Oracle-Based Local Search for PBO

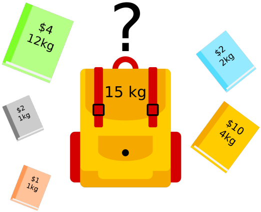
originally published at ECAI 2023

Jeremias Berg[2]
joint work with Markus Iser[1], and Matti Järvisalo[2]

1: Karlsruhe Institute of Technology,
2: University of Helsinki

November 2024

Motivation - Anytime Solving



- Constraint optimization, useful and important
- Guaranteed optimality nice but not always needed / achievable.
- *Anytime solving* $\rightarrow$ good solution within a limited time and memory.
- Our work: Harness recent advances in MaxSAT and conflict driven pseudo-Boolean solving to anytime settings.

Notation

Pseudo-Boolean Optimization (PBO)

- (F, O) where:
  - Formula F, a set of PB constraints $\sum_{i=1}^{n} c_i \ell_i \geq B$.
  - Literal ℓ, a $0-1$ variable x or its negation $\overline{x} = 1 - x$
  - Objective O a PB expression $\sum_{i=1}^{m} c_i b_i$ to be minimized.
- Goal: Compute assignment (solution) α that satisfies F and minimizes $\alpha(O) = \sum_{i=1}^{m} c_i \cdot \alpha(b_i)$

Note: MaxSAT is a special case

Notation

Pseudo-Boolean Optimization (PBO)

- (F, O) where:
  - Formula F, a set of PB constraints $\sum_{i=1}^{n} c_i \ell_i \geq B$.
  - Literal ℓ, a $0-1$ variable x or its negation $\overline{x} = 1 - x$
  - Objective O a PB expression $\sum_{i=1}^{m} c_i b_i$ to be minimized.
- Goal: Compute assignment (solution) α that satisfies F and minimizes $\alpha(O) = \sum_{i=1}^{m} c_i \cdot \alpha(b_i)$

Satisfiability under Assumptions

- *Assumptions* $\mathcal{A}$, a set of literals.
- `Solve-Assumps`$(F, \mathcal{A})$ returns α s.t. $\alpha(\ell) = 1$ for all $\ell \in \mathcal{A}$, or unsat.

Notation

Pseudo-Boolean Optimization (PBO)

- (F, O) where:
  - Formula F, a set of PB constraints $\sum_{i=1}^{n} c_i \ell_i \geq B$.
  - Literal ℓ, a $0-1$ variable x or its negation $\overline{x} = 1 - x$
  - Objective O a PB expression $\sum_{i=1}^{m} c_i b_i$ to be minimized.
- Goal: Compute assignment (solution) α that satisfies F and minimizes $\alpha(O) = \sum_{i=1}^{m} c_i \cdot \alpha(b_i)$

Satisfiability under Assumptions

- *Assumptions* $\mathcal{A}$, a set of literals.
- `Solve-Assumps`$(F, \mathcal{A}, \mathit{clim})$ returns α s.t. $\alpha(\ell) = 1$ for all $\ell \in \mathcal{A}$, unsat, or "unknown" after *clim* conflicts.
- We add a conflict limit to escape difficult calls.

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,$$
$$b_1 + b_4 \geq 1,$$
$$b_2 + b_5 \geq 1\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

- `Solve-Assumps`$(F, \{\overline{b_1}, \overline{b_5}\})$ returns $\alpha = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$, cost 10.
- `Solve-Assumps`$(F, \{\overline{b_1}, \overline{b_4}\})$ returns unsat.
- An optimal solution is $\{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$ has cost 9.

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,$$
$$b_1 + b_4 \geq 1,$$
$$b_2 + b_5 \geq 1\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

- Solve-Assumps$(F, \{\overline{b_1}, \overline{b_5}\})$ returns $\alpha = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$, cost 10.
- Solve-Assumps$(F, \{\overline{b_1}, \overline{b_4}\})$ returns unsat.
- An optimal solution is $\{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$ has cost 9.

