Tutorial on Conflict-Driven Pseudo-Boolean Optimization

Jakob Nordström

University of Copenhagen and Lund University

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

From Decision to Optimization Problems

So far in this series of tutorials we have talked about:

- deciding satisfiability of formulas in conjunctive normal form (CNF)
- generalizing this problem, and algorithms for it, to 0–1 integer linear programs (ILP)

From Decision to Optimization Problems

So far in this series of tutorials we have talked about:

- deciding satisfiability of formulas in conjunctive normal form (CNF)
- generalizing this problem, and algorithms for it, to 0–1 integer linear programs (ILP)

Many ways of extending this to (linear) optimization problems:

- Hard and soft constraints
  Maximum satisfiability (MaxSAT) and Weighted Boolean optimization (WBO)
- Constraints with preferences among solutions
  Weighted constraint satisfaction problems (WSCP)
- Optimization of linear objective function subject to CNF/pseudo-Boolean formula
  Pseudo-Boolean optimization (PBO) or 0–1 integer linear programming (0–1 ILP)

From Decision to Optimization Problems

So far in this series of tutorials we have talked about:

- deciding satisfiability of formulas in conjunctive normal form (CNF)
- generalizing this problem, and algorithms for it, to 0–1 integer linear programs (ILP)

Many ways of extending this to (linear) optimization problems:

- Hard and soft constraints
  Maximum satisfiability (MaxSAT) and Weighted Boolean optimization (WBO)
- Constraints with preferences among solutions
  Weighted constraint satisfaction problems (WSCP)
- Optimization of linear objective function subject to CNF/pseudo-Boolean formula
  Pseudo-Boolean optimization (PBO) or 0–1 integer linear programming (0–1 ILP)

We will focus on MaxSAT and PBO viewed as 0–1 ILP minimization problems

Recap: Some Problems Expressed as PBO (1/2)

Input:

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

Recap: Some Problems Expressed as PBO (1/2)

Input:

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

Weighted maximum clique

$$\begin{aligned} &\min \; -\textstyle\sum_{v \in V} w(v) \cdot x_v \\ &\overline{x}_u + \overline{x}_v \geq 1 && (u, v) \notin E \end{aligned}$$

Recap: Some Problems Expressed as PBO (1/2)

Input:

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

Weighted maximum clique

$$\begin{aligned} &\min \; -\textstyle\sum_{v \in V} w(v) \cdot x_v \\ &\overline{x}_u + \overline{x}_v \geq 1 \qquad\qquad (u, v) \notin E \end{aligned}$$

Weighted minimum vertex cover

$$\begin{aligned} &\min \; \textstyle\sum_{v \in V} w(v) \cdot x_v \\ &x_u + x_v \geq 1 \qquad\qquad (u, v) \in E \end{aligned}$$

Recap: Some Problems Expressed as PBO (2/2)

Input:

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

Recap: 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

Recap: 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

$$\min \sum_{e \in \mathcal{U}} w(e) \cdot x_e$$

$$\sum_{e \in S_i} x_e \geq 1 \qquad i \in [m]$$

Recap: 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

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

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

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

Without loss of generality, this is what a MaxSAT problem is (as we will soon see)

Outline of Tutorial on Pseudo-Boolean Optimization

1. Basics of MaxSAT and Pseudo-Boolean Optimization
   - Problem Definition
   - Maximum Satisfiability (MaxSAT) Solving
   - Solution-Improving SAT-UNSAT Search
2. Core-Directed UNSAT-SAT Search
   - Basic Core-Guided Search
   - Advanced Core-Guided Search Techniques
   - Implicit Hitting Set (IHS) and Abstract Cores
3. Some Open Problems

Classic Definition of MaxSAT Problem

Pseudo-Boolean optimization and MaxSAT solving intimately connected, so let's start with traditional description of MaxSAT problem

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 $\sum_{\rho(C_i)=1, i \in [m]} w_i$

Classic Definition of MaxSAT Problem

Pseudo-Boolean optimization and MaxSAT solving intimately connected, so let's start with traditional description of MaxSAT problem

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
  From now on, MaxSAT is a minimization problem
- 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

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

PBO instance

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

Add fresh variable b_i to each soft clause C_i and minimize $\sum_i w_i b_i$
So-called blocking variable transformation
Variables b_i are blocking or relaxation variables

