Solving Discrete Optimisation Problems using Satisfiability Solvers

Department seminar at KU Leuven

Hendrik 'Henk' Bierlee
Peter J. Stuckey Jip J. Dekker Guido Tack Graeme Gange

Monash University, Department of Data Science and AI
Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA)

Thursday 13th June, 2024

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure: From https://en.wikipedia.org/wiki/Sudoku

Modeling Sudoku as CSP in MiniZinc

Code Listing: sudoku.mzn

```
array [1..9, 1..9] of var 1..9: solution;

% Rows are all different
constraint forall (i in 1..9) (all_different(solution[i, ..]));
% Columns are all different
constraint forall (j in 1..9) (all_different(solution[.., j]));
% Subgrids are all different
constraint forall (i, j in 1..3) (
all_different(solution[
  3 * (i - 1) + 1 .. 3 * i,
  3 * (j - 1) + 1 .. 3 * j
])
);
```

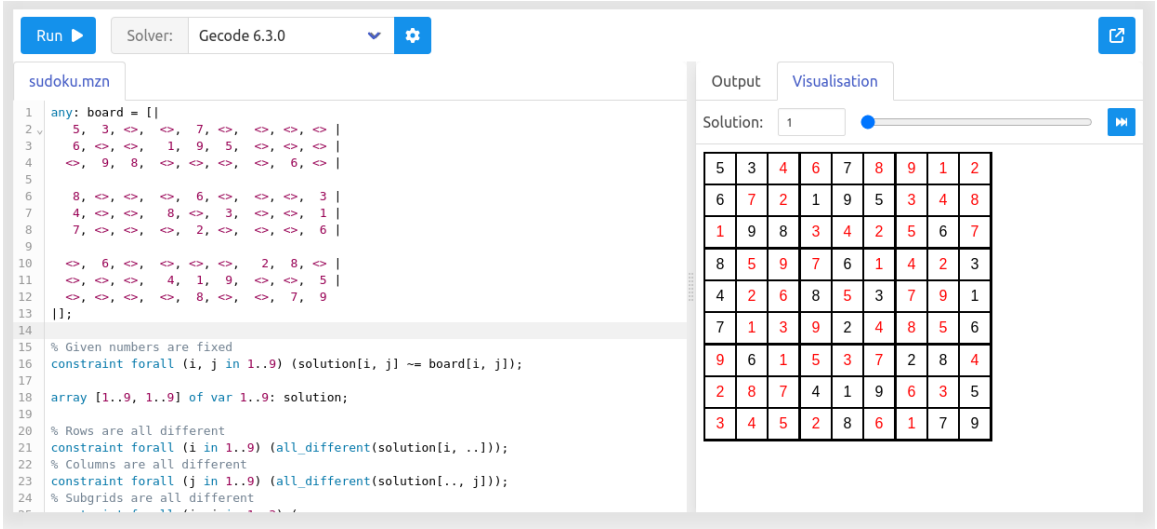

Figure: From https://www.minizinc.org/

Modeling Sudoku as CSP in Python and CPMpy

```
model = Model(
# Rows are all different
        [AllDifferent(row) for row in solution],
# Columns are all different
        [AllDifferent(col) for col in solution.T], # numpy's Transpose
)

# Subgrids are all different
for i in range(0,9,3):
for j in range(0,9,3):
model += AllDifferent(solution[i:i+3, j:j+3]) # python's indexing

# Given numbers are fixed
model += (solution[board!=e] == board[board!=e]) # numpy's indexing
```

From CSP to SAT

- A CSP $P = (\mathcal{X}^P, D, \mathcal{C}^P)$ has decision variables $\mathcal{X}^P$, domains D and constraints $\mathcal{C}^P$
  - **Ex.** $\mathcal{X}^P = \{x_{1,1}, x_{1,2}, x_{1,3}, \ldots\}, \quad D(x_{i,j}) = [1..9], \quad \mathcal{C}^P = \{x_{1,1} \neq x_{1,2} \neq x_{1,3} \neq \ldots\}$
- The SAT problem S is a special case of CSP
  - Every $b \in \mathcal{X}^S$ has $D(b) = \mathcal{B} = \{\text{false}, \text{true}\} = \{0, 1\}$
    - A *literal* l is b or its negation $\neg b$
    - **Ex.** $(x \vee y) \wedge (\neg x \vee \neg y)$

From CSP to SAT

- A CSP $P = (\mathcal{X}^P, D, \mathcal{C}^P)$ has decision variables $\mathcal{X}^P$, domains D and constraints $\mathcal{C}^P$
  - **Ex.** $\mathcal{X}^P = \{x_{1,1}, x_{1,2}, x_{1,3}, \ldots\}, \quad D(x_{i,j}) = [1..9], \quad \mathcal{C}^P = \{x_{1,1} \neq x_{1,2} \neq x_{1,3} \neq \ldots\}$
- The SAT problem S is a special case of CSP
  - Every $b \in \mathcal{X}^S$ has $D(b) = \mathcal{B} = \{\text{false}, \text{true}\} = \{0, 1\}$
    - A *literal* l is b or its negation $\neg b$
    - **Ex.** $(x \vee y) \wedge (\neg x \vee \neg y) \Rightarrow x = 1$

From CSP to SAT

- A CSP $P = (\mathcal{X}^P, D, \mathcal{C}^P)$ has decision variables $\mathcal{X}^P$, domains D and constraints $\mathcal{C}^P$
  - **Ex.** $\mathcal{X}^P = \{x_{1,1}, x_{1,2}, x_{1,3}, \ldots\}, \quad D(x_{i,j}) = [1..9], \quad \mathcal{C}^P = \{x_{1,1} \neq x_{1,2} \neq x_{1,3} \neq \ldots\}$
- The SAT problem S is a special case of CSP
  - Every $b \in \mathcal{X}^S$ has $D(b) = \mathcal{B} = \{\text{false}, \text{true}\} = \{0, 1\}$
    - A *literal* l is b or its negation $\neg b$
    - **Ex.** $(x \vee y) \wedge (\neg x \vee \neg y) \Rightarrow x = 1, y = 0$