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,$$
$$b_1 + b_4 \geq 1,$$
$$b_2 + b_5 \geq 1\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

- Solve-Assumps$(F, \{\overline{b_1}, \overline{b_5}\})$ returns $\alpha = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$, cost 10.
- Solve-Assumps$(F, \{\overline{b_1}, \overline{b_4}\})$ returns unsat.
- An optimal solution is $\{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$ has cost 9.

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3, \\ b_1 + b_4 \geq 1, \\ b_2 + b_5 \geq 1\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

- Solve-Assumps($F, \{\overline{b_1}, \overline{b_5}\}$) returns $\alpha = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$, cost 10.
- Solve-Assumps($F, \{\overline{b_1}, \overline{b_4}\}$) returns unsat.
- An optimal solution is $\{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$ has cost 9.

Existing approaches to PBO

- Complete:
  - CDCL-based
  - core-guided
  - implicit-hitting set
  - solution-improving
- Any-time:
  - Solution-Improving
  - Stochastic Local Search
  - Oracle-based Local Search
- Early work: convert to CNF
- More recent: direct reasoning with cutting planes

[Devriendt, Gocht, Demirovic, Nordström, and Stuckey, 2021b]

[Elffers and Nordström, 2018]

[Devriendt, Gleixner, and Nordström, 2021a]

[Smirnov, Berg, and Järvisalo, 2022]

[Smirnov, Berg, and Järvisalo, 2021]

[Berre and Parrain, 2010]

[Eén and Sörensson, 2006]

[Sheini and Sakallah, 2006]

[Gebser, Kaufmann, Neumann, and Schaub, 2007]

Our Approach

Combination of solution improving search and oracle-based local search.

Solution Improving Search

SIS(F, O)

Generate **initial solution** α_{best}
Update $F \leftarrow F \cup \{O < O(\alpha_{best})\}$
Repeat the following:

- Query: $\texttt{Solve-Assumps}(F, \emptyset)$
  - If SAT: Update α_{best}
    $F \leftarrow F \cup \{O < O(\alpha_{best})\}$
  - Else: α_{best} is optimal.

Can return the best α_{best} at any time

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = \infty$ $\alpha_{best} = \emptyset$

SIS

1. Initialize $\text{ub} = \infty$ and $\alpha_{best} = \emptyset$

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < \infty\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = \infty$ $\alpha_{best} = \emptyset$

SIS

1. Initialize $\text{ub} = \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O < \text{ub}$ to F
   - Invoke Solve-Assumps$(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < \infty\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= \infty$ $\qquad \alpha_{best} = \emptyset$

$\texttt{Solve-Assumps}(F, \emptyset) = \{b_1, b_2, b_3, b_4, \overline{b_5}\}$

SIS

1. Initialize ub $= \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O <$ ub to F
   - Invoke $\texttt{Solve-Assumps}(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < \infty\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = 13$ $\alpha_{best} = \{b_1, b_2, b_3, b_4, \overline{b_5}\}$

$\texttt{Solve-Assumps}(F, \emptyset) = \{b_1, b_2, b_3, b_4, \overline{b_5}\}$

SIS

1. Initialize $\text{ub} = \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O < \text{ub}$ to F
   - Invoke $\texttt{Solve-Assumps}(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < 13\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = 13$ $\alpha_{best} = \{b_1, b_2, b_3, b_4, \overline{b_5}\}$

SIS

1. Initialize $\text{ub} = \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O < \text{ub}$ to F
   - Invoke `Solve-Assumps`$(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < 13\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = 13$ $\qquad$ $\alpha_{best} = \{b_1, b_2, b_3, b_4, \overline{b_5}\}$

$\texttt{Solve-Assumps}(F, \emptyset) = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

SIS

1. Initialize $\text{ub} = \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O < \text{ub}$ to F
   - Invoke $\texttt{Solve-Assumps}(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < 13\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = 9$ $\alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

$\texttt{Solve-Assumps}(F, \emptyset) = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

