A MIP Perspective on Pseudo-Boolean Optimization

Alexander Tesch
Bool AI
tesch@boolai.com

1st SLOPPY Workshop 2024,
Lund University, Sweden
06. November 2024

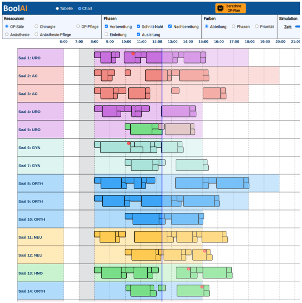
Outline

Motivation

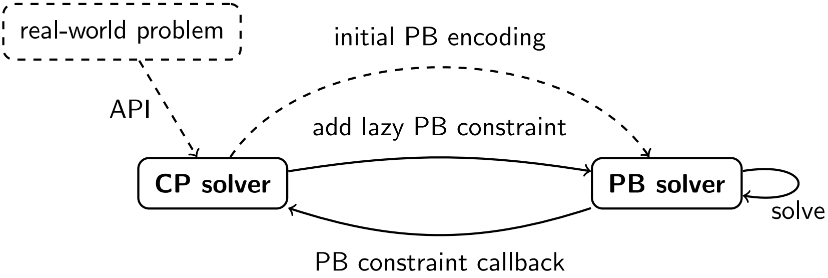
- formerly **Zuse Institute Berlin (ZIB)**
- **industries**: *healthcare, railways, steel production, conference scheduling*
- founded **own company** about 2 years ago

- development of a **CP solver** (easy API)
- runs a **lazy PB solver** at its core
- written in **C** (fast, memory management)

- main applications: **hospital optimization** (scheduling, rostering, planning, ...)



LazyCP - Solver Concept



- requires **lazy encodings** of CP constraints
- heavy focus on **adding PB constraints** during runtime
- automated **callbacks**

Example: `all_different` Constraint

Given: variables $y_1, \ldots, y_n$ with finite domains $D_1, \ldots, D_n$

$$\texttt{all\_different}((y_i)_{i=1}^n, (D_i)_{i=1}^n) \iff y_i \neq y_j \quad \forall i < j$$

Binary encoding: $x_{iv} = 1$ iff $y_i = v$ and $x_{iv} = 0$ otherwise

$$\sum_{v \in D_i} x_{iv} = 1 \qquad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{iv} \leq 1 \qquad \forall v \in D_1 \cup \ldots \cup D_n$$

Contains **PHP formulas** as special case $\implies$ exponentially stronger than CP with **lazy clause generation** (Ohrimenko et al. '09).

Pseudo-Boolean Solving Engine

LazyPB Features:

- **CDCL** as solving algorithm
- **lazy adding** of PB constraints by **(user-)callbacks**
- specific constraint handling: **clauses, cardinality, PB constraints**
- specific handling of **watched literals**
- PB conflict analysis: **clausal, MIR, lifted cover**
- restarts + constraint deletion strategies
- preprocessing: **coefficient strengthening, redundant literal detection**
- literal selection similar to **VSIDS**
- no phase saving, separate counters for x_i and $\bar{x}_i$

Currently:

- rather "standard" features implemented
- improve by more sophisticated techniques (also from MIP)
- solve **optimization problems** (cost function)
- focus on **performance**: fast feasible solutions + strong lower bounds (or UNSAT certificates)

Outline

Mixed-Integer Programming

- **cost vector** $c \in \mathbb{R}^n$
- **constraint matrix** $A \in \mathbb{R}^{m \times n}$ with **right-hand side** $b \in \mathbb{R}^m$
- **variables** x_i with $i = 1, \ldots, n$ which may take **real** or **integer** values

We solve the system:

$$\begin{aligned} \min c^T x \\ Ax \geq b \\ x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2} \end{aligned}$$

- many combinatorial problems can be modeled this way
- highly used in *operations research*