From CSP to SAT

- A CSP $P = (\mathcal{X}^P, D, \mathcal{C}^P)$ has decision variables $\mathcal{X}^P$, domains D and constraints $\mathcal{C}^P$
  - **Ex.** $\mathcal{X}^P = \{x_{1,1}, x_{1,2}, x_{1,3}, \ldots\}, \quad D(x_{i,j}) = [1..9], \quad \mathcal{C}^P = \{x_{1,1} \neq x_{1,2} \neq x_{1,3} \neq \ldots\}$
- The SAT problem S is a special case of CSP
  - Every $b \in \mathcal{X}^S$ has $D(b) = \mathcal{B} = \{\text{false}, \text{true}\} = \{0, 1\}$
    - A *literal* l is b or its negation $\neg b$
    - **Ex.** $(x \vee y) \wedge (\neg x \vee \neg y) \Rightarrow x = 1, y = 0 \quad (\equiv x \oplus y)$
  - $\mathcal{C}^S$ contains only *clauses* (i.e., disjunctions of literals)
- Modern SAT solvers are very fast
  - Solve CSP P by encoding it as an equisatisfiable SAT problem S

Direct, order and binary encoding of integer variables

Table: All assignments of x and of its encodings for $D(x)=[0..3]$

x
0
1
2
3

Direct, order and binary encoding of integer variables

Table: All assignments of x and of its encodings for $D(x)=[0..3]$

x	$[[x=0]]$	$[[x=1]]$	$[[x=2]]$	$[[x=3]]$
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

- **Direct encoding** $x{:}\mathbb{D}$ $\quad [\![x = v]\!]$ encodes *equality* of x to some value v

Direct, order and binary encoding of integer variables

Table: All assignments of x and of its encodings for $D(x)=[0..3]$

x	$[[x=0]]$	$[[x=1]]$	$[[x=2]]$	$[[x=3]]$	$[[x\geq 1]]$	$[[x\geq 2]]$	$[[x\geq 3]]$
0	1	0	0	0	0	0	0
1	0	1	0	0	1	0	0
2	0	0	1	0	1	1	0
3	0	0	0	1	1	1	1

- **Direct encoding** $x{:}\mathbb{D}$ $\quad [\![x = v]\!]$ encodes *equality* of x to some value v
- **Order encoding** $x{:}\mathbb{O}$ $\quad [\![x \geq v]\!]$ encodes *inequality* of x to some value v

Direct, order and binary encoding of integer variables

Table: All assignments of x and of its encodings for $D(x)=[0..3]$

x	$[[x=0]]$	$[[x=1]]$	$[[x=2]]$	$[[x=3]]$	$[[x\geq 1]]$	$[[x\geq 2]]$	$[[x\geq 3]]$	$[[\text{bit}(x,1)]]$	$[[\text{bit}(x,0)]]$
0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	1
2	0	0	1	0	1	1	0	1	0
3	0	0	0	1	1	1	1	1	1

- **Direct encoding** $x{:}\mathbb{D}$ $\quad [\![x = v]\!]$ encodes *equality* of x to some value v
- **Order encoding** $x{:}\mathbb{O}$ $\quad [\![x \geq v]\!]$ encodes *inequality* of x to some value v
- **Binary encoding** $x{:}\mathbb{B}$ $\quad [\![\text{bit}(x,k)]\!]$ encodes bit k in the binary representation of x

Encoding Sudoku to SAT

- Direct encoding $x_{1,1}{:}\mathbb{D}$ introduces 9 Boolean variables
  - $[\![x_{1,1} = 1]\!], [\![x_{1,1} = 2]\!], [\![x_{1,1} = 3]\!], \ldots, [\![x_{1,1} = 9]\!]$
  - $[\![x_{1,1} = 5]\!] \iff x_{1,1} = 5$
  - Inconsistent assignment: $[\![x_{1,1} = 5]\!] \wedge [\![x_{1,1} = 6]\!]$

Encoding Sudoku to SAT

- Direct encoding $x_{1,1}{:}\mathbb{D}$ introduces 9 Boolean variables
  - $[\![x_{1,1} = 1]\!], [\![x_{1,1} = 2]\!], [\![x_{1,1} = 3]\!], \ldots, [\![x_{1,1} = 9]\!]$
  - $[\![x_{1,1} = 5]\!] \iff x_{1,1} = 5$
  - Inconsistent assignment: $[\![x_{1,1} = 5]\!] \wedge [\![x_{1,1} = 6]\!]$
- Enforce **consistency** with exactly-one constraint: $\left(\sum_{d \in D(x_{1,1})} [\![x_{1,1} = d]\!]\right) = 1$
  - $\left(\sum_{d \in D(x)} [\![x = d]\!]\right) \geq 1 \equiv [\![x = 1]\!] \vee [\![x = 2]\!] \vee \ldots \vee [\![x = 9]\!]$
  - $\left(\sum_{d \in D(x)} [\![x = d]\!]\right) \leq 1 \equiv$

$$\neg[\![x = 1]\!] \vee \neg[\![x = 2]\!] \wedge \neg[\![x = 1]\!] \vee \neg[\![x = 3]\!] \qquad \wedge \neg[\![x = 1]\!] \vee \neg[\![x = 4]\!] \wedge \ldots$$
$$\wedge \neg[\![x = 2]\!] \vee \neg[\![x = 3]\!] \qquad \wedge \neg[\![x = 2]\!] \vee \neg[\![x = 4]\!] \wedge \ldots$$
$$\wedge \neg[\![x = 3]\!] \vee \neg[\![x = 4]\!] \wedge \ldots$$

