Tutorial on Conflict-Driven Pseudo-Boolean Optimization

Jakob Nordström

University of Copenhagen and Lund University

SAT + SMT Winter School
Chennai, India
December 15, 2022

Outline of Lecture on 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 $w_i \in \mathbb{N}^+$, $i \in [m]$
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

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

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

$$\min \; 5b_1 + 4b_2 + 3b_3$$
$$b_1 + \overline{x} \geq 1$$
$$b_2 + y + \overline{z} \geq 1$$
$$b_3 + \overline{y} + z \geq 1$$
$$x + y + z \geq 1$$
$$x + \overline{y} + \overline{z} \geq 1$$

From MaxSAT to Pseudo-Boolean Optimization

MaxSAT instance	**PBO instance**
	$\min\ 5b_1 + 4b_2 + 3b_3$
$(\overline{x})_5$	$b_1 + \overline{x} \geq 1$
$(y \vee \overline{z})_4$	$b_2 + y + \overline{z} \geq 1$
$(\overline{y} \vee z)_3$	$b_3 + \overline{y} + z \geq 1$
$(x \vee y \vee z)_\infty$	$x + y + z \geq 1$
$(x \vee \overline{y} \vee \overline{z})_\infty$	$x + \overline{y} + \overline{z} \geq 1$

So-called blocking variable transformation

Variables b_i are blocking or relaxation variables

From MaxSAT to Pseudo-Boolean Optimization

MaxSAT instance	PBO instance
	$\min \; 5b_1 + 4b_2 + 3b_3$
$(\overline{x})_5$	$b_1 + \overline{x} \geq 1$
$(y \vee \overline{z})_4$	$b_2 + y + \overline{z} \geq 1$
$(\overline{y} \vee z)_3$	$b_3 + \overline{y} + z \geq 1$
$(x \vee y \vee z)_\infty$	$x + y + z \geq 1$
$(x \vee \overline{y} \vee \overline{z})_\infty$	$x + \overline{y} + \overline{z} \geq 1$

So-called blocking variable transformation
Variables b_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

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

$$C_1$$

$$C_2$$

$$\vdots$$

$$C_M$$

From Pseudo-Boolean Optimization to MaxSAT/WBO

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

PBO instance	**MaxSAT/WBO instance**
$\min \; \sum_{i=1}^{n} w_i \ell_i$	
C_1	
C_2	$(\bar{\ell}_1)_{w_1}$
$\vdots$	$\vdots$
C_M	$(\bar{\ell}_n)_{w_n}$
	$(C_1)_\infty$
	$\vdots$
	$(C_M)_\infty$

Flavours of MaxSAT

- Partial MaxSAT: Hard and soft clauses
- MaxSAT: Only soft clauses
- Unweighted MaxSAT: Same weight for soft clauses (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 $\sum_{i=1}^{n} w_i \ell_i$
- subject to collection of PB constraints $F = C_1 \wedge \cdots \wedge C_m$ (possibly clausal)

Linear Search SAT-UNSAT (LSU) Algorithm

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

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

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

Linear Search Toy Example

1. Given PB formula F and objective function
$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$

Linear Search Toy Example

1. Given PB formula F and objective function $\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$
2. Solver run on F returns $\rho_1 = \{x_1 = x_2 = x_3 = x_6 = 0; x_4 = x_5 = 1\}$

Linear Search Toy Example

1. Given PB formula F and objective function $\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$
2. Solver run on F returns $\rho_1 = \{x_1 = x_2 = x_3 = x_6 = 0; x_4 = x_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 8$$

Linear Search Toy Example

1. Given PB formula F and objective function $\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$
2. Solver run on F returns $\rho_1 = \{x_1 = x_2 = x_3 = x_6 = 0; x_4 = x_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 8$$
4. Solver run on F plus this new constraint returns $\rho_2 = \{x_1 = x_3 = x_5 = x_6 = 0; x_2 = x_4 = 1\}$