Modern MIP Solvers

- LP solver: **Simplex** and **Interior Point** (Barrier)
- **Branch-and-Bound** (with Dual Simplex)
- **additional techniques:** *presolving, node selection, branching rules, heuristics, conflict analysis, symmetry handling, ...*
- **cutting plane generation:** *GC, MIR, cover, flow-cover, disjunctive, ...* (mostly in root node)

Branch-and-Bound

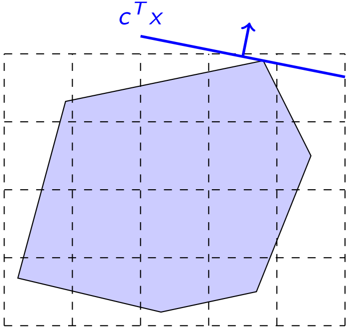
- **solve LP relaxation** with solution x^*
- **if** x^* is integer $\rightarrow$ **return** x^* optimal
- **else select** fractional $x_i^* = v$ and **branch** $x_i^* \geq \lceil v \rceil$ and $x_i^* \leq \lfloor v \rfloor$, **repeat** for each subproblem **recursively**

Note:

- B&B is more like DPLL - but with an LP solver!
- LP solving very often compensates for the absence of CDCL

Geometry of MIP

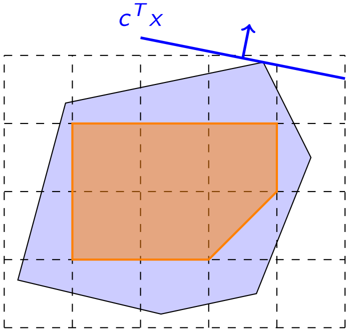
In MIP, we often look on the geometry of the solution space. A (bounded) MIP defines a polytope:



Geometry plays a crucial role for solving MIPs (and 0-1 IPs)!

Geometry of MIP

In MIP, we often look on the geometry of the solution space. A (bounded) MIP defines a polytope:

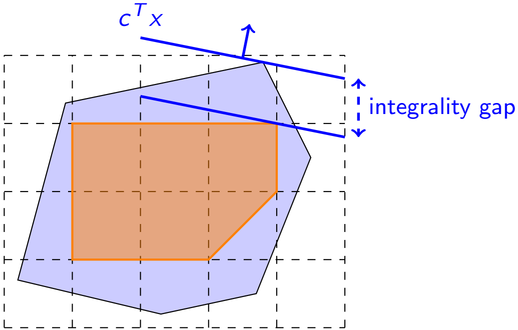


Geometry plays a crucial role for solving MIPs (and 0-1 IPs)!

- we want **strongest possible descriptions** (integer hull)
- **facets** define strongest possible inequalities
- complete description: solve LP $\implies$ solve IP

Geometry of MIP

In MIP, we often look on the geometry of the solution space. A (bounded) MIP defines a polytope:



Geometry plays a crucial role for solving MIPs (and 0-1 IPs)!

- we want **strongest possible descriptions** (integer hull)
- **facets** define strongest possible inequalities
- complete description: solve LP $\implies$ solve IP

MIP "Proof Systems"

- let $P = \{x \in \mathbb{R}^n : Ax \geq b\}$ and $P_I = conv(P \cap \mathbb{Z}^n)$
- consider multipliers $y_j \geq 0$ for each constraint C_j, the following linear inequality is valid:

$$\lceil y^T A \rceil x \geq \lceil y^T b \rceil$$

- define the (first) Chvátal-closure by

$$P^{(1)} = \bigcap_{y \geq 0} \left\{ x \in P^{(0)} : \lceil y^T A \rceil x \geq \lceil y^T b \rceil \right\}$$

- $P^{(1)}$ is a **polytope** (assuming $P^{(0)}$ was a polytope)
- we can **repeat** this procedure on $P^{(1)}$ to get $P^{(2)}$, then $P^{(3)}, \ldots$

MIP "Proof Systems"