From MaxSAT to Pseudo-Boolean Optimization

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

Add fresh variable b_i to each soft clause C_i and minimize $\sum_i w_i b_i$

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

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

From Pseudo-Boolean Optimization to MaxSAT/WBO

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

PBO instance

$$\begin{aligned} \min \ & \textstyle\sum_{i=1}^{n} w_i \ell_i \\ & C_1 \\ & C_2 \\ & \vdots \\ & C_M \end{aligned}$$

MaxSAT/WBO instance

$$\begin{aligned} & (\overline{\ell}_1)_{w_1} \\ & \vdots \\ & (\overline{\ell}_n)_{w_n} \\ & (C_1)_\infty \\ & \vdots \\ & (C_M)_\infty \end{aligned}$$

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 (and Pseudo-Boolean Optimization)

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

Main Approaches for MaxSAT (and Pseudo-Boolean Optimization)

1. Linear search SAT-UNSAT (LSU) (or solution-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)

Main Approaches for MaxSAT (and Pseudo-Boolean Optimization)

1. Linear search SAT-UNSAT (LSU) (or solution-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 $F = C_1 \land \cdots \land C_m$ (possibly clausal)

List above not exhaustive — omits, e.g., branch-and-bound MaxSAT [LXC+21]

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

1. Run SAT/PB solver
2. If solver returns `UNSATISFIABLE`, output ρ_{best} and terminate
3. Otherwise, let $\rho_{\mathrm{best}} :=$ 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 Encoding of Solution-Improving Constraint

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

- Encodings like dynamic polynomial watchdog [PRB18] state of the art
- More sophisticated than purely linear search

CNF Encoding of Solution-Improving Constraint

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

- Encodings like dynamic polynomial watchdog [PRB18] state of the art
- More sophisticated than purely linear search

For pseudo-Boolean solver, no re-encoding needed — solution-improving constraint can be added as-is

But also no access to auxiliary variables from CNF encodings

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

Quick Detour: Running Solvers with Assumptions

Given

- CNF or pseudo-Boolean formula F
- partial assignment σ

can run SAT or pseudo-Boolean solver on F with assumptions σ

Quick Detour: Running Solvers with Assumptions

Given

- CNF or pseudo-Boolean formula F
- partial assignment σ

can run SAT or pseudo-Boolean solver on F with assumptions σ

Solver works exactly as before, except when making decisions

- Start by assigning variables in σ
- When all of σ taken care of, switch to standard decision heuristic

Quick Detour: Running Solvers with Assumptions

Given

- CNF or pseudo-Boolean formula F
- partial assignment σ

can run SAT or pseudo-Boolean solver on F with assumptions σ

Solver works exactly as before, except when making decisions

- Start by assigning variables in σ
- When all of σ taken care of, switch to standard decision heuristic

Solver outputs

- either solution extending σ
- or explanation (clause/pseudo-Boolean inequality referred to as core constraint) why assumptions σ inconsistent with F

Explanation obtained by simple modification of conflict analysis (decision learning scheme)

Pseudo-Boolean Conflict Analysis Using Decision Learning Scheme

Precondition: Trail ρ has led to conflict $C_{\mathrm{confl}} \in \mathcal{D}$

Pseudo-Boolean Conflict Analysis Using Decision Learning Scheme

Precondition: Trail ρ has led to conflict $C_{\mathrm{confl}} \in \mathcal{D}$

decisionLearningPB$(\mathcal{D}, \rho, C_{\mathrm{confl}})$

$C_{\mathrm{learn}} \leftarrow C_{\mathrm{confl}}$;
$\boldsymbol{\rho}_{\mathbf{dec}} \leftarrow$ all decisions in $\boldsymbol{\rho}$;
while $\boldsymbol{\rho} \neq \emptyset$ **and** $C_{\mathrm{learn}} \neq \bot$ **do**
  $\ell \leftarrow$ literal assigned last on trail ρ ;
  if ℓ *propagated* **and** $\overline{\ell}$ *occurs in* C_{learn} **then**
    $C_{\mathrm{reason}} \leftarrow \mathrm{reason}(\ell, \rho, \mathcal{D})$;
    $C_{\mathrm{reduced}} \leftarrow \mathrm{reduce}(C_{\mathrm{reason}}, C_{\mathrm{learn}}, \ell, \boldsymbol{\rho} \cup \boldsymbol{\rho}_{\mathbf{dec}})$;
    $C_{\mathrm{learn}} \leftarrow \mathrm{resolve}(C_{\mathrm{learn}}, C_{\mathrm{reduced}}, \ell)$;
  $\rho \leftarrow \rho \setminus \{\ell\}$;
return C_{learn} ;