Linear Search Toy Example

1. Given PB formula F and objective function $\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$
2. Solver run on F returns $\rho_1 = \{x_1 = x_2 = x_3 = x_6 = 0; x_4 = x_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 8$$
4. Solver run on F plus this new constraint returns $\rho_2 = \{x_1 = x_3 = x_5 = x_6 = 0; x_2 = x_4 = 1\}$
5. Yields objective value 6, so add
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 5$$

Linear Search Toy Example

1. Given PB formula F and objective function $\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$
2. Solver run on F returns $\rho_1 = \{x_1 = x_2 = x_3 = x_6 = 0; x_4 = x_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 8$$
4. Solver run on F plus this new constraint returns $\rho_2 = \{x_1 = x_3 = x_5 = x_6 = 0; x_2 = x_4 = 1\}$
5. Yields objective value 6, so add
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 5$$
6. Now solver returns `UNSATISFIABLE`

Linear Search Toy Example

1. Given PB formula F and objective function $\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$
2. Solver run on F returns $\rho_1 = \{x_1 = x_2 = x_3 = x_6 = 0; x_4 = x_5 = 1\}$
3. Yields objective value $0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 0 = 9$, so add
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 8$$
4. Solver run on F plus this new constraint returns $\rho_2 = \{x_1 = x_3 = x_5 = x_6 = 0; x_2 = x_4 = 1\}$
5. Yields objective value 6, so add
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 5$$
6. Now solver returns `UNSATISFIABLE`
7. Hence, minimum value of objective function subject to F is 6

CNF Encodings

For SAT solver, need CNF encoding of solution-improving constraint $\sum_{i=1}^{n} w_i \ell_i \leq -1 + \sum_{i=1}^{n} w_i \cdot \rho(\ell_i)$

Lots of work on how to do this in smart ways (with [PRB18] maybe best right now)

For pseudo-Boolean solver, no re-encoding needed

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

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

Core-Guided Search

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

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

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

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

Core-Guided Search

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

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

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

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

Core-Guided Search

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

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

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

1. Run SAT solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns SATISFIABLE, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $\ell_1 \vee \cdots \vee \ell_k$

Core-Guided Search

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

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

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

1. Run SAT solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $\ell_1 \vee \cdots \vee \ell_k$
4. Means that soft clauses $K = \{C_1, \ldots, C_k\}$ form a core — can't satisfy K and all hard constraints

Core-Guided Search

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

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

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

1. Run SAT solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns `SATISFIABLE`, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $\ell_1 \vee \cdots \vee \ell_k$
4. Means that soft clauses $K = \{C_1, \ldots, C_k\}$ form a core — can't satisfy K and all hard constraints
5. Introduce new variables $z_j \Leftrightarrow \sum_{i=1}^{k} \ell_i \geq j$

Core-Guided Search

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

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

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

1. Run SAT solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns SATISFIABLE, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $\ell_1 \lor \cdots \lor \ell_k$
4. Means that soft clauses $K = \{C_1, \ldots, C_k\}$ form a core — can't satisfy K and all hard constraints
5. Introduce new variables $z_j \Leftrightarrow \sum_{i=1}^{k} \ell_i \geq j$
6. Update objective function and val_{best} using $\sum_{i=1}^{k} \ell_i = 1 + \sum_{j=2}^{k} z_j$ to cancel at least one literal ℓ_i

Core-Guided Search

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

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

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

1. Run SAT solver with assumptions (pre-made decisions) $\ell_i = 0$ for all ℓ_i in objective function
2. If solver returns SATISFIABLE, output val_{best} and terminate
3. Otherwise learn clause over assumption variables, say $\ell_1 \vee \cdots \vee \ell_k$
4. Means that soft clauses $K = \{C_1, \ldots, C_k\}$ form a core — can't satisfy K and all hard constraints
5. Introduce new variables $z_j \Leftrightarrow \sum_{i=1}^{k} \ell_i \geq j$
6. Update objective function and val_{best} using $\sum_{i=1}^{k} \ell_i = 1 + \sum_{j=2}^{k} z_j$ to cancel at least one literal ℓ_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 ℓ_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 ℓ_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 ℓ_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 ℓ_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 (1/5)