SIS

1. Initialize $\text{ub} = \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O < \text{ub}$ to F
   - Invoke $\texttt{Solve-Assumps}(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < 9\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 9$ $\qquad \alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

SIS

1. Initialize ub $= \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O <$ ub to F
   - Invoke `Solve-Assumps`$(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < 9\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = 9$ $\alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

$\texttt{Solve-Assumps}(F, \emptyset) = \text{unsat}$

SIS

1. Initialize $\text{ub} = \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O < \text{ub}$ to F
   - Invoke $\texttt{Solve-Assumps}(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Example

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1$$
$$3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5 < 9\}$$

$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\mathrm{ub} = 9$ $\alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

SIS

1. Initialize $\mathrm{ub} = \infty$ and $\alpha_{best} = \emptyset$
2. **Repeat:**
   - Add $O < \mathrm{ub}$ to F
   - Invoke `Solve-Assumps`$(F, \emptyset)$
   - If new solution update ub and α_{best}
   - Else return α_{best}

Oracle-Based Local Search

[Nadel, 2019, 2020]

Traditional SLS

objective vars

$b_1, \overline{b_2}, b_3, \overline{b_4} \mid \ell_1, \ell_2, \ell_3, \ell_4$

$\downarrow$ flip ℓ_3

$b_1, \overline{b_2}, b_3, \overline{b_4} \mid \ell_1, \ell_2, \overline{\ell_3}, \ell_4$

1. Pick a variable
2. Check feasibility (directly)
3. Update α_{best}.

Oracle-Based Local Search

[Nadel, 2019, 2020]

Traditional SLS

objective vars

$b_1, \overline{b_2}, b_3, \overline{b_4} \mid \ell_1, \ell_2, \ell_3, \ell_4$

$\downarrow$ flip ℓ_3

$b_1, \overline{b_2}, b_3, \overline{b_4} \mid \ell_1, \ell_2, \overline{\ell_3}, \ell_4$

1. Pick a variable
2. Check feasibility (directly)
3. Update α_{best}.

Oracle-based SLS

$b_3, \overline{b_2}, b_1, \overline{b_4}$

$\downarrow$ flip b_3

$\overline{b_3}, \overline{b_2}, b_1, \overline{b_4}$

1. Shuffle objective
2. Pick first variable that incurs cost
3. Check feasibility `Solve-Assumps` $(F, \{\overline{b_2}, \overline{b_3}, \overline{b_4}\})$
4. Update α_{best}.

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

Oracle-based local search

1. Initialize ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 17$ $\qquad \alpha_{best} = \{b_1, b_2, b_3, \overline{b_4}, b_5\}$

Oracle-based local search

1. Initialize ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 17$ $\alpha_{best} = \{b_1, b_2, b_3, \overline{b_4}, b_5\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = 17$ $\alpha_{best} = \{b_1, b_2, b_3, \overline{b_4}, b_5\}$

$\text{shuffled-objective-vars} = \{b_1, b_4, b_2, b_3, b_5\}$

$\texttt{Solve-Assumps}(F, \{\overline{b_1}\})$

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 17$ $\qquad \alpha_{best} = \{b_1, b_2, b_3, \overline{b_4}, b_5\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

`Solve-Assumps`$(F, \{\overline{b_1}\})$

Result (e.g.) $\{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$, cost 10

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 10$ $\alpha_{best} = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

`Solve-Assumps`$(F, \{\overline{b_1}\})$

Result (e.g.) $\{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$, cost 10

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
    - Shuffle objective vars
    - Try to locally improve the solution one variable at the time
    - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 10$ $\quad\alpha_{best} = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

`Solve-Assumps`$(F, \{\overline{b_1}, \overline{b_4}\})$

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 10$ $\alpha_{best} = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

`Solve-Assumps`$(F, \{\overline{b_1}, \overline{b_4}\})$ Result unsat

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 10$ $\alpha_{best} = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