Pseudo-Boolean Conflict Analysis Using Decision Learning Scheme

Precondition: Trail ρ has led to conflict $C_{\mathrm{confl}} \in \mathcal{D}$

decisionLearningPB$(\mathcal{D}, \rho, C_{\mathrm{confl}})$

$C_{\mathrm{learn}} \leftarrow C_{\mathrm{confl}}$;
$\boldsymbol{\rho}_{\mathbf{dec}} \leftarrow$ all decisions in $\boldsymbol{\rho}$;
while $\boldsymbol{\rho} \neq \emptyset$ **and** $C_{\mathrm{learn}} \neq \bot$ **do**
    $\ell \leftarrow$ literal assigned last on trail ρ ;
    if ℓ *propagated* **and** $\overline{\ell}$ *occurs in* C_{learn} **then**
        $C_{\mathrm{reason}} \leftarrow \mathrm{reason}(\ell, \rho, \mathcal{D})$;
        $C_{\mathrm{reduced}} \leftarrow \mathrm{reduce}(C_{\mathrm{reason}}, C_{\mathrm{learn}}, \ell, \boldsymbol{\rho} \cup \boldsymbol{\rho}_{\mathbf{dec}})$;
        $C_{\mathrm{learn}} \leftarrow \mathrm{resolve}(C_{\mathrm{learn}}, C_{\mathrm{reduced}}, \ell)$;
    $\rho \leftarrow \rho \setminus \{\ell\}$;
return C_{learn} ;

Postcondition: Learned constraint C_{learn} is violated by decisions ρ_{dec}

Solving with Assumptions and Decision Learning Scheme

Given assumptions σ with $S = |\rho|$

Assumptions and decision learning scheme for CDCL

- Run standard CDCL but with first S decisions coming from σ
- When conflict reached at level $\leq S$, switch to decision learning scheme and return learned constraint

Solving with Assumptions and Decision Learning Scheme

Given assumptions σ with $S = |\rho|$

Assumptions and decision learning scheme for CDCL

- Run standard CDCL but with first S decisions coming from σ
- When conflict reached at level $\leq S$, switch to decision learning scheme and return learned constraint

Assumptions and decision learning scheme for pseudo-Boolean solving

- Can reach level $\leq S$ during conflict analysis even for conflict at level $S' > S$
- If this happens, switch to decision learning scheme on-the-fly?
- Or only when conflict reached at level $\leq S$? (Will happen next conflict)
- Or keep going as long as conflict level decreases?

Core-Guided Search

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

To see where name comes from, consider 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 $F = C_1 \wedge \cdots \wedge C_m$

To see where name comes from, consider MaxSAT instance with ℓ_i as blocking variables

Core-guided search sets $val_{\text{best}} = 0$ and repeats the following:

1. Run SAT solver with assumptions $\ell_i = 0$ 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 $F = C_1 \wedge \cdots \wedge C_m$

To see where name comes from, consider MaxSAT instance with ℓ_i as blocking variables

Core-guided search sets $val_{\mathrm{best}} = 0$ and repeats the following:

1. Run SAT solver with assumptions $\ell_i = 0$ 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 $F = C_1 \wedge \cdots \wedge C_m$

To see where name comes from, consider MaxSAT instance with ℓ_i as blocking variables

Core-guided search sets $val_{\text{best}} = 0$ and repeats the following:

1. Run SAT solver with assumptions $\ell_i = 0$ 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 $F = C_1 \wedge \cdots \wedge C_m$

To see where name comes from, consider MaxSAT instance with ℓ_i as blocking variables

Core-guided search sets $val_{\mathrm{best}} = 0$ and repeats the following:

1. Run SAT solver with assumptions $\ell_i = 0$ 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 $F = C_1 \wedge \cdots \wedge C_m$

To see where name comes from, consider MaxSAT instance with ℓ_i as blocking variables

Core-guided search sets $val_{\mathrm{best}} = 0$ and repeats the following:

1. Run SAT solver with assumptions $\ell_i = 0$ 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 counter 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 $F = C_1 \wedge \cdots \wedge C_m$

To see where name comes from, consider MaxSAT instance with ℓ_i as blocking variables

