ldots, C_M$

Goal: Find assignment ρ such that

- for all hard clauses $C_{m+1}, \ldots, C_M$ it holds that $\rho(C_j) = 1$
- ρ maximizes $\sum_{\rho(C_i)=1, i \in [m]} w_i$

- All hard clauses must be satisfied
- Maximize weight of satisfied soft clauses = Minimize penalty of falsified soft clauses
- Write $(C)_w$ for clause C with weight w ($w = \infty$ for hard clause)

From MaxSAT to Pseudo-Boolean Optimization

MaxSAT instance

$$(\overline{x})_5$$
$$(y \vee \overline{z})_4$$
$$(\overline{y} \vee z)_3$$
$$(x \vee y \vee z)_\infty$$
$$(x \vee \overline{y} \vee \overline{z})_\infty$$

From MaxSAT to Pseudo-Boolean Optimization

MaxSAT instance

$$(\overline{x})_5$$
$$(y \lor \overline{z})_4$$
$$(\overline{y} \lor z)_3$$
$$(x \lor y \lor z)_\infty$$
$$(x \lor \overline{y} \lor \overline{z})_\infty$$

PBO instance

$$\min\ 5w_1 + 4w_2 + 3w_3$$
$$w_1 + \overline{x} \geq 1$$
$$w_2 + y + \overline{z} \geq 1$$
$$w_3 + \overline{y} + z \geq 1$$
$$x + y + z \geq 1$$
$$x + \overline{y} + \overline{z} \geq 1$$

From MaxSAT to Pseudo-Boolean Optimization

MaxSAT instance	**PBO instance**
	$\min\ 5w_1 + 4w_2 + 3w_3$
$(\overline{x})_5$	$w_1 + \overline{x} \geq 1$
$(y \lor \overline{z})_4$	$w_2 + y + \overline{z} \geq 1$
$(\overline{y} \lor z)_3$	$w_3 + \overline{y} + z \geq 1$
$(x \lor y \lor z)_\infty$	$x + y + z \geq 1$
$(x \lor \overline{y} \lor \overline{z})_\infty$	$x + \overline{y} + \overline{z} \geq 1$

So-called blocking variable transformation
Variables w_i are blocking or relaxation variables

From MaxSAT to Pseudo-Boolean Optimization

MaxSAT instance	**PBO instance**
	$\min\ 5w_1 + 4w_2 + 3w_3$
$(\overline{x})_5$	$w_1 + \overline{x} \geq 1$
$(y \vee \overline{z})_4$	$w_2 + y + \overline{z} \geq 1$
$(\overline{y} \vee z)_3$	$w_3 + \overline{y} + z \geq 1$
$(x \vee y \vee z)_\infty$	$x + y + z \geq 1$
$(x \vee \overline{y} \vee \overline{z})_\infty$	$x + \overline{y} + \overline{z} \geq 1$

So-called blocking variable transformation
Variables w_i are blocking or relaxation variables

Optimal solution $\rho = \{x = 0, y = 1, z = 0\}$ with penalty 3

From Pseudo-Boolean Optimization to MaxSAT/WBO

"MaxSAT instance" but with PB constraints:
Weighted Boolean Optimization [MMP09]

From Pseudo-Boolean Optimization to MaxSAT/WBO

"MaxSAT instance" but with PB constraints:
Weighted Boolean Optimization [MMP09]

PBO instance

$$\begin{array}{ll} \min & \sum_{i=1}^{n} a_i w_i \\ & C_1 \\ & C_2 \\ & \vdots \\ & C_M \end{array}$$

From Pseudo-Boolean Optimization to MaxSAT/WBO

"MaxSAT instance" but with PB constraints:
Weighted Boolean Optimization [MMP09]

PBO instance

$$\min \ \sum_{i=1}^{n} a_i w_i$$

$$C_1$$

$$C_2$$

$$\vdots$$

$$C_M$$

MaxSAT/WBO instance

$$(\overline{w}_1)_{a_1}$$

$$\vdots$$

$$(\overline{w}_n)_{a_n}$$

$$(C_1)_\infty$$

$$\vdots$$

$$(C_M)_\infty$$

Flavours of MaxSAT

- Partial MaxSAT: Hard and soft clauses
- MaxSAT: Only soft clauses
- Unweighted MaxSAT: All soft clauses have same weight (w.l.o.g. 1)
- Weighted MaxSAT: Different weights for soft clauses

4 different subproblems
But most current solvers deal with the most general problem

Main Approaches for MaxSAT Solving (and PBO)

1. Linear search SAT-UNSAT (LSU) (or model-improving search)
2. Core-guided search
3. Implicit hitting set (IHS) algorithm

Main Approaches for MaxSAT Solving (and PBO)

1. Linear search SAT-UNSAT (LSU) (or model-improving search)
2. Core-guided search
3. Implicit hitting set (IHS) algorithm

Will describe all of these algorithms as trying to

- minimize $\sum_{i=1}^{n} a_i w_i$
- subject to collection of PB constraints $F = C_1 \wedge \cdots \wedge C_m$ (possibly clausal)

Linear Search SAT-UNSAT (LSU) Algorithm

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

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 $\sum_{i=1}^{n} a_i w_i \leq -1 + \sum_{i=1}^{n} a_i \cdot \rho(w_i)$
5. Start over from the top

Linear Search Toy Example

1. Given PB formula F and objective function
$\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6$

Linear Search Toy Example

1. Given PB formula F and objective function $\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6$
2. Solver run on F returns $\rho_1 = \{w_1 = w_2 = w_3 = w_6 = 0; w_4 = w_5 = 1\}$