1. Given same PB formula F and objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \tag{1}$$

Core-Guided Search Toy Example (1/5)

1. Given same PB formula F and objective function
$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \tag{1}$$
2. Set $val_{\text{best}} = 0$

Core-Guided Search Toy Example (1/5)

1. Given same PB formula F and objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \tag{1}$$

2. Set $val_{\text{best}} = 0$
3. Run solver on F with assumptions $x_1 = x_2 = \ldots = x_6 = 0$

Core-Guided Search Toy Example (1/5)

1. Given same PB formula F and objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \tag{1}$$

2. Set $val_{\text{best}} = 0$
3. Run solver on F with assumptions $x_1 = x_2 = \ldots = x_6 = 0$
4. Suppose solver returns PB core constraint

$$3x_2 + 2x_3 + x_4 + x_5 \geq 4 \tag{2}$$

Core-Guided Search Toy Example (1/5)

1. Given same PB formula F and objective function
$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \tag{1}$$
2. Set $val_{\text{best}} = 0$
3. Run solver on F with assumptions $x_1 = x_2 = \ldots = x_6 = 0$
4. Suppose solver returns PB core constraint
$$3x_2 + 2x_3 + x_4 + x_5 \geq 4 \tag{2}$$
5. Round to nicer-to-work-with cardinality core constraint
$$x_2 + x_3 + x_4 + x_5 \geq 2 \tag{3}$$

Core-Guided Search Toy Example (1/5)

1. Given same PB formula F and objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \tag{1}$$

2. Set $val_{\text{best}} = 0$
3. Run solver on F with assumptions $x_1 = x_2 = \ldots = x_6 = 0$
4. Suppose solver returns PB core constraint

$$3x_2 + 2x_3 + x_4 + x_5 \geq 4 \tag{2}$$

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

$$x_2 + x_3 + x_4 + x_5 \geq 2 \tag{3}$$

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

$$x_2 + x_3 + x_4 + x_5 = 2 + y_3 + y_4 \tag{4a}$$

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

to enforce that y_j means "$x_2 + x_3 + x_4 + x_5 \geq j$"

Core-Guided Search Toy Example (2/5)

7. Multiply (4a) by 2 to get

$$4 + 2y_3 + 2y_4 - 2x_2 - 2x_3 - 2x_4 - 2x_5 = 0$$

and add to objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$$

in (1) to cancel x_2 and get updated, equivalent objective function

$$\min x_1 + x_3 + 2x_4 + 3x_5 + 6x_6 + 2y_3 + 2y_4 + 4 \qquad (5)$$

Core-Guided Search Toy Example (2/5)

7. Multiply (4a) by 2 to get

$$4 + 2y_3 + 2y_4 - 2x_2 - 2x_3 - 2x_4 - 2x_5 = 0$$

and add to objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$$

in (1) to cancel x_2 and get updated, equivalent objective function

$$\min x_1 + x_3 + 2x_4 + 3x_5 + 6x_6 + 2y_3 + 2y_4 + 4 \tag{5}$$

8. Update $val_{\text{best}} = 4$

Core-Guided Search Toy Example (2/5)

7. Multiply (4a) by 2 to get

$$4 + 2y_3 + 2y_4 - 2x_2 - 2x_3 - 2x_4 - 2x_5 = 0$$

and add to objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$$

in (1) to cancel x_2 and get updated, equivalent objective function

$$\min x_1 + x_3 + 2x_4 + 3x_5 + 6x_6 + 2y_3 + 2y_4 + 4 \qquad (5)$$

8. Update $val_{\text{best}} = 4$
9. Run solver on F assuming all literals in (5) being 0

Core-Guided Search Toy Example (3/5)

10. Suppose solver returns the clausal core constraint

$$x_4 + x_5 + x_6 + y_3 \geq 1 \tag{6}$$

Core-Guided Search Toy Example (3/5)

10. Suppose solver returns the clausal core constraint

$$x_4 + x_5 + x_6 + y_3 \geq 1 \tag{6}$$

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

$$x_4 + x_5 + x_6 + y_3 = 1 + z_2 + z_3 + z_4 \tag{7a}$$
$$z_2 \geq z_3 \tag{7b}$$
$$z_3 \geq z_4 \tag{7c}$$

to enforce that z_j means "$x_4 + x_5 + x_6 + y_3 \geq j$"