Core-guided search sets $val_{\text{best}} = 0$ and repeats the following:

1. Run SAT solver with assumptions $\ell_i = 0$ 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 counter 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 $F = C_1 \wedge \cdots \wedge C_m$

To see where name comes from, consider MaxSAT instance with ℓ_i as blocking variables

Core-guided search sets $val_{\text{best}} = 0$ and repeats the following:

1. Run SAT solver with assumptions $\ell_i = 0$ 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 counter 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 pseudo-Boolean 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 pseudo-Boolean 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 $val_{\text{best}} = 0$ and 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 $val_{\text{best}} = 0$ and run solver on F with assumptions $x_1 = x_2 = \ldots = x_6 = 0$
3. 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 $val_{\text{best}} = 0$ and run solver on F with assumptions $x_1 = x_2 = \ldots = x_6 = 0$
3. Suppose solver returns PB core constraint

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

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

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

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

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

5. Introduce new, fresh counter 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)

6. 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

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

Core-Guided Search Toy Example (2/5)

6. 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

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

7. Update $val_{\text{best}} = 4$ and run solver on F assuming $\ell = 0$ for all literals ℓ in rewritten objective (5)

Core-Guided Search Toy Example (3/5)

9. 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)

9. Suppose solver returns the clausal core constraint

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

10. 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)

10. 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)

10. 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

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

11. Update $val_{\text{best}} = 6$ and run solver on F assuming $\ell = 0$ for all literals ℓ in rewritten objective (8)

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 solution-improving search 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?

Weight Stratification [ABGL12]

Suppose objective function is

$$x_1 + 2x_2 + 3x_3 + 11x_4 + 12x_5 + 13x_6 + 101x_7 + 102x_8 + 103x_9$$

Weight Stratification [ABGL12]

Suppose objective function is

$$x_1 + 2x_2 + 3x_3 + 11x_4 + 12x_5 + 13x_6 + 101x_7 + 102x_8 + 103x_9$$

Focus on variables with largest weight in objective:

1. First assume $x_7 = x_8 = x_9 = 0$ and try to get core
2. If this fails, assume $x_4 = x_5 = x_6 = x_7 = x_8 = x_9 = 0$ and try to get core
3. Only then assume all of objective function is 0

Weight Stratification [ABGL12]

Suppose objective function is

$$x_1 + 2x_2 + 3x_3 + 11x_4 + 12x_5 + 13x_6 + 101x_7 + 102x_8 + 103x_9$$

Focus on variables with largest weight in objective:

1. First assume $x_7 = x_8 = x_9 = 0$ and try to get core
2. If this fails, assume $x_4 = x_5 = x_6 = x_7 = x_8 = x_9 = 0$ and try to get core
3. Only then assume all of objective function is 0

What do we gain from this?

- More compact core yielding larger increase of lower bound, or
- Decent solution found early on

Disjoint Cores and Weight-Aware Core Extraction

Consider our core-guided toy example with objective function

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

and assumptions $x_1 = x_2 = x_3 = \ldots = x_6 = 0$ yielding core $3x_2 + 2x_3 + x_4 + x_5 \geq 4$
rounded to cardinality constraint $x_2 + x_3 + x_4 + x_5 \geq 2$

Disjoint Cores and Weight-Aware Core Extraction

Consider our core-guided toy example with objective function

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

and assumptions $x_1 = x_2 = x_3 = \ldots = x_6 = 0$ yielding core $3x_2 + 2x_3 + x_4 + x_5 \geq 4$ rounded to cardinality constraint $x_2 + x_3 + x_4 + x_5 \geq 2$

Disjoint cores [DB11, DB13, Sai15]

- Remove variables in core found
- Call solver with remaining assumptions $x_1 = x_6 = 0$

Disjoint Cores and Weight-Aware Core Extraction

Consider our core-guided toy example with objective function

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

and assumptions $x_1 = x_2 = x_3 = \ldots = x_6 = 0$ yielding core $3x_2 + 2x_3 + x_4 + x_5 \geq 4$ rounded to cardinality constraint $x_2 + x_3 + x_4 + x_5 \geq 2$

Disjoint cores [DB11, DB13, Sai15]

- Remove variables in core found
- Call solver with remaining assumptions $x_1 = x_6 = 0$

Weight-aware core extraction [BJ17]