- let $P = \{x \in \mathbb{R}^n : Ax \geq b\}$ and $P_I = conv(P \cap \mathbb{Z}^n)$
- consider multipliers $y_j \geq 0$ for each constraint C_j, the following linear inequality is valid:

$$\lceil y^T A \rceil x \geq \lceil y^T b \rceil$$

- define the (first) Chvátal-closure by

$$P^{(1)} = \bigcap_{y \geq 0} \left\{ x \in P^{(0)} : \lceil y^T A \rceil x \geq \lceil y^T b \rceil \right\}$$

- $P^{(1)}$ is a **polytope** (assuming $P^{(0)}$ was a polytope)
- we can **repeat** this procedure on $P^{(1)}$ to get $P^{(2)}$, then $P^{(3)}, \ldots$

- **Theorem (Chvátal):** There exists a *finite* minimum $k \in \mathbb{N}$ such that $P_I = P^{(k)}$. This number k is called the *Chvátal rank* of P.

Outline

0-1 Integer Linear Programming

Essentially, a 0-1 integer (or binary) program:

$$\min c^T x$$
$$Ax \geq b$$
$$x \in \{0,1\}^n$$

- equivalent to pseudo-Boolean optimizaton
- defines a 0-1 polytope (subset of the unit cube)
- rich history of cutting plane theory

- **Theorem (Eisenbrand & Schulz '99):** *The Chvátal rank of 0-1 polytopes is* $O(n^2 \log n)$.
- **meaning:** "depth" of cutting plane proofs for 0-1 ILP is polynomial!

Knapsack Polytope

Consider a simple **knapsack inequality** of the form:

$$\sum_{i=1}^{n} a_i x_i \leq b$$

with $x_i \in \{0, 1\}$, $a_i \geq 0$ and $b > 0$.

- complete descriptions for the knapsack polytope are unknown in general
- many different classes of facets were defined and studied:
  (lifted) cover inequalities, (extended) strong covers, MIR, ...

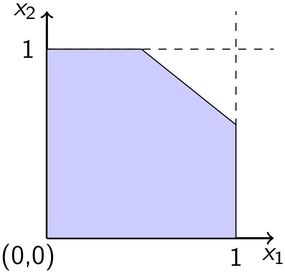
Knapsack Covers

A **knapsack cover** is a subset $C \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in C} a_i > b$. Hence, the inequality

$$\sum_{i \in C} x_i \leq |C| - 1$$

is valid. A cover is **minimal** if $\sum_{i \in C \setminus \{j\}} a_i \leq b$ for all $j \in C$.

- **Example:** $4x_1 + 6x_2 \leq 7$ with $x_1, x_2 \in \{0, 1\}$



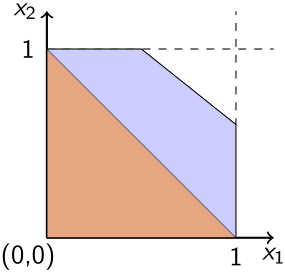
Knapsack Covers

A **knapsack cover** is a subset $C \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in C} a_i > b$. Hence, the inequality

$$\sum_{i \in C} x_i \leq |C| - 1$$

is valid. A cover is **minimal** if $\sum_{i \in C \setminus \{j\}} a_i \leq b$ for all $j \in C$.

- **Example:** $4x_1 + 6x_2 \leq 7$ with $x_1, x_2 \in \{0, 1\}$



- $x_1 + x_2 \leq 1$ is a minimal cover inequality.

Lifted Knapsack Covers

Assume we are given a **knapsack inequality** $\sum_{i=1}^{n} a_i x_i \leq b$ and an associated **minimal cover inequality** $\sum_{i \in C} x_i \leq |C| - 1$.

Consider an element $j \notin C$. We want to extend the cover inequality to

$$\sum_{i \in C} x_i + \alpha_j x_j \leq |C| - 1$$

How large can α_j be at maximum?

Lifted Knapsack Covers

Assume we are given a **knapsack inequality** $\sum_{i=1}^{n} a_i x_i \leq b$ and an associated **minimal cover inequality** $\sum_{i \in C} x_i \leq |C| - 1$.