Encoding Sudoku to SAT

- Direct encoding $x_{1,1}{:}\mathbb{D}$ introduces 9 Boolean variables
  - $[\![x_{1,1} = 1]\!], [\![x_{1,1} = 2]\!], [\![x_{1,1} = 3]\!], \ldots, [\![x_{1,1} = 9]\!]$
  - $[\![x_{1,1} = 5]\!] \iff x_{1,1} = 5$
  - Inconsistent assignment: $[\![x_{1,1} = 5]\!] \wedge [\![x_{1,1} = 6]\!]$
- Enforce **consistency** with exactly-one constraint: $\left(\sum_{d \in D(x_{1,1})} [\![x_{1,1} = d]\!]\right) = 1$
  - $\left(\sum_{d \in D(x)} [\![x = d]\!]\right) \geq 1 \equiv [\![x = 1]\!] \vee [\![x = 2]\!] \vee \ldots \vee [\![x = 9]\!]$
  - $\left(\sum_{d \in D(x)} [\![x = d]\!]\right) \leq 1 \equiv$

$$\neg[\![x = 1]\!] \vee \neg[\![x = 2]\!] \wedge \neg[\![x = 1]\!] \vee \neg[\![x = 3]\!] \qquad \wedge \neg[\![x = 1]\!] \vee \neg[\![x = 4]\!] \wedge \ldots$$
$$\wedge \neg[\![x = 2]\!] \vee \neg[\![x = 3]\!] \qquad \wedge \neg[\![x = 2]\!] \vee \neg[\![x = 4]\!] \wedge \ldots$$
$$\wedge \neg[\![x = 3]\!] \vee \neg[\![x = 4]\!] \wedge \ldots$$

- AllDifferent$(x_{1,1}, x_{1,2}, \ldots)$: $\left(\sum_{i=1}^{9} [\![x_{1,i} = 1]\!]\right) = 1, \left(\sum_{i=1}^{9} [\![x_{1,i} = 2]\!]\right) = 1, \ldots$

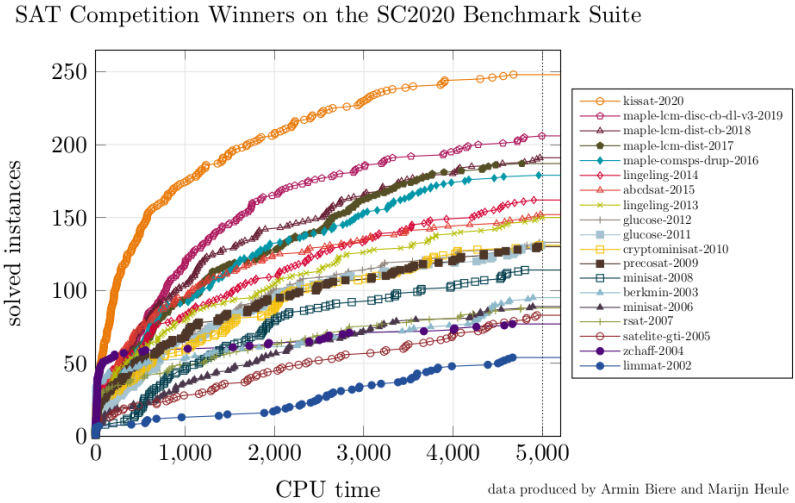

Figure: From https://x.com/ArminBiere

Medals

MiniZinc Challenge 2023

Category	Gold	Silver	Bronze
Fixed	OR-Tools	SICStus Prolog	Choco 4
Free	OR-Tools	PicatSAT	IZplus
Parallel	OR-Tools	PicatSAT	Choco 4
Local Search	Yuck		

MiniZinc Challenge 2022

Category	Gold	Silver	Bronze
Fixed	OR-Tools	SICStus Prolog	JaCoP
Free	OR-Tools	PicatSAT	Choco 4
Parallel	OR-Tools	PicatSAT	Geas
Local Search	Yuck		

MiniZinc Challenge 2021

Category	Gold	Silver	Bronze
Fixed	OR-Tools	JaCoP	SICStus Prolog
Free	OR-Tools	PicatSAT	iZplus
Parallel	OR-Tools	PicatSAT	iZplus + Choco 4
Open	OR-Tools	PicatSAT	iZplus + Choco 4
Local Search	Yuck	OscaR/CBLS	

Figure: From https://www.minizinc.org/challenge/

Different integer encodings for scheduling problems

- Schedule tasks over time horizon h
  - Every task has a start time and a duration
  - E.g., task 1 has start time x with $D(x) = 0..h$ and duration 16
  - The next task 2 with start time y cannot start before task 1 is finished: $x + 16 \leq y$

Different integer encodings for scheduling problems

- Schedule tasks over time horizon h
  - Every task has a start time and a duration
  - E.g., task 1 has start time x with $D(x) = 0..h$ and duration 16
  - The next task 2 with start time y cannot start before task 1 is finished: $x + 16 \leq y$
- $x{:}\mathbb{O}$ is the **order encoding** of x
  - h Boolean variables: $[\![x \geq 1]\!], [\![x \geq 2]\!], \ldots, [\![x \geq h]\!]$
    - A SAT solution of $[\![x \geq 2]\!] = 1 \wedge [\![x \geq 3]\!] = 0$ represents CSP solution of $x = 2$
    - Inconsistent assignment:

Different integer encodings for scheduling problems

- Schedule tasks over time horizon h
  - Every task has a start time and a duration
  - E.g., task 1 has start time x with $D(x) = 0..h$ and duration 16
  - The next task 2 with start time y cannot start before task 1 is finished: $x + 16 \leq y$
- $x{:}\mathbb{O}$ is the **order encoding** of x
  - h Boolean variables: $[\![x \geq 1]\!], [\![x \geq 2]\!], \ldots, [\![x \geq h]\!]$
    - A SAT solution of $[\![x \geq 2]\!] = 1 \wedge [\![x \geq 3]\!] = 0$ represents CSP solution of $x = 2$
    - Inconsistent assignment: $[\![x \geq 1]\!] = 0, [\![x \geq 2]\!] = 1, \ldots$