$\texttt{Solve-Assumps}(F, \{\overline{b_1}, b_4, \overline{b_2}\})$

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 10$ $\qquad \alpha_{best} = \{\overline{b_1}, b_2, b_3, b_4, \overline{b_5}\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

Solve-Assumps$(F, \{\overline{b_1}, b_4, \overline{b_2}\})$

Result (e.g.) $\{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$, cost 9

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 9$ $\alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

`Solve-Assumps`$(F, \{\overline{b_1}, b_4, \overline{b_2}\})$

Result (e.g.) $\{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$, cost 9

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\; b_1 + b_4 \geq 1,\; b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\mathrm{ub} = 9$ $\qquad \alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

$\texttt{Solve-Assumps}(F, \{\overline{b_1}, b_4, \overline{b_2}, \overline{b_3}\})$

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = 9$ $\alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

$\texttt{Solve-Assumps}(F, \{\overline{b_1}, b_4, \overline{b_2}, \overline{b_3}\})$ Result unsat

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

$\text{ub} = 9$ $\qquad \alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

$\texttt{Solve-Assumps}(F, \{\overline{b_1}, b_4, \overline{b_2}, b_3, \overline{b_5}\})$

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 9$ $\alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

$\texttt{Solve-Assumps}(F, \{\overline{b_1}, b_4, \overline{b_2}, b_3, \overline{b_5}\})$ Result unsat

Oracle-based local search

1. Initialize ub and α_{best}
2. **Repeat until timeout**
   - Shuffle objective vars
   - Try to locally improve the solution one variable at the time
   - If better solution update ub and α_{best}

In more detail

$$F = \{b_1 + b_2 + b_3 + b_4 + b_5 \geq 3,\ b_1 + b_4 \geq 1,\ b_2 + b_5 \geq 1\}$$
$$O = 3b_1 + 6b_2 + 3b_3 + b_4 + 5b_5$$

ub $= 9$ $\alpha_{best} = \{\overline{b_1}, \overline{b_2}, b_3, b_4, b_5\}$

shuffled-objective-vars $= \{b_1, b_4, b_2, b_3, b_5\}$

MS-Beaver

`Solve-Assumps`$(F, \{\overline{b_3}, b_4, \overline{b_2}\})$

PoloSAT

`Solve-Assumps`$(F, \{\overline{b_2}\})$

Variants

- Ms. Beaver $\rightarrow$ use the "full" prefix as assumptions.
- PoloSAT $\rightarrow$ use only the current variable

`OraSLS`(F, Objective O)

Generate initial solution α_{best}
Update $F \leftarrow F \cup \{O < O(\alpha_{best})\}$
Repeat the following:

- Select order of objective variables
- Initialize assumptions $\mathcal{A} \leftarrow \emptyset$
- For each $b \in$ **reordered** objective variables:
  - Solve F under $\mathcal{A} \cup \{\bar{b}\}$ with conflict-limit.
  - If SAT: Fix $\mathcal{A} \leftarrow \mathcal{A} \cup \{\bar{b}\}$
    If solution is better: Update α_{best} and $F \leftarrow F \cup \{O < O(\alpha_{best})\}$
  - Else: Fix $\mathcal{A} \leftarrow \mathcal{A} \cup \{b\}$
    If fail-limit is reached: Break
- If α_{best} was not improved for stagnation-limit number of times:
  - Solve F without conflict-limit or assumptions
  - If SAT: Update α_{best} and $F \leftarrow F \cup \{O < O(\alpha_{best})\}$
  - Else: Return α_{best}

Central Design Choice: How to Order the Objectives

Our approach: "Bucket Shuffle"

- Start with decreasing order of cost
- Partition into n buckets of equal size
- Shuffle each bucket separately

Central Design Choice: How to Order the Objectives

Our approach: "Bucket Shuffle"

- Start with decreasing order of cost
- Partition into n buckets of equal size
- Shuffle each bucket separately
- On every odd iteration reverse the list $\rightarrow$ diversification