Core-Guided Search Toy Example (4/5)

12. Multiply (7a) by 2 to get

$$2 + 2z_2 + 2z_3 + 2z_4 - 2x_4 - 2x_5 - 2x_6 - 2y_3 = 0$$

and add to rewritten objective

$$\min x_1 + x_3 + 2x_4 + 3x_5 + 6x_6 + 2y_3 + 2y_4 + 4$$

in (5) to get 3rd equivalent objective

$$\min x_1 + x_3 + x_5 + 4x_6 + 2y_4 + 2z_2 + 2z_3 + 2z_4 + 6 \tag{8}$$

Core-Guided Search Toy Example (4/5)

12. Multiply (7a) by 2 to get

$$2 + 2z_2 + 2z_3 + 2z_4 - 2x_4 - 2x_5 - 2x_6 - 2y_3 = 0$$

and add to rewritten objective

$$\min x_1 + x_3 + 2x_4 + 3x_5 + 6x_6 + 2y_3 + 2y_4 + 4$$

in (5) to get 3rd equivalent objective

$$\min x_1 + x_3 + x_5 + 4x_6 + 2y_4 + 2z_2 + 2z_3 + 2z_4 + 6 \tag{8}$$

13. Update $val_{\text{best}} = 6$

Core-Guided Search Toy Example (4/5)

12. Multiply (7a) by 2 to get

$$2 + 2z_2 + 2z_3 + 2z_4 - 2x_4 - 2x_5 - 2x_6 - 2y_3 = 0$$

and add to rewritten objective

$$\min x_1 + x_3 + 2x_4 + 3x_5 + 6x_6 + 2y_3 + 2y_4 + 4$$

in (5) to get 3rd equivalent objective

$$\min x_1 + x_3 + x_5 + 4x_6 + 2y_4 + 2z_2 + 2z_3 + 2z_4 + 6 \tag{8}$$

13. Update $val_{\text{best}} = 6$
14. For 3rd time run solver on F, assuming all literals in (8) being 0

Core-Guided Search Toy Example (5/5)

15. Suppose solver reports it is possible to achieve

$$\rho = \{x_1 = x_3 = x_5 = x_6 = y_4 = z_2 = z_3 = z_4 = 0\} \tag{9}$$

Core-Guided Search Toy Example (5/5)

15. Suppose solver reports it is possible to achieve

$$\rho = \{x_1 = x_3 = x_5 = x_6 = y_4 = z_2 = z_3 = z_4 = 0\} \tag{9}$$

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

$$x_2 + x_4 = 2 + y_3 \tag{10}$$

which can hold only if $y_3 = 0$ and $x_2 = x_4 = 1$, and this also satisfies (7a).

Core-Guided Search Toy Example (5/5)

15. Suppose solver reports it is possible to achieve

$$\rho = \{x_1 = x_3 = x_5 = x_6 = y_4 = z_2 = z_3 = z_4 = 0\} \tag{9}$$

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

$$x_2 + x_4 = 2 + y_3 \tag{10}$$

which can hold only if $y_3 = 0$ and $x_2 = x_4 = 1$, and this also satisfies (7a).

17. Hence, have recovered optimal solution yielding objective value 6 (as in LSU example before)

Properties of (Pure) Core-Guided Search

- Can get decent lower bounds on solution quickly
- Helps to cut off parts of search space “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

Disjoint cores [DB11, DB13, Sai15]

If found core constraint over $\ell_1, \ell_2, \ldots, \ell_k$, remove these literals and run solver again with remaining assumptions (or [BJ17] even better)

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

Disjoint cores [DB11, DB13, Sai15]

If found core constraint over $\ell_1, \ell_2, \ldots, \ell_k$, remove these literals and run solver again with remaining assumptions (or [BJ17] even better)