Different integer encodings for scheduling problems

- Schedule tasks over time horizon h
  - Every task has a start time and a duration
  - E.g., task 1 has start time x with $D(x) = 0..h$ and duration 16
  - The next task 2 with start time y cannot start before task 1 is finished: $x + 16 \leq y$
- $x{:}\mathbb{O}$ is the **order encoding** of x
  - h Boolean variables: $[\![x \geq 1]\!], [\![x \geq 2]\!], \ldots, [\![x \geq h]\!]$
    - A SAT solution of $[\![x \geq 2]\!] = 1 \wedge [\![x \geq 3]\!] = 0$ represents CSP solution of $x = 2$
    - Inconsistent assignment: $[\![x \geq 1]\!] = 0, [\![x \geq 2]\!] = 1, \ldots$
  - Enforce **consistency**: $[\![x \geq 2]\!] \rightarrow [\![x \geq 1]\!], [\![x \geq 3]\!] \rightarrow [\![x \geq 2]\!], \ldots$

Different integer encodings for scheduling problems

- Schedule tasks over time horizon h
  - Every task has a start time and a duration
  - E.g., task 1 has start time x with $D(x) = 0..h$ and duration 16
  - The next task 2 with start time y cannot start before task 1 is finished: $x + 16 \leq y$
- $x{:}\mathbb{O}$ is the **order encoding** of x
  - h Boolean variables: $[\![x \geq 1]\!], [\![x \geq 2]\!], \ldots, [\![x \geq h]\!]$
    - A SAT solution of $[\![x \geq 2]\!] = 1 \wedge [\![x \geq 3]\!] = 0$ represents CSP solution of $x = 2$
    - Inconsistent assignment: $[\![x \geq 1]\!] = 0, [\![x \geq 2]\!] = 1, \ldots$
  - Enforce **consistency**: $[\![x \geq 2]\!] \rightarrow [\![x \geq 1]\!], [\![x \geq 3]\!] \rightarrow [\![x \geq 2]\!], \ldots$
- Encode $x + 16 \leq y$ given $x{:}\mathbb{O}$ and $y{:}\mathbb{O}$:
  - $[\![x \geq 1]\!] \rightarrow [\![y \geq 17]\!], [\![x \geq 2]\!] \rightarrow [\![y \geq 18]\!], \ldots$
  - $x{:}\mathbb{O} + 16 \leq y{:}\mathbb{O}$ or $(x + 16 \leq y){:}\mathbb{O}$

Different integer encodings for scheduling problems

- Schedule tasks over time horizon h
  - Every task has a start time and a duration
  - E.g., task 1 has start time x with $D(x) = 0..h$ and duration 16
  - The next task 2 with start time y cannot start before task 1 is finished: $x + 16 \leq y$
- $x{:}\mathbb{O}$ is the **order encoding** of x
  - h Boolean variables: $[\![x \geq 1]\!], [\![x \geq 2]\!], \ldots, [\![x \geq h]\!]$
    - A SAT solution of $[\![x \geq 2]\!] = 1 \wedge [\![x \geq 3]\!] = 0$ represents CSP solution of $x = 2$
    - Inconsistent assignment: $[\![x \geq 1]\!] = 0, [\![x \geq 2]\!] = 1, \ldots$
  - Enforce **consistency**: $[\![x \geq 2]\!] \rightarrow [\![x \geq 1]\!], [\![x \geq 3]\!] \rightarrow [\![x \geq 2]\!], \ldots$
- Encode $x + 16 \leq y$ given $x{:}\mathbb{O}$ and $y{:}\mathbb{O}$:
  - $[\![x \geq 1]\!] \rightarrow [\![y \geq 17]\!], [\![x \geq 2]\!] \rightarrow [\![y \geq 18]\!], \ldots$
  - $x{:}\mathbb{O} + 16 \leq y{:}\mathbb{O}$ or $(x + 16 \leq y){:}\mathbb{O}$
- The **binary encoding** $x{:}\mathbb{B}$ represents $D(x)$ with only $\log h$ variables
  - $[\![\text{bit}(x, 0)]\!] = 1, [\![\text{bit}(x, 1)]\!] = 0, [\![\text{bit}(x, 2)]\!] = 1 \Rightarrow x = 5$
  - But requires more complex clauses

Coupling different integer encodings for SAT [Bierlee et al., 2022]

Ex. What if task 1 has a small time-window compared to task 2

- Situation: $|D(x)| \ll |D(y)|$
- Two uniform encoding approaches:
  - $(x + 16 \leq y)$:$\mathbb{O}$
  - $(x + 16 \leq y)$:$\mathbb{B}$
- Prefer x:$\mathbb{O}$ by annotation
  - var int: x::order_encoded;
- Resolve mixed integer encodings by:
  - *Coupling*: x:$\mathbb{O} + 16 \leq y$:$\mathbb{B}$
  - *Channelling*: y:$\mathbb{B} = y$:$\mathbb{O} \wedge x$:$\mathbb{O} + 16 \leq y$:$\mathbb{O}$

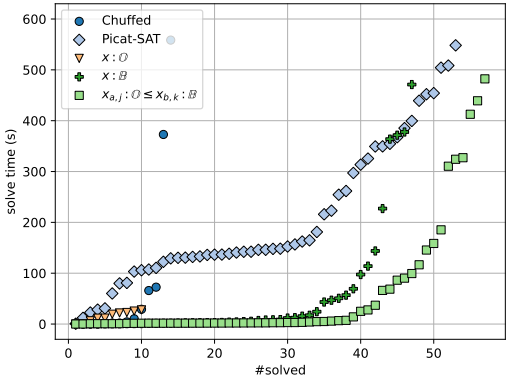

Figure: Cactus plot for JSS with(out) coupling

Channeling (equality) constraint $x{:}\mathbb{D} = y{:}\mathbb{O}$ and $x{:}\mathbb{O} = y{:}\mathbb{B}$