Central Design Choice: How to Order the Objectives

Our approach: "Bucket Shuffle"

- Start with decreasing order of cost
- Partition into n buckets of equal size
- Shuffle each bucket separately
- On every odd iteration reverse the list $\rightarrow$ diversification

Example with $n = 2$

$$O = 6b_2 + 5b_5 + 3b_3 + 3b_1 + b_4$$

Central Design Choice: How to Order the Objectives

Our approach: "Bucket Shuffle"

- Start with decreasing order of cost
- Partition into n buckets of equal size
- Shuffle each bucket separately
- On every odd iteration reverse the list $\rightarrow$ diversification

Example with $n = 2$

$$O = 6b_2 + 5b_5 + 3b_3 + 3b_1 + b_4$$

$\{b_2, b_5, b_3\}$ $\{b_1, b_4\}$ *partition*

Central Design Choice: How to Order the Objectives

Our approach: "Bucket Shuffle"

- Start with decreasing order of cost
- Partition into n buckets of equal size
- Shuffle each bucket separately
- On every odd iteration reverse the list $\rightarrow$ diversification

Example with $n = 2$

$$O = 6b_2 + 5b_5 + 3b_3 + 3b_1 + b_4$$

$\{b_2, b_5, b_3\}$	$\{b_1, b_4\}$	*partition*
$\{b_3, b_2, b_5\}$	$\{b_1, b_4\}$	*shuffle*

Central Design Choice: How to Order the Objectives

Our approach: "Bucket Shuffle"

- Start with decreasing order of cost
- Partition into *n* buckets of equal size
- Shuffle each bucket separately
- On every odd iteration reverse the list → diversification

Example with $n = 2$

$$O = 6b_2 + 5b_5 + 3b_3 + 3b_1 + b_4$$

$\{b_2, b_5, b_3\}$	$\{b_1, b_4\}$	*partition*
$\{b_3, b_2, b_5\}$	$\{b_1, b_4\}$	*shuffle*

$$\text{shuffled-objective-vars} = \{b_3, b_2, b_5, b_1, b_4\}$$

Heuristics

Initial Solution Computation

- Bump activities of objective variables proportionately to their coefficients.
- Try the assignment that does not incur cost first

Heuristics

Initial Solution Computation

- Bump activities of objective variables proportionately to their coefficients.
- Try the assignment that does not incur cost first

Oracle calls in "internal loop"

- Set polarity selection to match the best known solution.
- Also used in MaxSAT [Demirovic and Stuckey, 2019; Berg, Demirovic, and Stuckey, 2019; Nadel, 2020]

Experimental Setup

- Instantiate OraSLS on top of the decision procedure in RoundingSAT (commit: a7fe32d8)
- Compare solution quality of OraSLS with:
  - LS-PBO (Stochastic Local Search)
  - RoundingSAT (CDCL, core-guided, cutting planes)
    - designed for complete setting
  - SIS: solution-improving search in OraSLS codebase
  - SIS+: SIS with polarity selection and activity bumping.
- Metric: Average MSE score

$$\text{score}(\text{solver}, \text{instance}) = \frac{\text{best-cost}(\mathcal{I}) + 1}{\text{solver-cost}(\mathcal{I}) + 1} \in [0, 1]$$

$0 \rightarrow$no sol, $1 \rightarrow$best-known sol.

Parameters & Overall Results

Parameters

- Number of buckets $= 8$
- stagnation-limit $= 1$
- conflict-limit $= 20$
- fail-limit $= 10$

Parameters & Overall Results

Parameters

- Number of buckets $= 8$
- stagnation-limit $= 1$
- conflict-limit $= 20$
- fail-limit $= 10$

Results after 5 minutes (Average scores, TO $\rightarrow$ not solutions found)

Solver	Score	#TO
OraSLS	0.8423	89
RoundingSAT	0.8300	83
SIS+	0.8082	111
SIS	0.7363	114
LS-PBO	0.7280	194