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

Disjoint cores [DB11, DB13, Sai15]
If found core constraint over $\ell_1, \ell_2, \ldots, \ell_k$, remove these literals and run solver again with remaining assumptions (or [BJ17] even better)

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 (e.g., [Mar10, MIM15])
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 (e.g., [Mar10, MIM15])
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 (e.g., [Mar10, MIM15])
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 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 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 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**

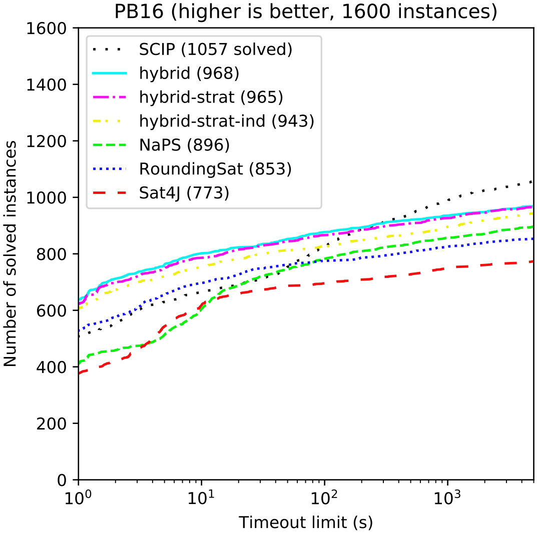
Significant improvement over PB state of the art, but MIP still better

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

Cumulative plot for solver performance on PB16 optimization benchmarks

Also including

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



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

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

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

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

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., $H = \{\ell_i\}$ s.t.
   - $H \cap C \neq \emptyset$ for all $C \in \mathcal{K}$ (H is hitting set)
   - $\sum_{\ell_i \in H} w_i$ minimal among H with this property.
2. Run the solver with assumptions $\{\ell_j = 0 \mid \ell_j \notin H\}$
3. If solver found solution, it must be optimal (since hitting set is optimal), so return solution with value $\sum_{\ell_i \in H} w_i$
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
  - Run IP solver [standard approach]
  - Or PB solver?
  - Or local search?!

Combine IHS with Pseudo-Boolean Optimization?

IHS and PB 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" H is partial assignment guaranteed to satisfy all constraints $C_1, \ldots, C_s$
- Want to find minimum-cost set H of literals (w.r.t. objective function) with this property

Combine IHS with Pseudo-Boolean Optimization?

IHS and PB 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" H is partial assignment guaranteed to satisfy all constraints $C_1, \ldots, C_s$
- Want to find minimum-cost set H of literals (w.r.t. objective function) with this property

- Explored by CoReO group in Helsinki in [SBJ21, SBJ22]
- Using RoundingSat version in [DGN21] as pseudo-Boolean "oracle solver"

IHS Algorithm for PB Optimization (Simplified)

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

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

1. Run optimization solver to minimize $\sum_{i=1}^{n} w_i \ell_i$ under $\mathcal{K}$, yielding solution ρ to objective variables
2. Run decision solver with assumptions ρ on decision problem F
3. If decision solver returns `SATISFIABLE`, we have found optimal solution extending ρ with value $\sum_{i=1}^{n} w_i \cdot \rho(\ell_i)$
4. Otherwise, solver returns new core C — add it to $\mathcal{K}$ and start over from top

IHS Toy Example (1/2)

1. Given same PB formula F and objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$$

IHS Toy Example (1/2)

1. Given same PB formula F and objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$$

2. For

$$\mathcal{K}_1 = \emptyset$$

optimization solver returns minimal solution
$\rho_1 = \{x_1 = x_2 = \ldots = x_6 = 0\}$

IHS Toy Example (1/2)

1. Given same PB formula F and objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$$

2. For

$$\mathcal{K}_1 = \emptyset$$

   optimization solver returns minimal solution
   $\rho_1 = \{x_1 = x_2 = \ldots = x_6 = 0\}$

3. Decision solver with assumptions ρ_1 returns PB core constraint

$$3x_2 + 2x_3 + x_4 + x_5 \geq 4$$

IHS Toy Example (1/2)