Linear Search Toy Example

1. Given PB formula F and objective function $\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6$
2. Solver run on F returns $\rho_1 = \{w_1 = w_2 = w_3 = w_6 = 0; w_4 = w_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add

$$w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \leq 8$$

Linear Search Toy Example

1. Given PB formula F and objective function $\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6$
2. Solver run on F returns $\rho_1 = \{w_1 = w_2 = w_3 = w_6 = 0; w_4 = w_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add

$$w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \leq 8$$

4. Solver run on F plus this new constraint returns $\rho_2 = \{w_1 = w_3 = w_5 = w_6 = 0; w_2 = w_4 = 1\}$

Linear Search Toy Example

1. Given PB formula F and objective function $\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6$
2. Solver run on F returns $\rho_1 = \{w_1 = w_2 = w_3 = w_6 = 0; w_4 = w_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add

$$w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \leq 8$$

4. Solver run on F plus this new constraint returns $\rho_2 = \{w_1 = w_3 = w_5 = w_6 = 0; w_2 = w_4 = 1\}$
5. Yields objective value 6, so add

$$w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \leq 5$$

Linear Search Toy Example

1. Given PB formula F and objective function $\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6$
2. Solver run on F returns $\rho_1 = \{w_1 = w_2 = w_3 = w_6 = 0; w_4 = w_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add
$$w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \leq 8$$
4. Solver run on F plus this new constraint returns $\rho_2 = \{w_1 = w_3 = w_5 = w_6 = 0; w_2 = w_4 = 1\}$
5. Yields objective value 6, so add
$$w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \leq 5$$
6. Now solver returns `UNSATISFIABLE`

Linear Search Toy Example

1. Given PB formula F and objective function $\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6$
2. Solver run on F returns $\rho_1 = \{w_1 = w_2 = w_3 = w_6 = 0; w_4 = w_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add

$$w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \leq 8$$

4. Solver run on F plus this new constraint returns $\rho_2 = \{w_1 = w_3 = w_5 = w_6 = 0; w_2 = w_4 = 1\}$
5. Yields objective value 6, so add

$$w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \leq 5$$

6. Now solver returns `UNSATISFIABLE`
7. Hence, minimum value of objective function subject to F is 6

Linear vs. Binary Search?

What if we run binary search instead of linear search?

Conventional wisdom appears to be that linear search is better

Linear vs. Binary Search?

What if we run binary search instead of linear search?
Conventional wisdom appears to be that linear search is better

Two possible explanations:

1. In theory, objective value could decrease by just 1 every time — in practice, tend to get much larger jumps
2. Potentially very different cost for
   - SAT calls (feasible instances where solver will find solution)
   - UNSAT calls (where solver determines no solution exists)

Linear vs. Binary Search?

What if we run binary search instead of linear search?
Conventional wisdom appears to be that linear search is better

Two possible explanations:

1. In theory, objective value could decrease by just 1 every time — in practice, tend to get much larger jumps
2. Potentially very different cost for
   - SAT calls (feasible instances where solver will find solution)
   - UNSAT calls (where solver determines no solution exists)

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 Search

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

Think first of this as MaxSAT instance with w_i as blocking variables

Core-Guided Search

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

Think first of this as MaxSAT instance with w_i as blocking variables

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

1. Run SAT solver with assumptions (pre-made decisions) $w_i = 0$ for all w_i in objective function

Core-Guided Search

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

Think first of this as MaxSAT instance with w_i as blocking variables

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

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

Core-Guided Search

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

Think first of this as MaxSAT instance with w_i as blocking variables

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

1. Run SAT solver with assumptions (pre-made decisions) $w_i = 0$ for all w_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $w_1 \vee \cdots \vee w_k$

Core-Guided Search

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

Think first of this as MaxSAT instance with w_i as blocking variables

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

1. Run SAT solver with assumptions (pre-made decisions) $w_i = 0$ for all w_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $w_1 \vee \cdots \vee w_k$
4. Means that soft clauses $\mathcal{K} = \{C_1, \ldots, C_k\}$ form a core — can't satisfy $\mathcal{K}$ and all hard constraints

Core-Guided Search

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

Think first of this as MaxSAT instance with w_i as blocking variables

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

1. Run SAT solver with assumptions (pre-made decisions) $w_i = 0$ for all w_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $w_1 \vee \cdots \vee w_k$
4. Means that soft clauses $\mathcal{K} = \{C_1, \ldots, C_k\}$ form a core — can't satisfy $\mathcal{K}$ and all hard constraints
5. Introduce new variables $z_j \Leftrightarrow \sum_{i=1}^{k} w_i \geq j$

Core-Guided Search

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

Think first of this as MaxSAT instance with w_i as blocking variables

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

1. Run SAT solver with assumptions (pre-made decisions) $w_i = 0$ for all w_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $w_1 \lor \cdots \lor w_k$
4. Means that soft clauses $\mathcal{K} = \{C_1, \ldots, C_k\}$ form a core — can't satisfy $\mathcal{K}$ and all hard constraints
5. Introduce new variables $z_j \Leftrightarrow \sum_{i=1}^{k} w_i \geq j$
6. Update objective function and val_{best} using $\sum_{i=1}^{k} w_i = 1 + \sum_{j=2}^{k} z_j$ to cancel at least one variable w_i