OraSLS vs. RoundingSAT

Domains with more than 0.1 score difference

		RoundingSAT		OraSLS	
Benchmark domain	**#**	**Score**	**#TO**	**Score**	**#TO**
MANETS	20	0.52	0	**0.87**	2
course-alloc	6	0.55	0	**0.92**	0
decomp	10	0.82	1	**0.99**	0
lecture-timetabling	20	0.69	0	**0.93**	0
unibo-various	20	0.61	4	**0.77**	4
wnq	16	**0.65**	0	0.42	0
All domains	865	0.83	83	0.84	89
Wins (#domains)			1		**5**

OraSLS vs. LS-PBO

Domains with more than 0.1 score difference

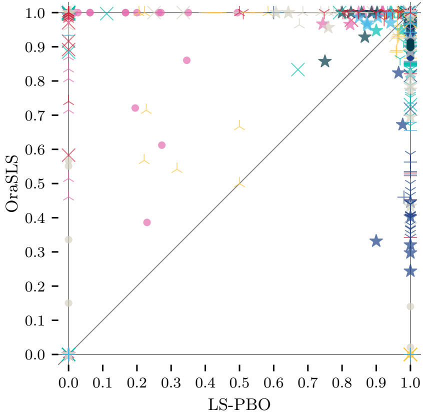
		LS-PBO		OraSLS	
Benchmark domain	**#**	**Score**	**#TO**	**Score**	**#TO**
MANETS	20	0.19	16	**0.87**	2
airplane-cost-quality	20	0.80	4	**1.00**	0
decomp	10	0.88	0	**0.99**	0
domset	15	**1.00**	0	0.85	0
golomb-rulers	20	0.52	8	**0.84**	3
haplotype-inf	20	**0.99**	0	0.77	0
lecture-timetabling	20	0.31	0	**0.93**	0
logic-synthesis	20	**1.00**	0	0.89	0
market-split	20	0.19	11	**0.35**	11
miplib-neos	20	0.67	5	**0.81**	3
miplib-various	20	0.48	10	**0.84**	2
netlib-various	20	0.43	11	**0.58**	7
number-factorization	20	0.20	16	**1.00**	0
plan-museum-visits	20	0.40	12	**0.89**	2
prime-implicants	20	0.55	9	**0.75**	5
radar-station-alloc	12	**1.00**	0	0.16	10
repair-bionet	20	0.82	0	**1.00**	0
transport-systems	20	0.89	0	**1.00**	0
transportation	20	0.20	16	**0.71**	4
unibo-various	20	0.18	15	**0.77**	4
upgradability	20	0.76	1	**1.00**	0
vm-workload	20	**0.60**	8	0.44	11
wnq	16	**1.00**	0	0.42	0
workshift-design	20	0.12	17	**0.98**	0
All domains	865	0.73	194	**0.84**	89
Wins (#domains)			6		**18**