- $x{:}\mathbb{D} = y{:}\mathbb{O}$ channelling constraint $\forall_{d \in D(x)} [\![x = d]\!] \leftrightarrow [\![y \geq d]\!] \wedge [\![y < d+1]\!]$
- $x{:}\mathbb{O} = y{:}\mathbb{B}$ constrains for each bit $k \in \mathcal{B}(y)$, for each **segment** $r_l..r_u$ where $[\![\text{bit}(r_l, k)]\!] = [\![\text{bit}(r_l + 1, k)]\!] = \ldots = [\![\text{bit}(r_u, k)]\!]$:

$$[\![\text{bit}(r_l, k)]\!] \vee [\![x < r_l]\!] \vee [\![x \geq r_u + 1]\!]$$

Example: $D(x) = D(y) = 0..7$

The bit $k = 1$ is always **0** in its third segment, 4..5:

$$[\![\text{bit}(y, 1)]\!] \vee [\![x < 4]\!] \vee [\![x \geq 6]\!]$$

$[\![\text{bit}(y, 1)]\!]$ does not constrain $[\![x \geq 5]\!]$ since $[\![x \geq 6]\!] \rightarrow [\![x \geq 5]\!]$

	2 1 0
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Counter-Example Guided Abstract Refinement (CEGAR) for JSS

- CEGAR [Clarke et al., 2000] provides another way to encode large domains
- Encode $x{:}\mathbb{O}$ only *partially*: $[\![x \geq\ 10]\!], [\![x \geq\ 20]\!], [\![x \geq\ 30]\!], \ldots$
  - Consistency: $[\![x \geq\ 20]\!] \rightarrow [\![x \geq\ 10]\!] \wedge [\![x \geq\ 30]\!] \rightarrow [\![x \geq\ 20]\!] \wedge \ldots$
- Relax clause $[\![x \geq\ 10]\!] \rightarrow [\![y \geq\ 26]\!]$ to $[\![x \geq\ 10]\!] \rightarrow [\![y \geq\ 20]\!]$ (there is no $[\![y \geq\ 26]\!]$)
- Solving the far smaller abstract problem returns either:
  a) *Spurious* solution (e.g., $x = 10, y = 20, \ldots$). Then:
     - Add $[\![y \geq\ 26]\!]$
     - Consistency: $[\![y \geq\ 30]\!] \rightarrow [\![y \geq\ 26]\!] \wedge [\![y \geq\ 26]\!] \rightarrow [\![y \geq\ 20]\!]$
     - Constrain: $[\![x \geq\ 10]\!] \rightarrow [\![y \geq\ 26]\!]$
  b) *Non-spurious* solution (e.g., $x = 10, y = 30, \ldots$), which is *guaranteed* to be optimal

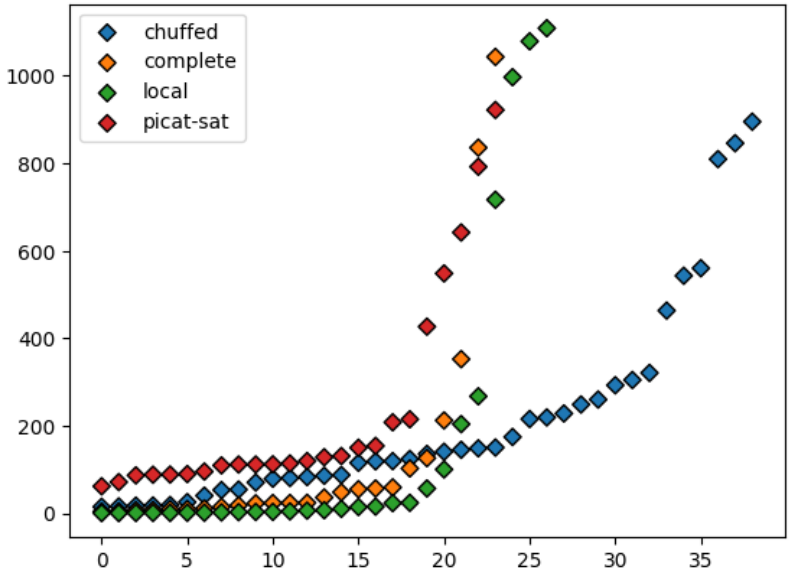

Figure: Cactus plot for JSS with CEGAR versus one-shot and chuffed (LCG)

PB encodings as decompositions

Ex. Budget constraint: $2b_1 + 3b_2 + 5b_3 \leq 6$ where $D(b_i) = \mathcal{B}$, $b_i \iff$ schedule tasks i

- Example of *Pseudo-Boolean* (PB) constraint for which many PB encodings exist
- Observation: many PB encodings follow the same pattern
  - pattern = the encoding of ternary inequality constraint, $(x + y \leq z){:}\mathbb{O}$
- But their **decomposition** is different

Theoretical result

Each PB encoding is a decomposition of *integer* ternary inequalities, encoded as:

$$(x + y \leq z){:}\mathbb{O} \equiv \bigwedge_{v \in D(x)} \bigwedge_{w \in D(y)} ([\![x \geq v]\!] \wedge [\![y \geq w]\!]) \rightarrow [\![z \geq v + w]\!]$$

Different decompositions

Decomposing and encoding with *Generalised Totaliser* (GT) [Joshi et al., 2015]:

$$
\begin{aligned}
& 2b_1 + 3b_2 + 5b_3 \leq 6 && D(b_1) = D(b_2) = D(b_3) = \{0, 1\} \\
\equiv\ & x_1 + x_2 + x_3 \leq 6 && D(x_1) = \{0, 2\}, D(x_2) = \{0, 3\}, D(x_3) = \{0, 5\} \\
\equiv\ & x_1 + x_2 \leq y, y + x_3 \leq 6 && D(y) = \{0, 2, 3, 5\} \\
\equiv\ & ([\![x_1 \geq\ 0]\!] \wedge [\![x_2 \geq\ 0]\!]) \rightarrow [\![y \geq\ 0]\!], ([\![x_1 \geq\ 0]\!] \wedge [\![x_2 \geq\ 3]\!]) \rightarrow [\![y \geq\ 3]\!], \dots
\end{aligned}
$$