Core-Guided Search

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

Think first of this as MaxSAT instance with w_i as blocking variables

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

1. Run SAT solver with assumptions (pre-made decisions) $w_i = 0$ for all w_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $w_1 \vee \cdots \vee w_k$
4. Means that soft clauses $\mathcal{K} = \{C_1, \ldots, C_k\}$ form a core — can't satisfy $\mathcal{K}$ and all hard constraints
5. Introduce new variables $z_j \Leftrightarrow \sum_{i=1}^{k} w_i \geq j$
6. Update objective function and val_{best} using $\sum_{i=1}^{k} w_i = 1 + \sum_{j=2}^{k} z_j$ to cancel at least one variable w_i
7. Start over from top with updated objective function

Core-Guided Search for Pseudo-Boolean Optimization

- Original core-guided idea from [FM06]; see [MHL$^+$13] for survey

Core-Guided Search for Pseudo-Boolean Optimization

- Original core-guided idea from [FM06]; see [MHL^{+}13] for survey
- Core-guided search is kind of UNSAT-SAT linear search

Core-Guided Search for Pseudo-Boolean Optimization

- Original core-guided idea from [FM06]; see [MHL$^+$13] for survey
- Core-guided search is kind of UNSAT-SAT linear search
- Updating objective with new variables: OLL algorithm — used in
  - answer set programming [AKMS12]
  - MaxSAT solving [MDM14]

Core-Guided Search for Pseudo-Boolean Optimization

- Original core-guided idea from [FM06]; see [MHL^{+}13] for survey
- Core-guided search is kind of UNSAT-SAT linear search
- Updating objective with new variables: OLL algorithm — used in
  - answer set programming [AKMS12]
  - MaxSAT solving [MDM14]
- In general pseudo-Boolean setting, no need to think of w_i as markers for soft clauses — they are just literals in objective function

Core-Guided Search for Pseudo-Boolean Optimization

- Original core-guided idea from [FM06]; see [MHL+13] for survey
- Core-guided search is kind of UNSAT-SAT linear search
- Updating objective with new variables: OLL algorithm — used in
  - answer set programming [AKMS12]
  - MaxSAT solving [MDM14]
- In general pseudo-Boolean setting, no need to think of w_i as markers for soft clauses — they are just literals in objective function
- And rewriting very convenient — just use PB constraints without re-encoding

Core-Guided Search for Pseudo-Boolean Optimization

- Original core-guided idea from [FM06]; see [MHL+13] for survey
- Core-guided search is kind of UNSAT-SAT linear search
- Updating objective with new variables: OLL algorithm — used in
  - answer set programming [AKMS12]
  - MaxSAT solving [MDM14]
- In general pseudo-Boolean setting, no need to think of w_i as markers for soft clauses — they are just literals in objective function
- And rewriting very convenient — just use PB constraints without re-encoding
- Core-guided PB search: assume optimistically that objective can reach best imaginable value; derive contradiction if not possible

Core-Guided Search for Pseudo-Boolean Optimization

- Original core-guided idea from [FM06]; see [MHL$^+$13] for survey
- Core-guided search is kind of UNSAT-SAT linear search
- Updating objective with new variables: OLL algorithm — used in
  - answer set programming [AKMS12]
  - MaxSAT solving [MDM14]
- In general pseudo-Boolean setting, no need to think of w_i as markers for soft clauses — they are just literals in objective function
- And rewriting very convenient — just use PB constraints without re-encoding
- Core-guided PB search: assume optimistically that objective can reach best imaginable value; derive contradiction if not possible
- Let us try to explain by concrete example

Core-Guided Search Toy Example (1/3)

1. Given same PB formula F and objective function

$$\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \tag{1}$$

Core-Guided Search Toy Example (1/3)

1. Given same PB formula F and objective function

$$\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \tag{1}$$

2. Run solver on F with assumptions $w_i = 0$, $i \in [6]$

Core-Guided Search Toy Example (1/3)

1. Given same PB formula F and objective function

$$\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \tag{1}$$

2. Run solver on F with assumptions $w_i = 0$, $i \in [6]$
3. Suppose solver returns PB core constraint

$$3w_2 + 2w_3 + w_4 + w_5 \geq 4 \tag{2}$$

Core-Guided Search Toy Example (1/3)

1. Given same PB formula F and objective function

$$\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \tag{1}$$

2. Run solver on F with assumptions $w_i = 0$, $i \in [6]$
3. Suppose solver returns PB core constraint

$$3w_2 + 2w_3 + w_4 + w_5 \geq 4 \tag{2}$$

4. Round to nicer-to-work-with cardinality core constraint

$$w_2 + w_3 + w_4 + w_5 \geq 2 \tag{3}$$

Core-Guided Search Toy Example (1/3)

1. Given same PB formula F and objective function

$$\min w_1 + 2w_2 + 3w_3 + 4w_4 + 5w_5 + 6w_6 \tag{1}$$

2. Run solver on F with assumptions $w_i = 0$, $i \in [6]$
3. Suppose solver returns PB core constraint

$$3w_2 + 2w_3 + w_4 + w_5 \geq 4 \tag{2}$$

4. Round to nicer-to-work-with cardinality core constraint

$$w_2 + w_3 + w_4 + w_5 \geq 2 \tag{3}$$

5. Introduce new, fresh variables y_3 and y_4 and constraints

$$w_2 + w_3 + w_4 + w_5 = 2 + y_3 + y_4 \tag{4a}$$

$$y_3 \geq y_4 \tag{4b}$$

to enforce that y_j means "$w_2 + w_3 + w_4 + w_5 \geq j$"

Core-Guided Search Toy Example (2/3)