- Remove only variables that cancel in objective rewriting, in our case x_2
- Call solver with assumptions $x_1 = x_3 = x_4 = x_5 = x_6 = 0$

Disjoint Cores and Weight-Aware Core Extraction

Consider our core-guided toy example with objective function

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

and assumptions $x_1 = x_2 = x_3 = \ldots = x_6 = 0$ yielding core $3x_2 + 2x_3 + x_4 + x_5 \geq 4$ rounded to cardinality constraint $x_2 + x_3 + x_4 + x_5 \geq 2$

Disjoint cores [DB11, DB13, Sai15]

- Remove variables in core found
- Call solver with remaining assumptions $x_1 = x_6 = 0$

Weight-aware core extraction [BJ17]

- Remove only variables that cancel in objective rewriting, in our case x_2
- Call solver with assumptions $x_1 = x_3 = x_4 = x_5 = x_6 = 0$

Find more independent cores that contribute to larger increase of lower bound

Improving the Core Constraints

Suppose solver returns a core $\sum_i a_i \ell_i \geq A$ (where in MaxSAT $a_i = A = 1$)

Core exhaustion [IMM19]

- Define counter variable
$$y_{A+1} \Leftrightarrow \sum_i a_i \ell_i \geq A + 1$$
- Run solver with assumption $y_{A+1} = 0$
- If solver returns `UNSATISFIABLE`, can strengthen core

Improving the Core Constraints

Suppose solver returns a core $\sum_i a_i \ell_i \geq A$ (where in MaxSAT $a_i = A = 1$)

Core exhaustion [IMM19]

- Define counter variable
$$y_{A+1} \Leftrightarrow \sum_i a_i \ell_i \geq A + 1$$
- Run solver with assumption $y_{A+1} = 0$
- If solver returns UNSATISFIABLE, can strengthen core

Core trimming/minimization [Mar10, MIM15]

- Choose different literals ℓ_{j^*}
- Check if $\sum_{i \neq j^*} a_i \ell_i \geq A$ still core
- Straightforward for clauses — less obvious how this would work for PB constraints

Lazy Counter Variables for Cores [MJML14, DGD+21]

Consider again core-guided toy example with first core $3x_2 + 2x_3 + x_4 + x_5 \geq 4$ rounded to cardinality constraint $x_2 + x_3 + x_4 + x_5 \geq 2$

Lazy Counter Variables for Cores [MJML14, DGD$^+$21]

Consider again core-guided toy example with first core $3x_2 + 2x_3 + x_4 + x_5 \geq 4$ rounded to cardinality constraint $x_2 + x_3 + x_4 + x_5 \geq 2$

Eager counter variables

- Add all counter variables y_3 and y_4 and constraints

$$x_2 + x_3 + x_4 + x_5 = 2 + y_3 + y_4$$
$$y_3 \geq y_4$$

Lazy Counter Variables for Cores [MJML14, DGD+21]

Consider again core-guided toy example with first core $3x_2 + 2x_3 + x_4 + x_5 \geq 4$ rounded to cardinality constraint $x_2 + x_3 + x_4 + x_5 \geq 2$

Eager counter variables

- Add all counter variables y_3 and y_4 and constraints

$$x_2 + x_3 + x_4 + x_5 = 2 + y_3 + y_4$$
$$y_3 \geq y_4$$

Lazy counter variables

- Add y_3 defined by

$$y_3 \Leftrightarrow x_2 + x_3 + x_4 + x_5 \geq 3$$

- Only introduce $y_4 \Leftrightarrow x_2 + x_3 + x_4 + x_5 \geq 4$ when y_3 cancels in rewritten objective
- Rewriting of objective can introduce huge numbers of variables, slowing down solver
- Adding variables lazily only when needed speeds things up

Combining Core-Guided and Solution-Improving Search

Core boosting [BDS19]

- Start with core-guided search to get good lower bound estimate
- Then switch to solution-improving search to find optimal solution

Combining Core-Guided and Solution-Improving Search

Core boosting [BDS19]

- Start with core-guided search to get good lower bound estimate
- Then switch to solution-improving search to find optimal solution

Hybrid/interleaving search [ADMR15]

- Switch back and forth repeatedly between core-guided and solution-improving search
- Cumbersome in CNF-based solver
- But fairly cheap (and efficient) in native pseudo-Boolean solver [DGD+21]

Some More Parameters to Play with (Not Exhaustive List)

1. Adjust phase saving
2. Detect intrinsic at-most-1 constraints [IMM19]
3. Rewrite the objective with non-cardinality core constraints [JBJ24]
4. Avoid storing solution-improving constraint in solver database [SBJ21, SBJ22]

Theoretical Analysis of Core-Guided Search?