Consider an element $j \notin C$. We want to extend the cover inequality to

$$\sum_{i \in C} x_i + \alpha_j x_j \leq |C| - 1$$

How large can α_j be at maximum? Assume $x_j = 1$ and compute

$$\alpha_j \leq |C| - 1 - \max \left\{ \sum_{i \in C} x_i : \sum_{i=1}^{n} a_i x_i \leq b, x_j = 1 \right\} = |C| - 1 - \phi$$

For a maximum coefficient set $\alpha_j = \phi$. The value ϕ can be computed by solving a **knapsack problem** with dynamic programming in $O(b \cdot |C|)$ (integer data).

Sequential Lifting

We can repeat this procedure for the already computed lifting coefficients. Hence, consider the sequence $\{j_1, \dots, j_k\}$ with $k = n - |C|$ of all $j \notin C$ and compute sequentially

$$\alpha_{j_{l+1}} = |C| - 1 - \max \left\{ \sum_{i \in C} x_i + \sum_{r=1}^{l} \alpha_{j_r} x_{j_r} : \sum_{i=1}^{n} a_i x_i \leq b, x_{j_{l+1}} = 1 \right\}$$

This yields the **lifted cover inequality**:

$$\sum_{i \in C} x_i + \sum_{r=1}^{k} \alpha_{j_r} x_{j_r} \leq |C| - 1$$

- the sequence $\{j_1, \dots, j_k\}$ is called the **lifting sequence**
- **different lifting sequences** yield **different inequalities**
- lifted cover inequalities define **facets** of the associated **knapsack polytope**
- **not all facets** of the knapsack polytope are lifted cover inequalities
- **complexity:** $O(bn^2)$ with DP but can be improved to $O(bn)$

Some Relaxations

Full sequential lifting in $O(bn)$ can still be **too slow**. We can look at **relaxations**.

Index Approach (Balas '75)
Assume we **sort the coefficients** of the cover elements $j \in C$ in *non-increasing order*, i.e. $\{j_1, \ldots, j_k\}$ with $a_{j_r} \geq a_{j_{r+1}}$ and let $A_r = \sum_{h=1}^{r} a_{j_r}$ for $r = 0, \ldots, k$.

For each element $j \notin C$ we can let

$$\alpha_j = \arg\max\{A_r \leq a_j : r = 0, \ldots, k\}$$

Complexity: $O(|C| \log |C|)$ but weaker than full sequential lifting

There are other **relaxations** and **approximations** for lifted cover inequalities.

Sequential Lifting - Example

Consider the **knapsack constraint**:

$$10x_1 + 6x_2 + 6x_3 + 4x_4 + 4x_5 + 2x_6 \leq 13$$

and the minimal cover $C = \{2, 3, 4\}$ with the **minimal cover inequality**

$$x_2 + x_3 + x_4 \leq 2$$

Consider the **lifting sequence** $\{6, 5, 1\}$ then we get

$$\alpha_6 = 2 - \max\{x_2 + x_3 + x_4 : 6x_2 + 6_3 + 4x_4 \leq 11\} = 0$$
$$\alpha_5 = 2 - \max\{x_2 + x_3 + x_4 : 6x_2 + 6_3 + 4x_4 \leq 9\} = 1$$
$$\alpha_1 = 2 - \max\{x_2 + x_3 + x_4 + x_5 : 6x_2 + 6_3 + 4x_4 + 4x_5 \leq 3\} = 2$$

The final **lifted cover inequality** is

$$2x_1 + x_2 + x_3 + x_4 + x_5 \leq 2$$

The 'relaxed' lifted cover inequality would be

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

Mixed Integer Rounding

Assume a PB constraint of the form:

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

For a given divisor $d \in \mathbb{N}$, let the **Mixed-Integer Rounding (MIR)** inequality be given by

$$\sum_{i \in I_1} \left\lceil \frac{a_i}{d} \right\rceil x_i + \sum_{i \in I_2} \left(\left\lfloor \frac{a_i}{d} \right\rfloor + \frac{f(a_i/d)}{f(b/d)} \right) x_i \geq \left\lceil \frac{b}{d} \right\rceil$$

with the partition

$$i \in I_1 \iff f(a_i/d) \geq f(b/d) \text{ or } f(a_i/d) \in \mathbb{Z}$$
$$i \in I_2 \iff f(a_i/d) < f(b/d) \text{ and } f(a_i/d) \notin \mathbb{Z}$$

where $f(x) = x - \lfloor x \rfloor$.