6. Multiply (4a) by 2 and add to (1) to cancel w_2 and get updated, equivalent objective function

$$w_1 + w_3 + 2w_4 + 3w_5 + 6w_6 + 2y_3 + 2y_4 + 4 \tag{5}$$

and update $val_{\text{best}} = 4$

Core-Guided Search Toy Example (2/3)

6. Multiply (4a) by 2 and add to (1) to cancel w_2 and get updated, equivalent objective function

$$w_1 + w_3 + 2w_4 + 3w_5 + 6w_6 + 2y_3 + 2y_4 + 4 \tag{5}$$

   and update $val_{\text{best}} = 4$

7. Run solver on F assuming all literals in (5) being 0

Core-Guided Search Toy Example (2/3)

6. Multiply (4a) by 2 and add to (1) to cancel w_2 and get updated, equivalent objective function

$$w_1 + w_3 + 2w_4 + 3w_5 + 6w_6 + 2y_3 + 2y_4 + 4 \tag{5}$$

   and update $val_{\text{best}} = 4$
7. Run solver on F assuming all literals in (5) being 0
8. Suppose solver returns the clausal core constraint

$$w_4 + w_5 + w_6 + y_3 \geq 1 \tag{6}$$

Core-Guided Search Toy Example (2/3)

6. Multiply (4a) by 2 and add to (1) to cancel w_2 and get updated, equivalent objective function

$$w_1 + w_3 + 2w_4 + 3w_5 + 6w_6 + 2y_3 + 2y_4 + 4 \tag{5}$$

and update $val_{\mathrm{best}} = 4$

7. Run solver on F assuming all literals in (5) being 0
8. Suppose solver returns the clausal core constraint

$$w_4 + w_5 + w_6 + y_3 \geq 1 \tag{6}$$

9. Introduce new variables z_2, z_3, z_4 and the constraints

$$w_4 + w_5 + w_6 + y_3 = 1 + z_2 + z_3 + z_4 \tag{7a}$$

$$z_2 \geq z_3 \tag{7b}$$

$$z_3 \geq z_4 \tag{7c}$$

to enforce that z_j means "$w_4 + w_5 + w_6 + y_3 \geq j$"

Core-Guided Search Toy Example (3/3)

10. Multiply (7a) by 2 and add to (5) to get 3rd equivalent objective

$$w_1 + w_3 + w_5 + 4w_6 + 2y_4 + 2z_2 + 2z_3 + 2z_4 + 6 \tag{8}$$

and update $val_{\text{best}} = 6$

Core-Guided Search Toy Example (3/3)

10. Multiply (7a) by 2 and add to (5) to get 3rd equivalent objective

$$w_1 + w_3 + w_5 + 4w_6 + 2y_4 + 2z_2 + 2z_3 + 2z_4 + 6 \tag{8}$$

   and update $val_{\text{best}} = 6$

11. For 3rd time run solver on F, assuming all literals in (8) being 0

Core-Guided Search Toy Example (3/3)

10. Multiply (7a) by 2 and add to (5) to get 3rd equivalent objective

$$w_1 + w_3 + w_5 + 4w_6 + 2y_4 + 2z_2 + 2z_3 + 2z_4 + 6 \qquad (8)$$

    and update $val_{\text{best}} = 6$

11. For 3rd time run solver on F, assuming all literals in (8) being 0
12. Suppose solver reports it is possible to achieve

$$\rho = \{w_1 = w_3 = w_5 = w_6 = y_4 = z_2 = z_3 = z_4 = 0\} \qquad (9)$$

Core-Guided Search Toy Example (3/3)

10. Multiply (7a) by 2 and add to (5) to get 3rd equivalent objective

$$w_1 + w_3 + w_5 + 4w_6 + 2y_4 + 2z_2 + 2z_3 + 2z_4 + 6 \tag{8}$$

and update $val_{\mathrm{best}} = 6$

11. For 3rd time run solver on F, assuming all literals in (8) being 0
12. Suppose solver reports it is possible to achieve

$$\rho = \{w_1 = w_3 = w_5 = w_6 = y_4 = z_2 = z_3 = z_4 = 0\} \tag{9}$$

13. Under assignment (9) the equality (4a) simplifies to

$$w_2 + w_4 = 2 + y_3 \tag{10}$$

which can hold only if $y_3 = 0$ and $w_2 = w_4 = 1$, and this also satisfies (7a). Hence, have recovered optimal solution 6 (as before)

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 — how to get the best of both worlds?

Improvements of Core-Guided Search (1/2)

Weight stratification [ABGL12]

Set only literals with largest weight in objective to $0 \Rightarrow$

1. More compact core; or
2. Decent solution found early on

Improvements of Core-Guided Search (1/2)

Weight stratification [ABGL12]

Set only literals with largest weight in objective to $0 \Rightarrow$

1. More compact core; or
2. Decent solution found early on

Independent cores [BJ17]

If found core constraint over $w_1, w_2, \ldots, w_k$, remove these literals from assumptions and immediately run solver again with remaining assumptions

Improvements of Core-Guided Search (1/2)

Weight stratification [ABGL12]
Set only literals with largest weight in objective to $0 \Rightarrow$

1. More compact core; or
2. Decent solution found early on

Independent cores [BJ17]
If found core constraint over $w_1, w_2, \ldots, w_k$, remove these literals from assumptions and immediately run solver again with remaining assumptions

Core boosting [BDS19]
Start with core-guided search to get good lower bound estimate;
then switch to linear search to find optimal solution

Improvements of Core-Guided Search (1/2)

Weight stratification [ABGL12]