Lower bound computed by core-guided search

- Core-guided search provides lower bound estimate based on cores found
- Can be viewed as optimum of LP relaxation of problem to minimize objective given cores and counter variable definitions [Kat23]

Inference strength of core-guided search?

- Extension variables very strong in theory — no proof complexity lower bounds
- So far hard to leverage this power in practice
- But core-guided search provides principled way of introducing extension variables
- And is very structured — possible to analyze power of this method with proof complexity?

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

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

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

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

RoundingSat with core-guided (CG) and linear SAT-UNSAT 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 Pseudo-Boolean Solver in [DGD+21]

RoundingSat with core-guided (CG) and linear SAT-UNSAT 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**

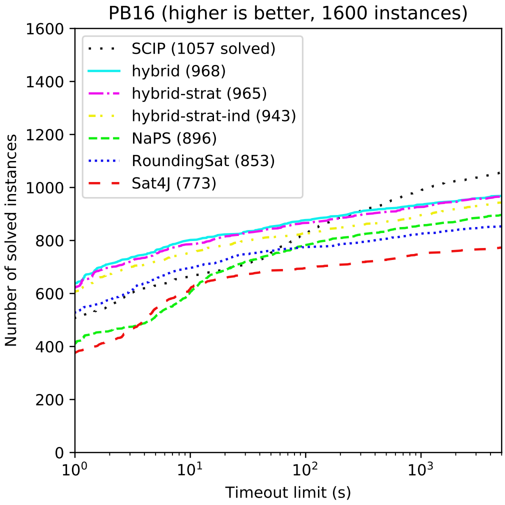
Significant improvement over PB state of the art, but MIP solver SCIP still better

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

Cumulative plot for solver performance on PB16 optimization benchmarks

Also including

- stratfication (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 $F = C_1 \land \cdots \land C_m$ (suppose for now clausal constraints)

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

$$
\begin{aligned}
C_1 &\doteq \ell_{1,1} \lor \ell_{1,2} \lor \cdots \lor \ell_{1,k_1} \\
C_2 &\doteq \ell_{2,1} \lor \ell_{2,2} \lor \cdots \lor \ell_{2,k_2} \\
&\vdots \\
C_s &\doteq \ell_{s,1} \lor \ell_{s,2} \lor \cdots \lor \ell_{s,k_s}
\end{aligned}
$$

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

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

1. Run optimization solver to 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 decision solver on F with assumptions $\{\ell_j = 0 \mid \ell_j \notin H\}$
3. If decision 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, decision solver returns new core C_{s+1} — add it to $\mathcal{K}$ and start over

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 (good enough to improve currently best solution)
- How to find hitting set?
- This is itself a pseudo-Boolean optimization problem
    - Run integer linear programming (ILP) solver [standard approach]
    - Or pseudo-Boolean solver?
    - Or local search?!

Combine IHS with Pseudo-Boolean Optimization?

IHS and PB Optimization

- In pseudo-Boolean 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 pseudo-Boolean 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 decision 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, decision 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 run 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 run 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
   $\rho_2 = \{x_2 = x_3 = 1; x_1 = x_4 = x_5 = x_6 = 0\}$

IHS Toy Example (2/2)

5. Decision solver run with assumptions $\rho'_2 = \{x_1 = x_4 = x_5 = x_6 = 0\}$ returns PB core constraint

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

IHS Toy Example (2/2)

5. Decision solver run with assumptions $\rho'_2 = \{x_1 = x_4 = x_5 = x_6 = 0\}$ 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 run with assumptions $\rho'_2 = \{x_1 = x_4 = x_5 = x_6 = 0\}$ 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 run with assumptions $\rho'_3 = \{x_1 = x_3 = x_5 = x_6 = 0\}$ returns `SATISFIABLE`

IHS Toy Example (2/2)

5. Decision solver run with assumptions $\rho'_2 = \{x_1 = x_4 = x_5 = x_6 = 0\}$ 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 run with assumptions $\rho'_3 = \{x_1 = x_3 = x_5 = x_6 = 0\}$ returns `SATISFIABLE`