Mixed-Integer Rounding - Example

Consider the PB constraint

$$8x_1 + 7x_2 + 6x_3 + 4x_4 \geq 16$$

which yields the **MIR inequalities**:

$$
\begin{aligned}
d = 2: &\quad 4x_1 + 4x_2 + 3x_3 + 2x_4 \geq 8 \\
d = 3: &\quad 3x_1 + 3x_2 + 2x_3 + 2x_4 \geq 6 \\
d = 4: &\quad 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \\
d = 5: &\quad 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \\
d = 6: &\quad 1.5x_1 + 1.25x_2 + x_3 + x_4 \geq 3 \\
d = 7: &\quad 1.5x_1 + x_2 + x_3 + x_4 \geq 3 \\
d = 8: &\quad 1.5x_1 + x_2 + x_3 + x_4 \geq 2
\end{aligned}
$$

However, a lifted cover inequality with the minimum cover $C = \{3, 4\}$ and lifting the sequence $\{1, 2\}$ yields the **lifted cover inequality** (facet)

$$
\begin{aligned}
8\bar{x}_1 + 7\bar{x}_2 + 6\bar{x}_3 + 4\bar{x}_4 \leq 9 \overset{cover}{\Longrightarrow} \bar{x}_3 + \bar{x}_4 \leq 1 &\overset{lifting}{\Longrightarrow} \bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 \leq 1 \\
&\overset{sense}{\Longrightarrow} \bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 \geq 3
\end{aligned}
$$

Hence, the lifted cover inequality **dominates** the MIR inequality.

Outline

PB Solvers vs. MIP Solvers

- PB solvers are based on SAT methodology
- solving method: **conflict-driven constraint (clause) learning**

  $\Longrightarrow$ conflict analysis **is** the solving method, no branching!

Pros:

- PB constraint propagation is very fast (faster than LP)
- can generate strong non-trivial cutting planes during conflict analysis
- often good for problems where LP-relaxation is weak (e.g. big-M)

Cons:

- no "global view" on the problem: conflicts may be detected later
- optimization can be difficult: no dual information

Pseudo-Boolean Conflict Analysis

Given a **conflict constraint** $C_{conflict}$ and a **reason constraint** C_{reason}.

C_{reason} propagates a literal x_i which is falsified in $C_{conflict}$.

Conflict Analysis in RoundingSAT (Elffers & Nordström '18):

1. $C_{weaken} \leftarrow$ weaken all non-false literals in C_{reason} that are not a multiple of a_i (coefficient of x_i in C_{reason})
2. divide C_{weaken} by a_i
3. cancel out x_i by adding C_{weaken} and $C_{conflict}$

Note:

- in general, adding $C_{conflict}$ and C_{reason} to cancel out x_i may lead to a non-conflicting constraint
- the LP-relaxation may still contain a feasible point but not the IP!
- we need the constraint (resp. the integer LP) to propagate tightly

Pseudo-Boolean Conflict Analysis

During unit propagation, observe a **conflict constraint**:

$$C_{conflict} : 2x_1 + 6x_2 + 5x_3 + x_4 + 3x_5 \geq 8$$

Look at the last propagated literal, say $\bar{x}_2$, and look at its **reason constraint**:

$$C_{reason} : 6x_1 + 3\bar{x}_2 + 3x_4 + 5x_6 + 7x_7 \geq 11$$

Weakening in RoundingSAT:

$$C'_{reason} : \quad 6x_1 + 3\bar{x}_2 + 3x_4 + 6x_7 \geq 6$$
$$C''_{reason} : \quad 2x_1 + \bar{x}_2 + x_4 + 2x_7 \geq 2$$