Set only literals with largest weight in objective to $0 \Rightarrow$

1. More compact core; or
2. Decent solution found early on

Independent cores [BJ17]

If found core constraint over $w_1, w_2, \ldots, w_k$, remove these literals from assumptions and immediately run solver again with remaining assumptions

Core boosting [BDS19]

Start with core-guided search to get good lower bound estimate; then switch to linear search to find optimal solution

Hybrid/interleaving search [ADMR15]

Switch back and forth repeatedly between core-guided and linear search — cumbersome in CNF-based solver, but fairly cheap (and efficient) in native pseudo-Boolean solver [DGD+21]

Improvements of Core-Guided Search (2/2)

Core minimization
In CDCL-based solver, try to get smaller core clauses. For PB solver, not so clear how to do this (constraint minimization also interesting problem in general for PB conflict analysis)

Improvements of Core-Guided Search (2/2)

Core minimization
In CDCL-based solver, try to get smaller core clauses. For PB solver, not so clear how to do this (constraint minimization also interesting problem in general for PB conflict analysis)

Lazy variables [MJML14, DGD$^+$21]
For real-world instances, rewriting of objective function can introduce huge numbers of new variables, slowing down the solver — so don't introduce all variables in one go but only lazily as needed

Improvements of Core-Guided Search (2/2)

Core minimization
In CDCL-based solver, try to get smaller core clauses. For PB solver, not so clear how to do this (constraint minimization also interesting problem in general for PB conflict analysis)

Lazy variables [MJML14, DGD^{+}21]
For real-world instances, rewriting of objective function can introduce huge numbers of new variables, slowing down the solver — so don't introduce all variables in one go but only lazily as needed

Inference strength of core-guided search?

- Extension variables very strong in theory, but hard to use in practice
- Core-guided search provides principled way of introducing them
- Can we characterize the power of this method?

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**

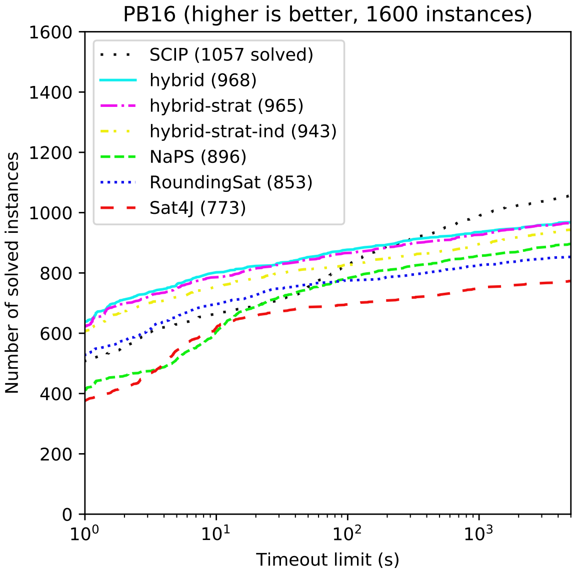
Significant improvement over PB state of the art, but MIP still better

Core-Guided PB Solving for PB16 benchmarks [DGD+21]

Cumulative plot for solver performance on PB16 optimization benchmarks

Also including

- weight stratification (strat)
- independent cores (ind)



Implicit Hitting Set (IHS) Algorithm (1/2)

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

As in core-guided search, use solving with assumptions, but maintain collection $\mathcal{K}$ of learned core clauses

$$\begin{aligned}
C_1 &\doteq w_{1,1} \vee w_{1,2} \vee \cdots \vee w_{1,k_s} \\
C_2 &\doteq w_{2,1} \vee w_{2,2} \vee \cdots \vee w_{2,k_s} \\
&\vdots \\
C_s &\doteq w_{s,1} \vee w_{s,2} \vee \cdots \vee w_{s,k_s}
\end{aligned}$$

Implicit Hitting Set (IHS) Algorithm (2/2)

Set $\mathcal{K} = \emptyset$ and repeat the following:

1. Compute minimum hitting set for $\mathcal{K}$, i.e., $W = \{w_i\}$ s.t.
   - $W \cap C \neq \emptyset$ for all $C \in \mathcal{K}$ (W is hitting set)
   - $\sum_{w_i \in W} w_i$ minimal among W with this property.
2. Run the solver with assumptions $\{w_i = 1 \mid w_i \in W\} \cup \{w_j = 0 \mid w_j \notin W\}$
3. If solver found solution, it must be optimal (since hitting set is optimal), so return solution with value $\sum_{w_i \in W} w_i$
4. Otherwise, solver returns new core C_{s+1} — add it to $\mathcal{K}$ and start over from top

More About the Hitting Sets

- Minimality is actually not needed except in the very final step
- Save time by computing "decent" hitting sets earlier on in the search
- How to find hitting set?
- This is itself a pseudo-Boolean optimization problem
  [as discussed in Part I of tutorial]
  - Run MIP solver
  - Or PB solver

Implicit Hitting Set vs. Core-Guided

- IHS and core-guided approaches for MaxSAT seem orthogonal [Bac21]
- For MaxSAT problems with many interchangeable soft clauses core-guided seems better (i.e., when it is not important exactly which of these clauses end up in core)
- For MaxSAT problems with many distinct weights, IHS seems better

Relation between IHS and core-guided search?

Provide a more precise theoretical comparison of IHS and core-guided search with simulations and/or separations

(Some theoretical work on related problems in, e.g., [FMSV20, MIB$^+$19])

Some More Open Questions

Combine IHS and core-guided search in MaxSAT solving?