8. Hence, we have found an optimal solution with objective value 6 (as for solution-improving search and core-guided search)

Comparison of Core-Guided Search and Implicit Hitting Set

Suppose solver with assumptions returns core

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

Comparison of Core-Guided Search and Implicit Hitting Set

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 next call (cancelled when objective rewritten)
- Instead assume that "somehow $x_1 + x_2 + x_3 + x_4 \leq 2$ holds" (i.e., assume $y_3 = 0$)
- Lower bound estimate in rewritten objective from LP relaxation of cores [Kat23]

Comparison of Core-Guided Search and Implicit Hitting Set

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 next call (cancelled when objective rewritten)
- Instead assume that "somehow $x_1 + x_2 + x_3 + x_4 \leq 2$ holds" (i.e., assume $y_3 = 0$)
- Lower bound estimate in rewritten objective from LP relaxation of cores [Kat23]

Implicit Hitting Set

- Add C to collection of cores $\mathcal{K}$
- Find concrete assignment satisfying all of $\mathcal{K}$ as cheaply as possible
- Get lower bound estimate from actual 0–1 solution for cores, not from LP relaxation
- Try that candidate solution as starting point for next call to decision solver

Competitive Advantages of Core-Guided vs. Implicit Hitting Set

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

Competitive Advantages of Core-Guided vs. Implicit Hitting Set

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

Abstract Cores for MaxSAT [BBP20]

Combination of implicit hitting set and core-guided search:

- Run solver with assumption to collect cores $C \doteq \ell_1 + \ell_2 + \cdots + \ell_k \geq 1$

Abstract Cores for MaxSAT [BBP20]

Combination of implicit hitting set and core-guided search:

- Run solver with assumption to collect cores $C \doteq \ell_1 + \ell_2 + \cdots + \ell_k \geq 1$
- Identify "interesting literal set" L
  - maybe literals in C
  - or literals appearing in many different cores

Abstract Cores for MaxSAT [BBP20]

Combination of implicit hitting set and core-guided search:

- Run solver with assumption to collect cores $C \doteq \ell_1 + \ell_2 + \cdots + \ell_k \geq 1$
- Identify "interesting literal set" L
  - maybe literals in C
  - or literals appearing in many different cores
- Introduce same counter variables as for core-guided OLL rewriting

$$y_j \Leftrightarrow \sum_{\ell \in L} \ell \geq j$$

Abstract Cores for MaxSAT [BBP20]

Combination of implicit hitting set and core-guided search:

- Run solver with assumption to collect cores $C \doteq \ell_1 + \ell_2 + \cdots + \ell_k \geq 1$
- Identify "interesting literal set" L
  - maybe literals in C
  - or literals appearing in many different cores
- Introduce same counter variables as for core-guided OLL rewriting

$$y_j \Leftrightarrow \sum_{\ell \in L} \ell \geq j$$

- Solve weighted hitting set problem over cores + definitions of counter variables

Abstract Cores for MaxSAT [BBP20]

Combination of implicit hitting set and core-guided search:

- Run solver with assumption to collect cores $C \doteq \ell_1 + \ell_2 + \cdots + \ell_k \geq 1$
- Identify "interesting literal set" L
  - maybe literals in C
  - or literals appearing in many different cores
- Introduce same counter variables as for core-guided OLL rewriting

$$y_j \Leftrightarrow \sum_{\ell \in L} \ell \geq j$$

- Solve weighted hitting set problem over cores $+$ definitions of counter variables
- Assumptions for next round:
  - use solution to hitting set (as in standard IHS)
  - plus set non-propagated counter variables to 0

Abstract Cores for MaxSAT [BBP20]

Combination of implicit hitting set and core-guided search:

- Run solver with assumption to collect cores $C \doteq \ell_1 + \ell_2 + \cdots + \ell_k \geq 1$
- Identify "interesting literal set" L
  - maybe literals in C
  - or literals appearing in many different cores
- Introduce same counter variables as for core-guided OLL rewriting

$$y_j \Leftrightarrow \sum_{\ell \in L} \ell \geq j$$

- Solve weighted hitting set problem over cores + definitions of counter variables
- Assumptions for next round:
  - use solution to hitting set (as in standard IHS)
  - plus set non-propagated counter variables to 0
- How could/should this be generalized to pseudo-Boolean setting?