Or *Binary Decision Diagram* (BDD) [Eén and Sörensson, 2006, Bailleux et al., 2006]:

$$
\begin{aligned}
& x_1 \leq y_1, && D(x_1) = \{0, 2\}, D(y_1) = \{0, 1\} \\
& x_2 + y_1 \leq y_2, && D(x_2) = \{0, 3\}, D(y_2) = \{1, 5\} \\
& x_3 + y_2 \leq 6 && D(x_3) = \{0, 5\}
\end{aligned}
$$

Other decompositions for *Sequential Weight Counter* (SWC) [Hölldobler et al., 2012] and *Sorting Network* (SN) [Bailleux et al., 2009, Eén and Sörensson, 2006, Abío et al., 2013]

Extensions that apply to all decompositions

Extensions:

- Support integer linear constraints
  - Generalizing side-constraints [Abío et al., 2015, Bofill et al., 2019]
- Support = constraints
- Support redundant consistency constraints
- Support bounds propagation
- Prove bounds consistency

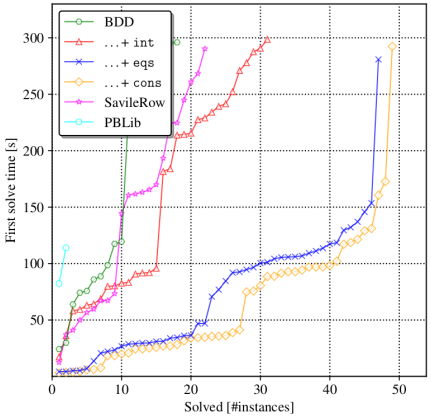

Figure: Cactus plot solving knapsack-type problem using various extensions

Mixed order-binary encoding of decompositions

- Once decomposed, the encoding is still flexible!
- For BDD, the y_i domains go from small, to large, to small

$$
\begin{aligned}
x_1{:}\mathbb{O} &\leq y_1{:}\mathbb{O}, \\
x_2{:}\mathbb{O} + y_1{:}\mathbb{O} &\leq y_2{:}\mathbb{O}, \\
x_3{:}\mathbb{O} + y_2{:}\mathbb{O} &\leq y_3{:}\mathbb{B}, \\
x_4{:}\mathbb{O} + y_3{:}\mathbb{B} &\leq y_4{:}\mathbb{B}, \\
x_5{:}\mathbb{O} + y_4{:}\mathbb{B} &\leq y_5{:}\mathbb{O}, \\
x_6{:}\mathbb{O} + y_5{:}\mathbb{O} &\leq 1234
\end{aligned}
$$

Single Constant Multiplication (SCM) for SAT [Bierlee et al., 2024]

Ex. Compute $(117 \times x)$:$\mathbb{B}$ using only additions, subtractions and shifts $(2^k \times x, k \geq 0)$

Single Constant Multiplication (SCM) for SAT [Bierlee et al., 2024]

Ex. Compute $(117 \times x)$:$\mathbb{B}$ using only additions, subtractions and shifts $(2^k \times x, k \geq 0)$

$$64 \times x - x = 63x$$
$$63x - 4 \times x = 59x$$
$$2 \times 59x\ (= 118x) - x = 117x$$

Single Constant Multiplication (SCM) for SAT [Bierlee et al., 2024]

Ex. Compute $(117 \times x)$:$\mathbb{B}$ using only additions, subtractions and shifts $(2^k \times x, k \geq 0)$

$$64 \times x - x = 63x$$
$$63x - 4 \times x = 59x$$
$$2 \times 59x\ (= 118x) - x = 117x$$

$$2 \times x + x = 3x$$
$$16 \times 3x\ (= 48x) + x = 49x$$
$$x + 16 \times x = 17x$$
$$4 \times 17x\ (= 68x) + 49x = 117x$$

Single Constant Multiplication (SCM) for SAT [Bierlee et al., 2024]

Ex. Compute $(117 \times x)$:$\mathbb{B}$ using only additions, subtractions and shifts $(2^k \times x, k \geq 0)$

$$64 \times x - x = 63x$$
$$63x - 4 \times x = 59x$$
$$2 \times 59x\ (= 118x) - x = 117x$$

$$2 \times x + x = 3x$$
$$16 \times 3x\ (= 48x) + x = 49x$$
$$x + 16 \times x = 17x$$
$$4 \times 17x\ (= 68x) + 49x = 117x$$

				x_3	x_2	x_1	x_0	x
x_3	x_2	x_1	x_0	0	0	0	0	$x \ll 4$
x_3	x_2	x_1	x_0	x_3	x_2	x_1	x_0	$17 \times x$

Single Constant Multiplication (SCM) for SAT [Bierlee et al., 2024]

Ex. Compute $(117 \times x)$:$\mathbb{B}$ using only additions, subtractions and shifts $(2^k \times x, k \geq 0)$

$$64 \times x - x = 63x$$
$$63x - 4 \times x = 59x$$
$$2 \times 59x\ (= 118x) - x = 117x$$

$$2 \times x + x = 3x$$
$$16 \times 3x\ (= 48x) + x = 49x$$
$$x + 16 \times x = 17x$$
$$4 \times 17x\ (= 68x) + 49x = 117x$$

				x_3	x_2	x_1	x_0	x
x_3	x_2	x_1	x_0	0	0	0	0	$x \ll 4$
x_3	x_2	x_1	x_0	x_3	x_2	x_1	x_0	$17 \times x$

From 3 equations using 31 adders to 4 equations using **12** adders!