Recent work on this in [BBP20]

Some More Open Questions

Combine IHS and core-guided search in MaxSAT solving?

Recent work on this in [BBP20]

Combine IHS with pseudo-Boolean optimization?

- In PB setting, cores will not be subsets of clauses but PB constraints $C_1, \ldots, C_s$ over objective function literals
- Hitting set W is partial assignment guaranteed to satisfy all constraints $C_1, \ldots, C_s$
- Want to find minimum-cost set W of literals (w.r.t. objective function) with this property
- Not implemented in native PB solvers (to best of my knowledge)

Organization of This Tutorial

Part I: Pseudo-Boolean Preliminaries

Part II: Pseudo-Boolean Solving

Part III: Pseudo-Boolean Optimization

Part IV: Mixed Integer Linear Programming

Outline of Part IV: Mixed Integer Linear Programming

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 best for optimization

Two Differences Compared to SAT/PB

Academia vs. industry

- Best solvers are commercial and closed-source
- E.g., CPLEX [CPL], GUROBI [Gur], and XPRESS [Xpr]
- Academic solvers like SCIP [SCI] 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], GUROBI [Gur], and XPRESS [Xpr]
- Academic solvers like SCIP [SCI] 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

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 $j = n+1, \ldots, N$

- Fast to solve (just linear programming)
- LP solution x^* yields lower bound
- Or, if x^* "accidentally" feasible, have optimal solution
- Use simplex algorithm — will have 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

PB 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

PB 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
  - 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 $C \subset I$ 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 $C \subset I$ 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

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

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 $C \subset I$ 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

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

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

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

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

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

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

Concretely, MIR cut with divisor 3 applied on

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

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

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

Concretely, MIR cut with divisor 3 applied on

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

yields

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

For comparison, standard division by 3 and multiplication by 2 produces

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

Presolving

Topic for a separate talk (well, like everything else in this part. . .)
Important for performance (but not as important as in CDCL?)

Presolving

Topic for a separate talk (well, like everything else in this part. . .)
Important for performance (but not as important as in CDCL?)

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

Topic for a separate talk (well, like everything else in this part. . .)
Important for performance (but not as important as in CDCL?)

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 $\Rightarrow$ indistinguishable from 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 $\Rightarrow$ indistinguishable from resolution on CNF encoding $\Rightarrow$ exponential lower bound in [Hak85] applies

A bit stupid example. . . solved immediately, since LP relaxation infeasible

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 $\Rightarrow$ indistinguishable from resolution on CNF encoding $\Rightarrow$ exponential lower bound in [Hak85] applies

A bit stupid example. . . solved immediately, since LP relaxation infeasible

But can find other, more interesting benchmarks where MIP conflict analysis seems to 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 best variable

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 best variable

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 best variable

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
- 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
- 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
(Interestingly, this turns ILP into 0-1 ILP subproblem)

And More. . .

1. Decomposition
   - Branch-and-price / column generation
   - Bender's decomposition
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]
    - 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]
  - 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 solvers
- Though known that even best commercial solvers sometimes give wrong results
- Some work on proof logging in [CGS17] — 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 (borrow ideas from native pseudo-Boolean solvers?)
3. Provide rigorous understanding of MIP solver performance
4. Develop families of theory benchmarks and computational complexity results for them (cf. SAT solving and proof complexity [BN21])
5. Steal best MIP ideas 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 (borrow ideas from native pseudo-Boolean solvers?)
3. Provide rigorous understanding of MIP solver performance
4. Develop families of theory benchmarks and computational complexity results for them (cf. SAT solving and proof complexity [BN21])
5. Steal best MIP ideas and use for pseudo-Boolean solving?! [next and final topic]

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?

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
    - PB 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
    - PB 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
    - PB 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
   - PB 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
   - PB 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 call shows LP relaxation infeasible under current trail?

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

Backtracking from LP Infeasibility?

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

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

More subtle issue:

- Efficient LP solvers use inexact floating-point arithmetic
- How to incorporate 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 Attempted PB-LP Integration [DGN21]

1. Interleave incremental LP solving within conflict-driven PB search
   - Limit LP solver 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 Attempted PB-LP Integration [DGN21]

1. Interleave incremental LP solving within conflict-driven PB search
   - Limit LP solver 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 $\Rightarrow$ linear combination of constraints violated by trail
   - 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 Attempted PB-LP Integration [DGN21]

1. Interleave incremental LP solving within conflict-driven PB search
   - Limit LP solver 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 $\Rightarrow$ linear combination of constraints violated by trail
   - 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 side and LP side
   - Try to use LP solution to guide PB search (e.g., variable decisions)

Report on Attempted PB-LP Integration [DGN21]

1. Interleave incremental LP solving within conflict-driven PB search
   - Limit LP solver 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 $\Rightarrow$ linear combination of constraints violated by trail
   - 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 side 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 $F\restriction_\rho$ infeasible
Then $\exists$ coefficients $k_i \in \mathbb{N}$ such that linear combination

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

is violated by ρ, i.e.,

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

Observed in [MM04] that $\sum_{i=1}^{m} k_i \cdot C_i$ is valid starting point for pseudo-Boolean 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 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 C to "wrong value"
2. Apply division/saturation to generate 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 C to "wrong value"
2. Apply division/saturation to generate 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 C to "wrong value"
2. Apply division/saturation to generate 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 C to "wrong value"
2. Apply division/saturation to generate 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 C to "wrong value"
2. Apply division/saturation to generate 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 C to "wrong value"
2. Apply division/saturation to generate 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
- Incurs exponential loss in reasoning 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
- Incurs exponential loss in reasoning power compared to operating on actual linear constraints (follows from [BKS04, CCT87, Hak85])