Challenges and Opportunities for Pseudo-Boolean Optimization (PBO)

1. **Make pseudo-Boolean optimization better known**
    - Personal observation: Most MaxSAT applications seem to be naturally described as pseudo-Boolean optimization problem + PB-to-CNF translation
    - Why not use pseudo-Boolean optimization directly?!

Challenges and Opportunities for Pseudo-Boolean Optimization (PBO)

1. **Make pseudo-Boolean optimization better known**
   - Personal observation: Most MaxSAT applications seem to be naturally described as pseudo-Boolean optimization problem + PB-to-CNF translation
   - Why not use pseudo-Boolean optimization directly?!
2. **Make PBO competitive on CNF formulas**
   - Since many problems already encoded in CNF, improve PB solver performance on CNF
   - Could on-the-fly cardinality detection [EN20] help?

Challenges and Opportunities for Pseudo-Boolean Optimization (PBO)

1. **Make pseudo-Boolean optimization better known**
   - Personal observation: Most MaxSAT applications seem to be naturally described as pseudo-Boolean optimization problem + PB-to-CNF translation
   - Why not use pseudo-Boolean optimization directly?!

2. **Make PBO competitive on CNF formulas**
   - Since many problems already encoded in CNF, improve PB solver performance on CNF
   - Could on-the-fly cardinality detection [EN20] help?

3. **Integrate IHS, core-guided, solution-improving search, . . .**
   - Present algorithms in unified framework
   - Make it possible to switch dynamically and easily between different approaches
   - Attempt in [IBJ24], but fixed concrete approach after algorithm instantiation

Challenges and Opportunities for Pseudo-Boolean Optimization (PBO)

1. **Make pseudo-Boolean optimization better known**
   - Personal observation: Most MaxSAT applications seem to be naturally described as pseudo-Boolean optimization problem + PB-to-CNF translation
   - Why not use pseudo-Boolean optimization directly?!
2. **Make PBO competitive on CNF formulas**
   - Since many problems already encoded in CNF, improve PB solver performance on CNF
   - Could on-the-fly cardinality detection [EN20] help?
3. **Integrate IHS, core-guided, solution-improving search, . . .**
   - Present algorithms in unified framework
   - Make it possible to switch dynamically and easily between different approaches
   - Attempt in [IBJ24], but fixed concrete approach after algorithm instantiation
4. **Leverage ideas from mixed integer programming (MIP)**
   - *Next and final lecture. . .*

Summing up

- MaxSAT problems can be attacked with combination of powerful tools
  - Core-guided solving
  - Implicit hitting set (IHS) solving
  - Integer linear programming
- Approaches with complementary strengths — room for exploiting synergies?
- Lifting core-guided and IHS algorithms to 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 — more choices needed to get right
- Many interesting questions to explore — and rich pickings of low-hanging fruit?

Summing up

- MaxSAT problems can be attacked with combination of powerful tools
  - Core-guided solving
  - Implicit hitting set (IHS) solving
  - Integer linear programming
- Approaches with complementary strengths — room for exploiting synergies?
- Lifting core-guided and IHS algorithms to 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 — more choices needed to get right
- Many interesting questions to explore — and 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.

[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.

References II

[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.

[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.

References III

[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*.

[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.

[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.

References IV

[IBJ24] Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Unifying SAT-based approaches to maximum satisfiability solving. *Journal of Artificial Intelligence Research*, 80:931–976, July 2024. Preliminary version in *IJCAI '23*.

[IMM19] Alexey Ignatiev, António Morgado, and João P. Marques-Silva. RC2: an efficient MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):53–64, September 2019.

[JBJ24] Christoph Jabs, Jeremias Berg, and Matti Järvisalo. PB-OLL-RS and Mixed-Bag in Pseudo-Boolean Competition 2024. Available at `https://www.cril.univ-artois.fr/PB24/descr/mixed-bag-pb-oll-rs.pdf`, August 2024.

[Kat23] George Katsirelos. An analysis of core-guided maximum satisfiability solvers using linear programming. In *Proceedings of the 26th International Conference on Theory and Applications of Satisfiability Testing (SAT '23)*, volume 271 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:19, July 2023.

[LXC+21] Chu-Min Li, Zhenxing Xu, Jordi Coll, Felip Manyà, Djamal Habet, and Kun He. Combining clause learning and branch and bound for MaxSAT. 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 38:1–38:18, October 2021.

References V

[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.

[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.

References VI

[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.

[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`.

References VII

[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.

[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.