Different SCM objectives for SAT

Pre-compute $c \times x$ for $1 \leq c \leq 2047$:

- min-k minimizes #nodes
- min-a minimizes #adders
  - For bit widths $1 \leq b \leq 16$

Compare:

- baseline = naive SCM solution
- espresso (Boolean minimization)
- picat-SAT = mix of base and espresso

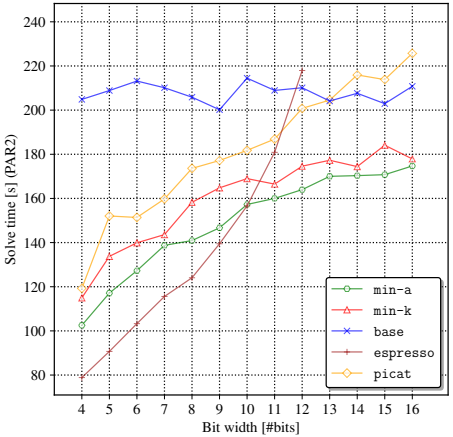

Figure: Solve times for bit widhts $1 \leq b \leq 16$

Some hard-learned lessons from SAT

How do we evaluate SAT encodings?

- Static metrics (variables, clauses, literals)
- Solver statistics (e.g., propagations, conflicts, ...)
- *Propagation completeness* is whether given an assignment of literal l and set of clauses C, unit propagation over C assigns all literals which are consequences of l
  - Propagation complete encodings using direct or order encoding respectively yield *domain* or *bounds(Z) consistency*[1]
- **Solve time** of the SAT solver (for multiple seeds and solvers)
- Compilation time of the SAT encoder

[1] [Choi et al., 2006]

Some hard-learned lessons from SAT (cont'd)

How do we verify SAT encodings?

- Testing correctness
  - Check correctness of decomposition before encoding
  - Label SAT output
  - Comprehensive testing of all combinations of configurations and integer variable encodings
- Unit test small CSPs:
  - Brute-force (and cache) all solutions
  - Full comparison to get extra and missing solutions
  - Also show assignment of auxiliary (decomposition) variables

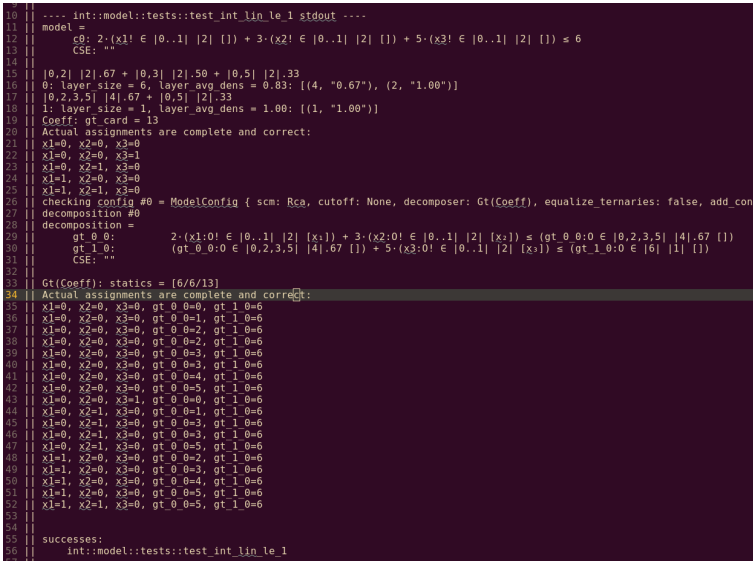

Figure: Succesfull unit test

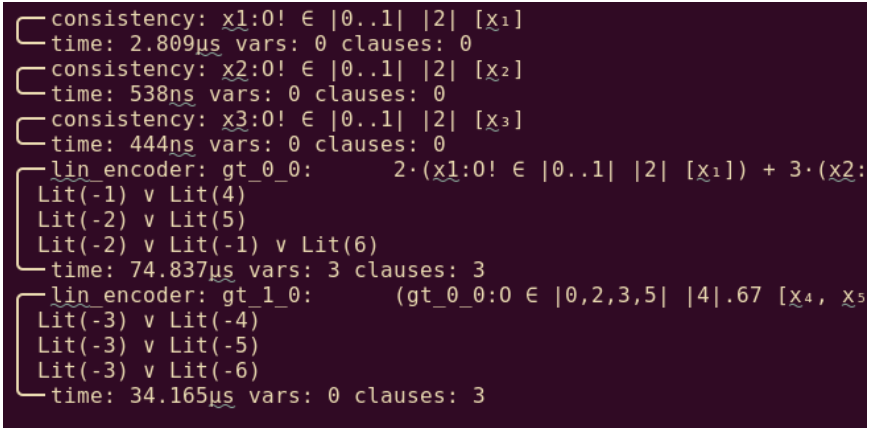

Figure: Label encoding literals

```
|| Decomposition error:
|| Inconsistency: Inconsistency in c0: 2 * x1=1 (=2) + 3 * x2=0 (=0) + 5 * x3=1 (=5) == 7 !≤ 6
||
|| Decomposition error:
||
|| ModelConfig { scm: Rca, cutoff: None, decomposer: Gt(Coeff), equalize_ternaries: false, add_
|| Extra solutions:
|| + x1=1, x2=0, x3=1
|| Missing solutions:
||
|| Expected assignments:
|| x1=0, x2=0, x3=0
|| x1=0, x2=0, x3=1
|| x1=0, x2=1, x3=0
|| x1=1, x2=0, x3=0
|| x1=1, x2=1, x3=0
|| Actual assignments:
|| x1=0, x2=0, x3=0
|| x1=0, x2=0, x3=1
|| x1=0, x2=1, x3=0
|| x1=1, x2=0, x3=0
|| x1=1, x2=0, x3=1
|| x1=1, x2=1, x3=0
```