Arithmetic

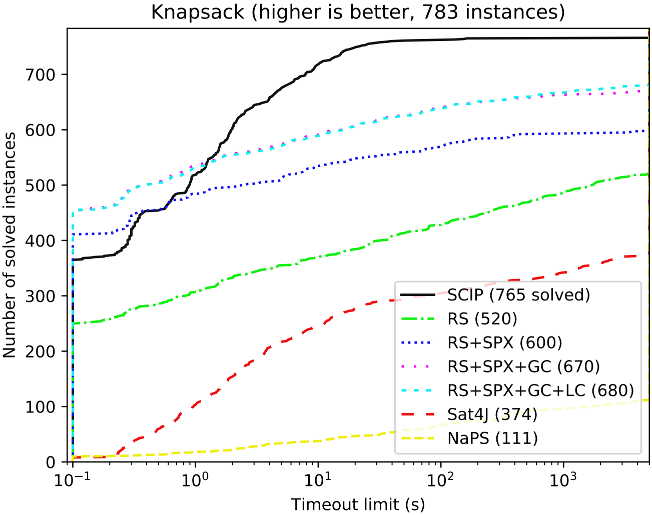
- SCIP uses floating point
- Reasoning steps in PB solver computed with exact integer arithmetic
- No issues with possible rounding errors

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
   - ... But SCIP is hard to beat

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
    - ... But SCIP is hard to beat
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
   - ... But SCIP is hard to beat
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

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 constraint 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 constraint conflicts **not** useful?
3. Make more intelligent use in PB 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 constraint conflicts **not** useful?
3. Make more intelligent use in PB 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 constraint conflicts **not** useful?
3. Make more intelligent use in PB 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 pseudo-Boolean conflict analysis

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

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

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

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

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

6. Combine LP solver with core-guided search or IHS approach
7. Improve pseudo-Boolean search
   - ROUNDINGSAT with LP integration or core-guided search seems to be state of the art for PB solving
   - But solver much better on unsatisfiable instances (proving optimality) than on satisfiable ones (finding solutions)
8. Export pseudo-Boolean conflict analysis to MIP

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

6. Combine LP solver with core-guided search or IHS approach
7. Improve pseudo-Boolean search
   - RoundingSat with LP integration or core-guided search seems to be state of the art for PB solving
   - But solver much better on unsatisfiable instances (proving optimality) than on satisfiable ones (finding solutions)
8. Export pseudo-Boolean conflict analysis to MIP
9. Use hybrid PB-LP solver to solve 0-1 MIP problems
   - PB solver decides on Boolean variables and propagates
   - LP solver takes care of real-valued variables

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?
- Some highly nontrivial challenges regarding
  - Algorithm design
  - Efficient implementation
  - Theoretical understanding
- But maybe also quite a bit of low-hanging fruit?
- And in any case lots of fun questions to work on! ☺

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?
- Some highly nontrivial challenges regarding
  - Algorithm design
  - Efficient implementation
  - Theoretical understanding
- But maybe also quite a bit of low-hanging fruit?
- And in any case lots of fun questions to work on! ☺

Thank you for your attention!

References I

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

[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.

[ADMR15] Mario Alviano, Carmine Dodaro, João P. Marques-Silva, and Francesco Ricca. Optimum stable model search: Algorithms and implementation. *Journal of Logic and Computation*, 30(4):863–897, August 2015.

References II

[AKMS12] Benjamin Andres, Benjamin Kaufmann, Oliver Matheis, and Torsten Schaub. Unsatisfiability-based optimization in clasp. In *Technical Communications of the 28th International Conference on Logic Programming (ICLP '12)*, volume 17 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 211–221, September 2012.

[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.

[Bac21] Fahiem Bacchus. Personal communication, 2021.

[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.

[BB03] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP '03)*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, September 2003.

References III

[BBP20] Jeremias Berg, Fahiem Bacchus, and Alex Poole. Abstract cores in implicit hitting set MaxSat solving. In *Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing (SAT '20)*, volume 12178 of *Lecture Notes in Computer Science*, pages 277–294. Springer, July 2020.

[BBR09] Olivier Bailleux, Yacine Boufkhad, and Olivier Roussel. New encodings of pseudo-Boolean constraints into CNF. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT '09)*, volume 5584 of *Lecture Notes in Computer Science*, pages 181–194. Springer, June 2009.

[BDS19] Jeremias Berg, Emir Demirović, and Peter J. Stuckey. Core-boosted linear search for incomplete MaxSAT. In *Proceedings of the 16th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR '19)*, volume 11494 of *Lecture Notes in Computer Science*, pages 39–56. Springer, June 2019.

[BF20] Armin Biere and Mathias Fleury. Chasing target phases. Presented at the workshop *Pragmatics of SAT 2020*. Paper available at `http://fmv.jku.at/papers/BiereFleury-POS20.pdf`, July 2020.

References IV

[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.

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

[BJ17] Jeremias Berg and Matti Järvisalo. Weight-aware core extraction in SAT-based MaxSAT solving. In *Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming (CP '17)*, volume 10416 of *Lecture Notes in Computer Science*, pages 652–670. Springer, August 2017.

[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*.

[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.

References V

[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.

[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.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

[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 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.

References VI

[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*.

[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.

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

[Dev20] Jo Devriendt. Watched propagation of 0-1 integer linear constraints. 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 160–176. Springer, September 2020.

[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.

References VII

[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.