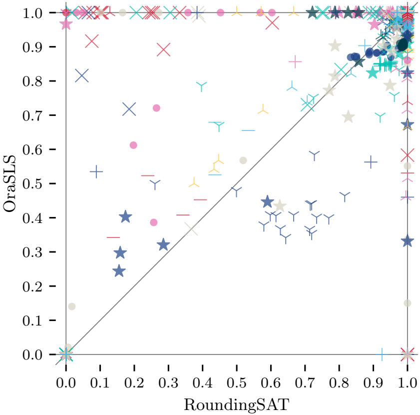
OraSLS vs. SIS and SIS+

Domains with more than 0.1 score difference

		SIS		SIS+		OraSLS	
Benchmark domain	**#**	**Score**	**#TO**	**Score**	**#TO**	**Score**	**#TO**
MANETS	20	0.45	0	0.74	2	**0.87**	2
aries-da-nrp	20	0.44	1	0.63	2	**0.68**	2
course-alloc	6	0.54	0	**0.93**	0	0.92	0
cryptography	11	0.59	0	0.66	0	**0.73**	0
decomp	10	0.33	6	0.98	0	**0.99**	0
haplotype-inf	20	0.61	0	0.74	0	**0.77**	0
miplib-neos	20	0.69	5	0.72	5	**0.81**	3
netlib-various	20	0.34	11	0.37	11	**0.58**	7
repair-bionet	20	0.00	0	**1.00**	0	**1.00**	0
transportation	20	0.48	10	0.44	10	**0.71**	4
unibo-various	20	0.46	8	0.55	8	**0.77**	4
upgradability	20	0.26	1	0.95	1	**1.00**	0
wnq	16	**0.53**	0	0.41	0	0.42	0
workshift-design	20	0.53	0	0.94	0	**0.98**	0
All domains	865	0.74	114	0.81	111	**0.84**	89
Wins (#domains)			1		2		**12**

OraSLS vs. LS-PBO (left) and RoundingSAT (right)





Conclusions

- Anytime constraint optimization finds applications in numerous applications.
- We study the integration of oracle-based SLS and solution-improving search in PBO.
- A careful combination of the two achieves state-of-the-art performance
  - While being orthogonal with previous approaches.

The solver is available at:

`https://bitbucket.org/coreo-group/orasls/src/master/`

Bibliography I

Jeremias Berg, Emir Demirovic, and Peter J. Stuckey. Core-boosted linear search for incomplete maxsat. In *CPAIOR*, volume 11494 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2019.

Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *J. Satisf. Boolean Model. Comput.*, 7(2-3):59–6, 2010. doi: 10.3233/sat190075.

Emir Demirovic and Peter J. Stuckey. Techniques inspired by local search for incomplete maxsat and the linear algorithm: Varying resolution and solution-guided search. In *CP*, volume 11802 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2019.

Jo Devriendt, Ambros Gleixner, and Jakob Nordström. Learn to relax: Integrating 0-1 integer linear programming with pseudo-Boolean conflict-driven search. *Constraints*, 2021a. ISSN 1383-7133. doi: 10.1007/s10601-020-09318-x.

Jo Devriendt, Stephan Gocht, Emir Demirovic, Jakob Nordström, and Peter J. Stuckey. Cutting to the core of pseudo-boolean optimization: Combining core-guided search with cutting planes reasoning. In *AAAI*, pages 3750–3758. AAAI Press, 2021b.

Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into SAT. *J. Satisf. Boolean Model. Comput.*, 2(1-4):1–26, 2006. doi: 10.3233/SAT190014. URL https://doi.org/10.3233/sat190014.

Jan Elffers and Jakob Nordström. Divide and conquer: Towards faster pseudo-Boolean solving. In Jérôme Lang, editor, *IJCAI*, pages 1291–1299. ijcai.org, 2018.

Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. Conflict-driven answer set solving. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, page 386, 2007. URL http://ijcai.org/Proceedings/07/Papers/060.pdf.

Alexander Nadel. Anytime weighted MaxSAT with improved polarity selection and bit-vector optimization. In *FMCAD*, pages 193–202. IEEE, 2019.

Alexander Nadel. Polarity and variable selection heuristics for SAT-based anytime MaxSAT. *J. Satisf. Boolean Model. Comput.*, 12(1):17–22, 2020.

Hossein M. Sheini and Karem A. Sakallah. Pueblo: A hybrid pseudo-boolean SAT solver. *J. Satisf. Boolean Model. Comput.*, 2(1-4):165–189, 2006. doi: 10.3233/SAT190020. URL https://doi.org/10.3233/sat190020.

Bibliography II

Pavel Smirnov, Jeremias Berg, and Matti Järvisalo. Pseudo-boolean optimization by implicit hitting sets. In *CP*, volume 210 of *LIPIcs*, pages 51:1–51:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

Pavel Smirnov, Jeremias Berg, and Matti Järvisalo. Improvements to the implicit hitting set approach to pseudo-boolean optimization. In *SAT*, volume 236 of *LIPIcs*, pages 13:1–13:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.