1. Given same PB formula F and objective function

$$\min x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6$$

2. For

$$\mathcal{K}_1 = \emptyset$$

optimization solver returns minimal solution
$\rho_1 = \{x_1 = x_2 = \ldots = x_6 = 0\}$

3. Decision solver with assumptions ρ_1 returns PB core constraint

$$3x_2 + 2x_3 + x_4 + x_5 \geq 4$$

4. For

$$\mathcal{K}_2 = \{3x_2 + 2x_3 + x_4 + x_5 \geq 4\}$$

optimization solver returns minimal solution
$\rho_2 = \{x_2 = x_3 = 1;\ x_1 = x_4 = \ldots = x_6 = 0\}$

IHS Toy Example (2/2)

5. Decision solver with assumptions ρ_2 returns PB core constraint

$$x_2 + x_4 + x_5 + x_6 \geq 2$$

IHS Toy Example (2/2)

5. Decision solver with assumptions ρ_2 returns PB core constraint

$$x_2 + x_4 + x_5 + x_6 \geq 2$$

6. For

$$\mathcal{K}_3 = \{3x_2 + 2x_3 + x_4 + x_5 \geq 4,\ x_2 + x_4 + x_5 + x_6 \geq 2\}$$

optimization solver returns minimal solution
$\rho_3 = \{x_2 = x_4 = 1;\ x_1 = x_3 = x_5 = x_6 = 0\}$

IHS Toy Example (2/2)

5. Decision solver with assumptions ρ_2 returns PB core constraint

$$x_2 + x_4 + x_5 + x_6 \geq 2$$

6. For

$$\mathcal{K}_3 = \{3x_2 + 2x_3 + x_4 + x_5 \geq 4,\ x_2 + x_4 + x_5 + x_6 \geq 2\}$$

   optimization solver returns minimal solution
   $\rho_3 = \{x_2 = x_4 = 1;\ x_1 = x_3 = x_5 = x_6 = 0\}$

7. Decision solver with assumptions ρ_2 returns `SATISFIABLE`

IHS Toy Example (2/2)

5. Decision solver with assumptions ρ_2 returns PB core constraint

$$x_2 + x_4 + x_5 + x_6 \geq 2$$

6. For

$$\mathcal{K}_3 = \{3x_2 + 2x_3 + x_4 + x_5 \geq 4,\ x_2 + x_4 + x_5 + x_6 \geq 2\}$$

   optimization solver returns minimal solution
   $\rho_3 = \{x_2 = x_4 = 1;\ x_1 = x_3 = x_5 = x_6 = 0\}$
7. Decision solver with assumptions ρ_2 returns SATISFIABLE
8. Hence, we have found an optimal solution with objective value 6 (as for LSU and core-guided search)

Comparison of Core-Guided Search and IHS

Suppose solver with assumptions returns core

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

Comparison of Core-Guided Search and IHS

Suppose solver with assumptions returns core

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

Core-guided search

- Introduce new variables by $x_1 + x_2 + x_3 + x_4 = 2 + y_3 + y_4$
- Ignore all x_i with smallest weight in objective in next call (get cancelled when objective rewritten)
- Instead assume that "somehow $x_1 + x_2 + x_3 + x_4 \leq 2$ holds" (i.e., $y_3 = 0$)

Comparison of Core-Guided Search and IHS

Suppose solver with assumptions returns core

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

Core-guided search

- Introduce new variables by $x_1 + x_2 + x_3 + x_4 = 2 + y_3 + y_4$
- Ignore all x_i with smallest weight in objective in next call (get cancelled when objective rewritten)
- Instead assume that "somehow $x_1 + x_2 + x_3 + x_4 \leq 2$ holds" (i.e., $y_3 = 0$)

IHS

- Add C to collection of cores $\mathcal{K}$
- Find concrete assignment satisfying all of $\mathcal{K}$ as cheaply as possible
- Try that assignment as starting point for next call to decision solver

Competitive Advantages of Core-Guided vs. IHS

- IHS and core-guided approaches for MaxSAT 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 the core)
- For MaxSAT problems with many distinct weights, IHS seems better