[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*.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[EGMN20] Jan Elffers, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Justifying all differences using pseudo-Boolean reasoning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*, pages 1486–1494, February 2020.

References VIII

[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.

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

References IX

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

[FMSV20] Yuval Filmus, Meena Mahajan, Gaurav Sood, and Marc Vinyals. MaxSAT resolution and subcube sums. In *Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing (SAT '20)*, volume 12178 of *Lecture Notes in Computer Science*, pages 295–311. Springer, July 2020.

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

[GMN20a] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Subgraph isomorphism meets cutting planes: Solving with certified solutions. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*, pages 1134–1140, July 2020.

References X

[GMN20b] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. VeriPB: The easy way to make your combinatorial search algorithm trustworthy. Presented at the workshop *From Constraint Programming to Trustworthy AI* at the *26th International Conference on Principles and Practice of Constraint Programming (CP '20)*. Paper available at `http://www.cs.ucc.ie/~bg6/cptai/2020/papers/CPTAI_2020_paper_2.pdf`, September 2020.

[GN21] Stephan Gocht and Jakob Nordström. Certifying parity reasoning efficiently using pseudo-Boolean proofs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*, February 2021. To appear.

[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.

[Goc17] Stephan Gocht. Personal communication, 2017.

[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/`.

References XI

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

[Hoo88] John N. Hooker. Generalized resolution and cutting planes. *Annals of Operations Research*, 12(1):217–239, December 1988.

[Hoo92] John N. Hooker. Generalized resolution for 0-1 linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 6(1):271–286, March 1992.

[HS09] Hyojung Han and Fabio Somenzi. On-the-fly clause improvement. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT '09)*, volume 5584 of *Lecture Notes in Computer Science*, pages 209–222. Springer, July 2009.

[JdM13] Dejan Jovanovic and Leonardo de Moura. Cutting to the chase solving linear integer arithmetic. *Journal of Automated Reasoning*, 51(1):79–108, June 2013. Preliminary version in *CADE-23* 2011.

References XII

[JMM15] Saurabh Joshi, Ruben Martins, and Vasco M. Manquinho. Generalized totalizer encoding for pseudo-Boolean constraints. In *Proceedings of the 21st International Conference on Principles and Practice of Constraint Programming (CP '15)*, volume 9255 of *Lecture Notes in Computer Science*, pages 200–209. Springer, August-September 2015.

[KMP13] Thorsten Koch, Alexander Martin, and Marc E. Pfetsch. Progress in academic computational integer programming. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*, pages 483–506. Springer, 2013.

[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.

[LMMW20] Daniel Le Berre, Pierre Marquis, Stefan Mengel, and Romain Wallon. On irrelevant literals in pseudo-Boolean constraint learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*, pages 1148–1154, July 2020.

References XIII

[LMW20] Daniel Le Berre, Pierre Marquis, and Romain Wallon. On weakening strategies for PB solvers. In *Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing (SAT '20)*, volume 12178 of *Lecture Notes in Computer Science*, pages 322–331. Springer, July 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.

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

[MHL+13] António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João P. Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, October 2013.

References XIV

[MIB$^+$19] António Morgado, Alexey Ignatiev, María Luisa Bonet, João P. Marques-Silva, and Samuel R. Buss. DRMaxSAT with MaxHS: First contact. In *Proceedings of the 22nd International Conference on Theory and Applications of Satisfiability Testing (SAT '19)*, volume 11628 of *Lecture Notes in Computer Science*, pages 239–249. Springer, July 2019.

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

[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`.

[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.

References XV

[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.

[MMP09] Vasco M. Manquinho, João P. Marques-Silva, and Jordi Planes. Algorithms for weighted Boolean optimization. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT '09)*, volume 5584 of *Lecture Notes in Computer Science*, pages 495–508. Springer, June 2009.

[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.

[MN14] Mladen Mikša and Jakob Nordström. Long proofs of (seemingly) simple formulas. 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 121–137. Springer, July 2014.

[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.

References XVI

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

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

[PD07] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In *Proceedings of the 10th International Conference on Theory and Applications of Satisfiability Testing (SAT '07)*, volume 4501 of *Lecture Notes in Computer Science*, pages 294–299. Springer, May 2007.

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

[Rya04] Lawrence Ryan. Efficient algorithms for clause-learning SAT solvers. Master's thesis, Simon Fraser University, February 2004. Available at `https://www.cs.sfu.ca/~mitchell/papers/ryan-thesis.ps`.

[SB09] Niklas Sörensson and Armin Biere. Minimizing learned clauses. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT '09)*, volume 5584 of *Lecture Notes in Computer Science*, pages 237–243. Springer, July 2009.

References XVII

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

[Sin05] Carsten Sinz. Towards an optimal CNF encoding of Boolean cardinality constraints. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP '05)*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, October 2005.

[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*.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

References XVIII

[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.

[Ver19] VeriPB: Verifier for pseudo-Boolean proofs. `https://doi.org/10.5281/zenodo.3548581`, 2019.

[Wal20] Romain Wallon. *Pseudo-Boolean Reasoning and Compilation*. PhD thesis, Université d'Artois, 2020.

[Wil76] H. P. Williams. Fourier-Motzkin elimination extension to integer programming problems. *Journal of Combinatorial Theory, Series A*, 21(1):118–123, July 1976.

[Wol08] Laurence A. Wolsey. Mixed integer programming. In *Wiley Encyclopedia of Computer Science and Engineering*. Wiley, 2008. Available at `https://doi.org/10.1002/9780470050118.ecse244`.

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