Figure: Fail decomposition

```
decomposition =
     gt_0_0:         2·(x1:O! ∈ |0..1| |2| [x₁]) + 3·(x2:O! ∈ |0..1| |2| [x₂]) ≤ (gt_0_0:O ∈ |0,2,3,5| |4|.67 [])
     gt_1_0:         (gt_0_0:O ∈ |0,2,3,5| |4|.67 []) + 5·(x3:O! ∈ |0..1| |2| [x₃]) ≤ (gt_1_0:O ∈ |6| |1| [])
     CSE: ""

Gt(Coeff): statics = [6/5/11]
Inconsistency: Inconsistency in gt_0_0: 2 * x1=1 (=2) + 3 * x2=1 (=3) + -1 * gt_0_0=4 (=-4) == 1 !≤ 0

Inconsistency: Inconsistency in gt_1_0: 1 * gt_0_0=2 (=2) + 5 * x3=1 (=5) + -1 * gt_1_0=6 (=-6) == 1 !≤ 0

Inconsistency: Inconsistency in gt_0_0: 2 * x1=0 (=0) + 3 * x2=1 (=3) + -1 * gt_0_0=2 (=-2) == 1 !≤ 0

Inconsistency: Inconsistency in gt_0_0: 2 * x1=0 (=0) + 3 * x2=1 (=3) + -1 * gt_0_0=2 (=-2) == 1 !≤ 0

Inconsistency: Inconsistency in gt_1_0: 1 * gt_0_0=2 (=2) + 5 * x3=1 (=5) + -1 * gt_1_0=6 (=-6) == 1 !≤ 0


ModelConfig { scm: Rca, cutoff: None, decomposer: Gt(Coeff), equalize_ternaries: false, add_consistency: false, p
Extra solutions:
+ x1=0, x2=1, x3=1, gt_0_0=2, gt_1_0=6
+ x1=1, x2=0, x3=1, gt_0_0=2, gt_1_0=6
Missing solutions:

Expected assignments:
x1=0, x2=0, x3=0
x1=0, x2=0, x3=1
x1=0, x2=1, x3=0
x1=1, x2=0, x3=0
x1=1, x2=1, x3=0
Actual assignments:
x1=0, x2=0, x3=0, gt_0_0=0, gt_1_0=6
x1=0, x2=0, x3=0, gt_0_0=1, gt_1_0=6
x1=0, x2=0, x3=0, gt_0_0=2, gt_1_0=6
x1=0, x2=0, x3=0, gt_0_0=2, gt_1_0=6
x1=0, x2=0, x3=0, gt_0_0=3, gt_1_0=6
x1=0, x2=0, x3=0, gt_0_0=3, gt_1_0=6
```

Conclusion

- Different SAT encodings impact performance, but SAT solvers can be finicky
- Encoding size versus propagation strength
- Focus on simple constraints first, then decompose complex constraints


Abío, I., Mayer-Eichberger, V., and Stuckey, P. J. (2015).
Encoding linear constraints with implication chains to CNF.
In Pesant, G., editor, *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, volume 9255 of *Lecture Notes in Computer Science*, pages 3–11. Springer.

Abío, I., Nieuwenhuis, R., Oliveras, A., and Rodríguez-Carbonell, E. (2013).
A parametric approach for smaller and better encodings of cardinality constraints.
In Schulte, C., editor, *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 80–96. Springer.


Bailleux, O., Boufkhad, Y., and Roussel, O. (2006).
A translation of pseudo boolean constraints to SAT.
J. Satisf. Boolean Model. Comput., 2(1-4):191–200.


Bailleux, O., Boufkhad, Y., and Roussel, O. (2009).
New encodings of pseudo-boolean constraints into CNF.

In Kullmann, O., editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 181–194. Springer.

Bierlee, H., Dekker, J. J., Lagoon, V., Stuckey, P. J., and Tack, G. (2024).
Single constant multiplication for SAT.
In Dilkina, B., editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 21st International Conference, CPAIOR 2024, Uppsala, Sweden, May 28-31, 2024, Proceedings, Part I*, volume 14742 of *Lecture Notes in Computer Science*, pages 84–98. Springer.

Bierlee, H., Gange, G., Tack, G., Dekker, J. J., and Stuckey, P. J. (2022).
Coupling different integer encodings for SAT.
In Schaus, P., editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 19th International Conference, CPAIOR 2022, Los Angeles, CA, USA, June 20-23, 2022, Proceedings*, volume 13292 of *Lecture Notes in Computer Science*, pages 44–63. Springer.

Bofill, M., Coll, J., Suy, J., and Villaret, M. (2019).

SAT encodings of pseudo-boolean constraints with at-most-one relations.
In Rousseau, L. and Stergiou, K., editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4-7, 2019, Proceedings*, volume 11494 of *Lecture Notes in Computer Science*, pages 112–128. Springer.


Choi, C., Harvey, W., Lee, J., and Stuckey, P. (2006).
Finite domain bounds consistency revisited.
In *Proceedings of the Australian Conference on Artificial Intelligence 2006*, volume 4304 of *LNCS*, pages 49–58. Springer-Verlag.


Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., and Veith, H. (2000).
Counterexample-guided abstraction refinement.
In Emerson, E. A. and Sistla, A. P., editors, *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer.


Eén, N. and Sörensson, N. (2006).
Translating pseudo-boolean constraints into SAT.

J. Satisf. Boolean Model. Comput., 2(1-4):1–26.


Hölldobler, S., Manthey, N., and Steinke, P. (2012).
A compact encoding of pseudo-boolean constraints into SAT.
In Glimm, B. and Krüger, A., editors, *KI 2012: Advances in Artificial Intelligence - 35th Annual German Conference on AI, Saarbrücken, Germany, September 24-27, 2012. Proceedings*, volume 7526 of *Lecture Notes in Computer Science*, pages 107–118. Springer.


Joshi, S., Martins, R., and Manquinho, V. M. (2015).
Generalized totalizer encoding for pseudo-boolean constraints.
In Pesant, G., editor, *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, volume 9255 of *Lecture Notes in Computer Science*, pages 200–209. Springer.