Competitive Advantages of Core-Guided vs. IHS

- IHS and core-guided approaches for MaxSAT 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 the core)
- For MaxSAT problems with many distinct weights, IHS seems better

Theoretical relations 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])

More Questions Aboute Core-Guided Search and IHS

1. Use assumptions $\{\ell_j = 0 \mid \ell_j \notin H\}$ or add also $\{\ell_i = 1 \mid \ell_i \in H\}$ for pseudo-Boolean IHS? (The latter done in [SBJ21, SBJ22])
2. Use cores in pseudo-Boolean core-guided search for objective reformulation without converting to cardinality constraints first?
3. How to do core minimization/strengthening in a PB setting?
4. Use something other than IP solver for pseudo-Boolean "hitting set problem"?
5. Abstract cores [BBP20] used to get IHS plus core counting variables — is it possible to do full integration of core-guided search and IHS in same solver in meaningful way?
6. Certify correctness using proof logging? [work in progress]

Summing up

- MaxSAT problems can be attacked with combination of powerful tools
  - Core-guided search
  - Implicit hitting set (IHS) search
  - Integer linear programming
- Approaches with complementary strengths — room for synergies?
- Lifting core-guided and IHS search to a pseudo-Boolean setting presents opportunities and challenges
  - No need for CNF re-encoding
  - More powerful pseudo-Boolean reasoning
  - But also slower than clausal reasoning
  - And more degrees of freedom in algorithm design
- Should provide rich pickings of low-hanging fruit!

Summing up

- MaxSAT problems can be attacked with combination of powerful tools
  - Core-guided search
  - Implicit hitting set (IHS) search
  - Integer linear programming
- Approaches with complementary strengths — room for synergies?
- Lifting core-guided and IHS search to a pseudo-Boolean setting presents opportunities and challenges
  - No need for CNF re-encoding
  - More powerful pseudo-Boolean reasoning
  - But also slower than clausal reasoning
  - And more degrees of freedom in algorithm design
- Should provide rich pickings of low-hanging fruit!

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.

[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.

[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.

[Bac21] Fahiem Bacchus. Personal communication, 2021.

References II

[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.

[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.

[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.

References III

[DB11] Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP '11)*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer, September 2011.

[DB13] Jessica Davies and Fahiem Bacchus. Exploiting the power of MIP solvers in MAXSAT. In *Proceedings of the 16th International Conference on Theory and Applications of Satisfiability Testing (SAT '13)*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer, July 2013.

[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)*, pages 3750–3758, February 2021.

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

References IV

[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.

[Mar10] João P. Marques-Silva. Minimal unsatisfiability: Models, algorithms and applications (Invited paper). In *Proceedings of the 40th IEEE International Symposium on Multiple-Valued Logic*, pages 9–14, May 2010.

[MDM14] António 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.

References V

[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.

[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.

[MIM15] António Morgado, Alexey Ignatiev, and João P. Marques-Silva. MSCG: Robust core-guided MaxSAT solving. *Journal on Satisfiability, Boolean Modeling and Computation*, 9(1):129–134, December 2015.

[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.

References VI

[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.

[PRB18] Tobias Paxian, Sven Reimer, and Bernd Becker. Dynamic polynomial watchdog encoding for solving weighted MaxSAT. 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 37–53. Springer, July 2018.

[Sai15] Paul Saikko. Re-implementing and extending a hybrid SAT-IP approach to maximum satisfiability. Master's thesis, University of Helsinki, November 2015. Available at `http://hdl.handle.net/10138/159186`.

[SBJ21] Pavel Smirnov, Jeremias Berg, and Matti Järvisalo. Pseudo-Boolean optimization by implicit hitting sets. In *Proceedings of the 27th International Conference on Principles and Practice of Constraint Programming (CP '21)*, volume 210 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:20, October 2021.

References VII

[SBJ22] Pavel Smirnov, Jeremias Berg, and Matti Järvisalo. Improvements to the implicit hitting set approach to pseudo-Boolean optimization. In *Proceedings of the 25th International Conference on Theory and Applications of Satisfiability Testing (SAT '22)*, volume 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:18, August 2022.