Partial Weakening:

$$C'_{reason} : \quad 6x_1 + 3\bar{x}_2 + 3x_4 + 3x_6 + 7x_7 \geq 9$$
$$C''_{reason} : \quad 2x_1 + \bar{x}_2 + x_4 + x_6 + 3x_7 \geq 3$$

Conflict Analysis in LazyPB

1. $C_{cover} \leftarrow$ minimum cover constraint from C_{reason} which propagates x_i
2. $C_{lifted} \leftarrow$ perform lifting on literals in $C_{reason} \setminus C_{cover}$ which yields C_{lifted}
3. cancel out x_i by adding C_{lifted} and $C_{conflict}$

Note: lifting does not change the slack of C_{cover}

Back to Example:

$$\begin{aligned} C_{reason}: &\quad 6x_1 + 3\bar{x}_2 + 3x_4 + 5x_6 + 7x_7 \geq 11 \\ \iff &\quad 6\bar{x}_1 + 3x_2 + 3\bar{x}_4 + 5\bar{x}_6 + 7\bar{x}_7 \leq 13 \end{aligned}$$

$$\begin{aligned} C_{cover}: &\quad \bar{x}_1 + x_2 + \bar{x}_7 \leq 2 \\ C_{lifted}: &\quad \bar{x}_1 + x_2 + \bar{x}_6 + \bar{x}_7 \leq 2 \\ \iff &\quad x_1 + \bar{x}_2 + x_6 + x_7 \geq 2 \end{aligned}$$

MIR with division of 3 yields some seemingly incomparable constraint:

$$2x_1 + \bar{x}_2 + x_4 + 2x_6 + 2.5x_7 \geq 4$$

Notes on Pseudo-Boolean Conflict Analysis

- we can also convert $C_{conflict}$ to a lifted cover inequality
- **lifted covers** and **MIR** are **complementary**, not exclusive

- weakening seems somewhat "odd" to me
- the **strongest possible** tightly propagating constraints are the **facets** of the associated knapsack polytope
- tons of research on facets of knapsack polytopes

- Which **tightly propagating facet** should we use?
- Which **lifting sequence** to prefer? High activation literals? (currently some experiments)

Conjecture:

- stronger LP-relaxations lead to better PB solving
- Can we measure this relationship somehow?

Some Computational Results

Scheduling:

- RCPSP: achieve results close to state-of-the-art, sometimes better

PSPLib	# Inst.	SAT #opt	SAT #conf	PB #opt	PB #confl
J30	480	480	5577	480	5426
J60	480	426	70295	423	52866

MIPLIB2017 (integer version):

- solver not yet competitive for pure optimization problems
- some "interesting" instances after 300s

Instance	obj SAT	obj PB	#conflicts SAT	#conflicts PB
circ10-3	390.00	-	264,599	106,696
decomp2	-160.00*	76.00	310,538	225,539
neos-953928	-99.75	-	220,826	50,670
cvs16r70-62	-32.00	-39.00	253,813	108,619
neos-3555904-turama	-	-34.70*	121,797	6,377

Solution time for large instances:

- ivu59 (2.5 million vars and cons): 29s.
- ivu06-big (2.2 million vars and cons): 5.5s.
- supportcase11: (8 million vars and 17 millions cons): 0.58s.

Outline

Research Interests

- MIP: **"first"** order method (duals)
- PBO: **"zero"** order method (no duals)

- How to "simulate" dual information in PBO?
- How to use information of more than two PB constraints in conflict analysis or propagation?

$$\frac{C_1 \oplus \cdots \oplus C_k}{D}$$

- are there any other strong derivation rules? (proof complexity?)
- effect of stronger LP-relaxations on PBO?
- column-generation in PBO?
- technical details on core-guided search